FEDERAL UNIVERSITY OF PELOTAS
Centre of Technological Development
Postgraduate Program in Computing

Doctoral Thesis

Towards Teaching Abstraction: Approaching Modeling and Problem-Solving Skills with Graph Grammars and Game-Based Learning

Braz Araujo da Silva Junior

Pelotas, 2024

Braz Araujo da Silva Junior

Towards Teaching Abstraction: Approaching Modeling and Problem-Solving Skills with Graph Grammars and Game-Based Learning

Doctoral Thesis presented to the Postgraduate Program in Computing of the Centre of Technological Development from the Federal University of Pelotas, as partial requirement for obtaining the title of Doctor of Philosofy in Computer Science.

Advisor: Prof. Dr. Simone André da Costa Cavalheiro
Coadvisor: Prof. Dr. Luciana Foss
Collaborator: Prof. Dr. Anthony Robins

Pelotas, 2024

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação da Publicação

S586t Silva Junior, Braz Araujo da

Towards Teaching Abstraction [recurso eletrônico] : approaching modeling and problem-solving skills with graph grammars and game-based learning / Braz Araujo da Silva Junior ; Simone André da Costa Cavalheiro, orientadora ; Luciana Foss, coorientadora. — Pelotas, 2024.
235 f. : il.

Tese (Doutorado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2024.

1. Graph grammar. 2. Computing education. 3. Specification. 4. Abstraction. 5. Psychometrics. I. Cavalheiro, Simone André da Costa, orient. II. Foss, Luciana, coorient. III. Título.

CDD 005

Elaborada por Maria Inez Figueiredo Figas Machado CRB: 10/1612

Braz Araujo da Silva Junior

Towards Teaching Abstraction: Approaching Modeling and Problem-Solving Skills with Graph Grammars and Game-Based Learning

Doctoral Thesis approved as partial requirement for obtaining the title of Doctor of Philosofy in Computer Science, Postgraduate Program in Computing, Centre of Technological Development, Federal University of Pelotas.

Defense Date: October 7, 2024

Examination Board:

Prof. Dr. Simone André da Costa Cavalheiro (Advisor)
PhD in Computer Science by the Federal University of Rio Grande do Sul (UFRGS).

Prof. Dr. Luciana Foss (Coadvisor)
PhD in Computer Science by the Federal University of Rio Grande do Sul (UFRGS).

Prof. Dr. Tiago Thompsen Primo
PhD in Computer Science by the Federal University of Rio Grande do Sul (UFRGS).

Prof. Dr. Leila Ribeiro
PhD in Computing by the Technische Universitat Berlin (TUBerlin).

Prof. Dr. Dilma da Silva
PhD in Computer Science by the Georgia Institute of Technology (GT).

To the worm who first gnaws on the cold flesh of my corpse, I dedicate with fond remembrance this thesis. That is an adaptation of a classic book of the Brazilian literature, whose protagonist has the same name as me. Beyond the coincident names, I use this dedicatory to highlight the power and relevance of science contributions and knowledge sharing. While my body shall perish at some time, this work shall prevail.

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor, Simone André da Costa Cavalheiro, along with my co-advisor, Luciana Foss, for guiding my path since the very first step I took towards science. For literally making me a scientist, something that back in the day I would just admire and wonder how far from me it was. For feeding my dreams on our projects while holding me back to reality, keeping a balance of motivation and viability. For carefully going along every character of our scientific papers and this work, while suggesting changes and corrections. That is an exceptional job, worth an exceptional acknowledgment.

Second, I need to thank Anthony Robins and CAPES for allowing us to bring our research to New Zealand. Harnessing Dr. Robins' psychology background was an amazing and fruitful experience, leading us to the whole psychometric development in this work. I thank the University of Otago as a whole for receiving me and the IS/CS departments for all the seminars, lab work and invaluable contributions.

My family makes me very grateful too. My mother, Edna, and my brother, Bruno, who both stood by my side and supported me at all times. In memory of my father, whose name I inherited. And my newborn nephew, Conrado, who brought happiness and life back into our paths. A special thanks to my first love and partner, Jean, for adventuring himself across the globe with me during the course of this work. In addition to the social and emotional support, presence and help with daily life. A big, state-crossing thanks to my dearest friend, Paulo, who kept in touch despite the physical distance between us and now accompanies me on a new journey, partially resulting from this work.

I thank too all my colleagues, fellow scholars and professors of the Centre for Technological Development of UFPel who I had the opportunity to work with during this work. Namely: Ana Pernas, with the Computational Thinking database project; Larissa Freitas, with the Text-To-Graph Grammar project using natural language processing; Patricia Maillard, with the contributions to the pedagogical agents' development; and Tiago Primo, with the Robopel events and all the important insights from the creative learning perspective.

A thanks to the next generation of scientists and the students I've oriented throughout the programme: Julia Veiga, who worked with GrameStation from the start and pushed important projects such as User Experience and Usability tests, the User Interface Pedagogical Agent and automata activities; and Wagner Loch, who started his doctoral studies interested in working with GrameStation and artificial intelligence. As well as the rest of the students who have worked with GrameStation in various ways.

A special thanks to all those who were my virtual students during the pandemic and inspired me to create the PQNACOMP series. And, of course, the little students who

have been on the other end of this work, collaborating with our research and training their layers of abstraction skills. Lastly, a thank you to ExpPC volunteers who have helped with some of the activities reported here.

This work was supported by CAPES – Brazil – Financing Code 001, SMED/Pelotas, PREC, and PRPPG/UFPel.

Measurement is the first step that leads to control and eventually to improvement. If you can't measure something, you can't understand it.
— H. JAMES HARRINGTON

ABSTRACT

SILVA JUNIOR, Braz Araujo da. **Towards Teaching Abstraction: Approaching Modeling and Problem-Solving Skills with Graph Grammars and Game-Based Learning**. Advisor: Simone André da Costa Cavalheiro. 2024. 235 f. Thesis (Doctorate in Computer Science) – Centre of Technological Development, Federal University of Pelotas, Pelotas, 2024.

Computing education has risen and evolved, conquering new spaces and acquiring recognition from governments and institutions around the world. Beyond the consistent and increasing demand for professionals in Information and Communications Technology, computing pervasiveness called for its introduction in basic education, for all citizens. Powerful education trends, such as Computational Thinking (CT), problem and project-based learning, maker's culture, gamification and educational games progressed powered by the advance of computing. A set of problem-solving skills based on computing is how CT was conceptualized and gained popularty. However, the difficulty in defining and treating those skills, such as abstraction, algorithmic thinking, decomposition and pattern recognition has made CT manifest mostly through programming. This reignites an old concern, computing education has struggled to demystify being equated to programming. Computing being far more than the technical competence to code is the main argument for its introduction in general education. And this is important even for programming, which should be a step deeper into computing, not the first. In this regard, this work is an effort to advance the knowledge and operationability of CT, treating specifically abstraction, centered on the iconic form it presents itself in computing: the layers of abstraction. The proposed approach revolves around using Graph Grammars (GG) to specify and play "graph games" (grames, for short), which is the concept of "graming". It includes the development of: an educational game engine based on GGs; tools to manage layers of abstraction based on Hierarchical GG; and an assessment grame to evaluate competencies related to layers of abstraction, under the principles of the Evidence-Centered Design (ECD), guiding the creation of proficiency, task and evidence models. This whole ecosystem showed that it is possible to approach deep, fundamental and abstract skills, such as abstraction itself, as solid psychometric constructs to be reliably operationalized while preserving creative and engaging environments. This work also opened up a wide range of future applications and investigations when it brought GG from its formal specification origins to game-based learning in basic education. At last, the engine reinforces the emerging potential of educational tools when powered by AI.

Keywords: Graph Grammar. Computing Education. Specification. Abstraction. Psychometrics.

RESUMO

SILVA JUNIOR, Braz Araujo da. **Rumo ao Ensino de Abstração: Trabalhando Habilidades de Modelagem e Resolução de Problemas Utilizando Gramática de Grafos e Aprendizado Baseado em Jogos**. Orientador: Simone André da Costa Cavalheiro. 2024. 235 f. Tese (Doutorado em Ciência da Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2024.

O ensino de computação cresceu e evoluiu, conquistando novos espaços e o reconhecimento de governos e instituições de todo o mundo. Além da consistente e crescente demanda por profissionais de Tecnologia da Informação e Comunicação, a difusão da computação clamou por sua introdução na educação básica, para todos os cidadãos. Tendências poderosas da educação, como o Pensamento Computacional (PC), aprendizado baseado em problemas e projetos, cultura maker, gamificação e jogos educacionais progrediram impulsionados pelo avanço da computação. Um conjunto de hablidades de resolução de problemas é como o PC foi concebido e ganhou popularidade. No entanto, a dificuldade em definir e tratar tais habilidades, como abstração, pensamento algorítmico, decomposição e reconhecimento de padrões tem feito o PC se manifestar majoritariamente através de programação. Isto reacende uma preocupação antiga, o ensino de computação tem lutado para desmistificar o fato de ser equiparado à programação. A computação ser muito mais do que a competência técnica para programar é o principal argumento para sua introdução no ensino geral. E isso é importante até mesmo para programação, que deve ser um passo mais adiante na computação, não o primeiro. Neste sentido, este trabalho é um esforço para avançar o conhecimento e a operacionalização do PC, tratando especificamente abstração, centrada na forma icônica que ela se revela na computação: as camadas de abstração. A abordagem proposta gira em torno do uso de Gramática de Grafos (GG) para especificar e jogar "jogos de grafos" (*grames*, abreviação do inglês *graph games*), que é o conceito de "*graming*". O trabalho inclui o desenvolvimento de: um motor de jogos educacional baseado em GG; ferramentas para manusear camadas de abstração baseadas em GG Hierárquicas; e um *grame*-teste para avaliar competências relacionadas a camadas de abstração, sob os princípios do Design Centrado em Evidências (ECD), guiando a criação de modelos de proficiência, tarefa e evidênca. Todo este ecossistema demonstrou que é possível abordar habilidades complexas, fundamentais e abstratas, como a abstração em si, como construtos psicométricos sólidos para serem confiavelmente operacionalizados enquanto mantemos os ambientes criativos e atraentes. Este trabalho também abriu uma vasta gama de aplicações e investigações futuras ao trazer GG de suas origens na especificação formal para o aprendizado baseado em jogos na educação básica.

Por fim, o motor de jogos reforça o imenso potencial emergente de ferramentas educacionais ao serem potencializadas por inteligência artificial.

Palavras-chave: Gramática de Grafos. Educação em Computação. Especificação. Abstração. Psicometria.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
b-learning	Blended Learning
CEdR	Computing Education Research
CPS	Cyber-Physical Systems
CS	Computer Science
CSEd	Computer Science Education
CSU	Computer Science Unplugged
CT	Computational Thinking
DIY	Do It Yourself
DL	Digital Literacy
e-learning	Electronic Learning
FC	Flipped Classroom
GG	Graph Grammar
GGasCT	Graph Grammars for developing and assessing Computational Thinking
GPT	Generative Pre-training Transformer
Grame	Graph Game
GUI	Graphical User Interface
IoT	Internet of Things
IT	Information Technology
ICT	Information and Communication Technologies
k-12	Kindergarten to Twelve
LHS	Left-Hand Side
LLM	Large Language Model
LoA	Layers of Abstraction

MOOC	Massive Open Online Course
moodle	modular object-oriented dynamic learning environment
NLP	Natural Language Processing
OOP	Object-Oriented Programming
PA	Pedagogical Agent
PACT	Partnership for Advancing Computational Thinking
PBL	Problem Based Learning
PjBL	Project Based Learning
PC	Personal Computer
PS	Proposed Solution
RQ	Research Question
RH	Research Hypothesis
RR	Research Result
RHS	Right-Hand Side
SBC	Brazilian Society of Computing
SG	Specific Goal
SLR	Systematic Literature Review
TA	Think Aloud
TPL	Textual Programming Languages
TUI	Tangible User Interface
UML	Unified Modeling Language
UX	User Experience
VPL	Visual Programming Languages
VR	Virtual Reality
XML	Extensible Markup Language
XSD	Extensible Markup Language Schema Definition
WYSIWYG	What you see is what you get

CONTENTS

1 INTRODUCTION

Chapters distribution. Opening this work, this first chapter introduces the reader to the problem and provides information about how the text was structured and formatted. The second chapter presents what has been found in the current literature around the theme. The third chapter elucidates the direction we took to solve the problem and how we built the platform our approach takes place. The fourth chapter brings the process of development of specific tools that were added to the platform to support our approach. The fifth chapter describes the approach and reports the experiment. The sixth section concludes the work discussing contributions and future work.

Content distribution. In this chapter, the first section contextualizes and situates it within the fields of education and computing. The second section characterizes it and clarifies why it is a problem. The third section objectively states the research questions driving this work, its goals and what is being proposed to pursue them. Additionally, the following paragraphs before the first section explain how the thesis text was formatted and which notations could be expected by the reader.

Textual resources. Each paragraph is named after the topic it addresses, they always start with their names highlighted as **bold text**. Acronyms always appear in their first use as their extended form highlighting with **bold text** the letters that compose the abbreviated form, which immediately follows it between parenthesis. For instance, **C**omputer **S**cience **(CS)**. They are also listed in alphabetic order in the page before the table of contents.

Default paragraphs. Chapters start with a paragraph called **Content distribution**, describing the purpose of the chapter and each of its sections. Chapters end with a paragraph called **Recap**, summarizing the content previously presented in the chapter. Sections start with a paragraph called **What can I learn here?**, informing the reader what to expect, the purpose of the section and why it is there.

1.1 Context

What can I learn here? This section offers general notions on the fields of computing and education, individually and their intersections. It is taken the opportunity to present the reader with a series of terms and expressions from these fields that will be explored throughout this work. As well as to differ similar terms that can be rather confusing or used interchangeably on our daily lives or even in the literature. If the reader is familiar with these fields, they might want to simply skim this section.

1.1.1 Computing in the 21st century

Computing. There might be several definitions for it already, with different ranges, scopes and depths. This work will use the term "computing" as broadly as it can be, involving everything related to computations, computer systems and information processing. This includes their products, impacts and consequences on society, economy, science and many more as will be discussed in the following paragraphs. Respecting the history of computing, back when "computers" were human beings, it is considered that: a computer is "An information processing agent that is able to perform a set of computations"; where a computation is any "set of linked/chained calculations"; and calculations are "processes that transforms one or more inputs into one or more results" (SILVA JUNIOR, 2020). Therefore, computing involves as much of techniques for solving and analyzing problems (computations) as the crafting and use of machines to support the solutions (computer machines). This work considers as part of computing all phenomena surrounding those as well. The **B**razilian **S**ociety of **C**omputing **(SBC)** proposes three axes to organize computing that provides a glance of its breadth: **C**omputational **T**hinking **(CT)**, towards the problem-solving aspect; Digital World, towards the machinery aspect, the whole ecosystem of machines, programs and data that computing brought to life; and Digital Culture, towards the interaction of the society with the digital world (RAABE et al., 2017).

Disruptive Technology. Computing brought to the world various technologies that drastically changed entire markets, how we live our daily lives, how we communicate, and even how we solve problems. Digital transport services now bring you a personal driver in minutes after a couple of touches on your smartphone. Video streaming services resignified multimedia content creation and freed us from scheduled content, allowing us to choose when and what to watch. Online Lodging services introduced a much more versatile concept of accommodation, not only facilitating booking hotels, but also allowing alternatives such as residential houses' room rents for short stays. Cross-platform instant messaging services became the main form of personal and even professional communication for many, if not most, of us. Online encyclopedias are regularly consulted as our first source of information we are not familiar with. E-

commerce boosted importation, heated up and diversified all kinds of markets (SHAW, 2015). All these technologies are so ingrained in our society nowadays that I bet at least one organization or brand name came to your mind when reading each of them.

Ubiquity. Humanity warmly embraced computers. Since the desktops there was already the idea of having a **P**ersonal **C**omputer **(PC)** at home. Not much later, technological advancement gave the idea of having pocket computers (smartphones) to carry everywhere. Consequently, with the necessary hardware present in most companies, homes and pockets, a big software boom started the digital age, allowing us to access and interact with a digital world. Taking into account that computers are everywhere at all times, that each person has access to and interacts with many computers every day, is called ubiquitous computing (TERZIMEHIĆ, 2021). This is here to stay, new technology advancements, such as **I**nternet **o**f **T**hings **(IoT)**, domotics (residential automation) and smart cities (ROSE; ELDRIDGE; CHAPIN, 2015; SIMONET; NOYCE, 2021; KIM et al., 2021), reveal that ubiquitous computing is only being pushed even further.

Cultural transformation. With such a presence in our lives, it is expected that computing would impact our culture. The cult of technological gadgets gave birth to a whole new cultural tribe, the “geeks”, while programming redesigned the stereotype of “nerds” (WOO, 2018). By the way, social media invented a new way of bullying, the “cyberbullying” (ZHU et al., 2021), which is a good reminder that computing brought many new things, but unfortunately, not all of them are good. On the other hand, digital games leveled up the entertainment industry, not only turning into really popular hobbies, but also establishing a new kind of sport (VAUDOUR; HEINZE, 2020). Known as e-sports, digital game tournaments conquered professional space and raised huge communities around millionaire events (ABANAZIR, 2019). Above it all, we created our digital identities across social media platforms and revisited our concept of sharing. Either for publicly sharing our moments, feelings, thoughts, projects and creations, or for sharing knowledge, content and resources we found somewhere (ZINGALE, 2013).

Big Data. A society with computers everywhere, being used by everyone, at all times, for basically everything: work, study, socialization, entertainment, payments and shopping. It is no surprise that a humongous amount of data would be generated. Even less of a surprise that we would be storing, processing and commercializing these data. Consequently, it raises critical ethical concerns on privacy (STAHL; WRIGHT, 2018) and mandatory discussions on safety and law regulation (BRAYNE, 2018). While for the regular citizen, it means being used to filling forms, creating profiles and accounts, being suggested content they are likely to enjoy and products they are likely to buy.

Information Technology. Providing this digital society with the means to access, use and maintain computers, systems and data running non-stop requires a great contingent of trained computing professionals. This professional, commercial and technical field dedicated to computational systems development and maintenance is what

this work refers to as **I**nformation **T**echnology **(IT)**. The amount and variety of IT jobs have been rising and keeping high employability, examples of such IT occupations are: computer and information research scientists; network architects and administrators; programmers; support specialists; system analysts; database administrators; information security analysts; software developers; and web developers (HUSSEIN; TRAUTMAN; HOLLOWAY, 2021). Besides that, the increasing automation of services comes with the fear of mass unemployment, which is much more likely to rather be a mass displacement of workforce (BESSEN, 2015), but a major concern for professionals anyway. As one could expect, IT has been considered one of the less likely professional fields to disappear (or be replaced) in the near future (FREY; OSBORNE, 2017).

Future Disruption. Not just jobs, the disruptive power of computing to radically transform various aspects of our lives is far from over. Many technologies are taking promising and exciting directions, such as: 3d printed organs and tissues for surgeries (QIU; HAGHIASHTIANI; MCALPINE, 2018); **V**irtual **R**eality **(VR)** for distance tourism (MERKX; NAWIJN, 2021); wearable computer glasses/lens (DANIELSSON; HOLM; SYBERFELDT, 2020) for ubiquitous **A**ugmented **R**eality **(AR)** (XIONG et al., 2021); IoT integrated autonomous cars (AHMAD; POTHUGANTI, 2020) for yet another transport revolution; social/domestic robots for emotional support (BAECKER et al., 2020; ROSSI et al., 2022); and **A**rtificial **I**ntelligence **(AI)** for creating art (CETINIC; SHE, 2022; RAMESH et al., 2021). Meaning that many big transformations are yet to come, and not even social and creative tasks are out of reach for the still ongoing computing revolution.

Computer Science. The mother of these life-changing technologies is science. Computing has generously contributed to many areas of science, bringing powerful simulations to biology, chemistry and physics; computational power enabling superhumanly fast calculations to mathematics; large-scale data gathering and processing to humanities; and its own dedicated space within science. The field of study concerned with information phenomena, including its modeling, computation and automation, is what this work refers to as **C**omputing **S**cience **(CS)**. An academic area, focused on theoretical development and scientific experiments. Fairly related to IT, since most CS-related courses also prepare for IT jobs. Noteworthy, there are other definitions for CS and IT, the ones used in this work are meant to make a clear distinction between different contexts: academy and industry.

Information and Communication Technologies. Another similar term this work care to differ is **I**nformation and **C**ommunication **T**echnology **(ICT)**, which is commonly used in the education field referring specifically to the products of computing and telecommunications (FU, 2013). From radios, televisions, projectors and smartphones to video conference platforms, e-books, **m**odular **o**bject-**o**riented **d**ynamic **l**earning

environment **(moodle)**-like systems and **M**assive **O**pen **O**nline **C**ourses **(MOOC)**. A term this work won't differ and will avoid using is "Informatics", since it may be found as broader, narrower or synonymous to computing, CS, IT or ICT, in addition to meaning variation across languages.

Protagonist of the century. At this point, you shall have been convinced that computing was quite impactful across many sectors of our society. Economy was shaken as various businesses were revolutionized by disruptive technology. Culture was transformed as we entered the digital world. Law and ethics were pressed as technology and data escalated beyond privacy and safety. Industry was reformed as automation displaced workers. Science was expanded as a new field of study emerged providing the others with powerful tools. As for education, since it is the focus of this work, it will be investigated more thoroughly ahead. But it should already be clear that, despite the impacts of computing directly on education, a society where computing has reached such unparalleled pervasiveness simply demands computing education. How can we live alienated from the processes surrounding us every second, that made it up to our very pockets?

1.1.2 Education in the 21st century

Education. Just like "computing", "education" is also found defined in a myriad of ways, considering various different perspectives. This work will again use the term as broad as it can be, considering that "education" involves everything related to teaching and learning. This includes the resources used for it, the institutions and environments where it happens, the methodologies followed, the people that make it happen (not just the teachers and the learners), the logistics enabling it, and so on. It is a worldwide agreement that education is a right for everyone, and shall be compulsory and free, at least in its elementary stage, as explicitly put in the Universal Declaration of Human Rights (UNITED NATIONS, 1948). What is "elementary" in education though, raises ever-renewing discussions (BRIGHOUSE, 2006). In the end, what are the aims of compulsory education? Literate individuals with a reasonable understanding of how the world around them works? According to science, religion(s) or both? Skilled soon-to-be professionals on their way to become productive workers? Informed citizens aware of their duties and rights ready to start a life of their own, assuming the responsibility for their actions? It is not in the scope of this work to answer such controversial questions, but it will be assumed that all those have their space in education (not necessarily compulsory) and thus, shall be taken in consideration.

Stages of Education. What is considered compulsory to learn and to complete by means of grade/years of education varies from government to government. The **I**nternational **S**tandard **C**lassification of **Ed**ucation **(ISCED)** brings guidelines for governments around the world to organize their education systems (UNITED NATIONS,

2011). Based on the document, this work will make use of the terminology detailed in the Table 1, indicating the term, typical entry and egress age ranges, institutions in charge and the aims of the stage. In addition to those in the table, this work will use the following terms: Basic Education, referring to early childhood, primary and secondary education altogether (ISCED levels 0-3), which is also known as **K**indergarten to (grade) **T**welve **(k-12)**; Higher Education, as synonym to tertiary education plus technical schools (ISCED levels 4-8); and **H**igher **E**ducation **I**nstitution, referring to any institution offering higher education courses.

Table 1 – Stages of Education based on ISCED (UNITED NATIONS, 2011).

Lvl	Stage of Education	Age	Institutions	Aims
0	Early Childhood	5-	nursery school, pre-school.	Early development.
1	Primary	6-10	primary/elementary school.	Literacy (Reading, Writing, Arithmetic).
2	Lower Secondary	11-14	junior secondary/middle school.	Foundational knowledge, generic subjects.
3	Upper Secondary	15-17	senior secondary/high school.	Broader range and choice of subjects.
4	Post-Secondary Non-Tertiary	17+	technical school.	Direct labour market entry.
5	Tertiary - Short Cycle	17+	HEI, university.	Professional and scientific maturity.
6	Tertiary - Bachelor's			
7	Tertiary - Master's			
8	Tertiary - Doctoral			

21st Century Skills. When it comes to the 21st century, education has been largely influenced by technological advancement. That is because the challenge of preparing the population to keep up with this advancement is for education to assume. As a result, a unified effort involving governments and businesses led to the creation of a framework for developing the skills, aptitudes, and attitudes to succeed in this emerging workplace and society: the 21st-century skills. It includes the following competencies: learning skills, such as creativity, innovation, critical thinking, problem-solving, communication and collaboration; literacy skills, such as information literacy, media literacy and ICT literacy; and life skills, such as flexibility, adaptability, initiative, self-direction, social and intercultural skills, productivity, accountability, leadership and responsibility (GONZÁLEZ-PÉREZ; RAMÍREZ-MONTOYA, 2022).

Education 4.0. Tightening ties with technology, education makes a direct reference to the Industry 4.0, which is the era initialized by the fourth industrial revolution. After steam engines, electric power and electronics causing each a new revolution, the **C**yber-**P**hysical **S**ystems **(CPS)** are causing the fourth, blurring the boundaries between the physical, digital and biological worlds (HUSSIN, 2018) through AI, IoT, robotics, 3-D printing, nanotechnology, cloud and quantum computing, among others. Responding to that, Education 4.0 is referred as a period in which education is dominated by digital transformation and innovation, interacting with those Industry 4.0 technologies (KESER; SEMERCI, 2019). Delving into the concept, this work elaborates on 9 of its trends (HUSSIN, 2018) to elect as principles of Education 4.0 the following notions of technology supported:

1. **Ubiquity**, extending education beyond classrooms;

2. **Personalization**, tackling limitations of “one fits all” models;
3. **Flexibility**, promoting variety of options on assignment and diversity on delivery;
4. **Project**, encouraging organizational, collaborative and time management skills while applying knowledge;
5. **Hands-on Learning**, immersing students into real-world environments and situations;
6. **Data Interpretation**, preparing for a data-driven society;
7. **Dynamic Assessment**, refining judgement upon learning through multiple sources and forms;
8. **Student Participation**, granting active voice over their education;
9. **Role Blending**, bridging both ends of the teaching-learning process.

Active Methodologies. It is a clear trend of these principles to increase student independence. It is part of the paradigm shift from a teacher-centered education to a student-centered one, transitioning the focus on teaching to a focus on learning. Aligned with this shift comes the concept of active learning, “the process of acquiring knowledge, skills, values and attitudes by any educational strategy that involves or engages students in the process by leading them to activities and debates, instead of just putting them in the position of passively listening to the information given by the teacher” (KONOPKA et al., 2015). In turn, active methodologies are a set of techniques supporting active learning, efforts to give the student a greater role in their education; foster collaborative work; organize teaching based on the competencies to be acquired; and stimulate the acquisition of autonomous, permanent learning (MOYA, 2017).

Problem Based Learning. A first example of that is the **P**roblem **B**ased **L**earning **(PBL)**. In general terms, it is the idea of providing a problem that requires the knowledge or its application to be solved, instead of providing the knowledge itself. Its origin is credited to medical schools, where students were presented with hypothetical clinical boards and expected to: identify relevant information; organize their prior knowledge; question based on self-identified gaps in their knowledge; complement their knowledge with independent research; discuss their findings and theories with colleagues; and achieve an according diagnosis and treatment recommendation. When adopting PBL, teachers shall change their roles from lecturers to mediators, monitoring discussions, intervening when appropriate, asking convenient driving questions, raising issues for consideration and fostering full and even participation (ALLEN; DONHAM; BERNHARDT, 2011).

Project Based Learning. The second example of active methodologies share initials with the first, **P**ro**j**ect **B**ased **L**earning **(PjBL)**. In general terms, it is the idea of providing on-demand knowledge to support students' production of artefacts that show application of the knowledge. The main difference to PBL is that PjBL: needs to culminate in an end product; and instead of giving the students a very specific problem to work with, they are given a rather wider subject or theme. On one hand it weakens the problem-solving training, loosening the solution requirement to meet specific needs. On the other hand, such openness boosts creativity training and potentially engagement, allowing the student to come with solutions that are meaningful to them (KOKOTSAKI; MENZIES; WIGGINS, 2016).

Flipped Classroom. The third example of active methodologies emerges from new education settings, that turned easy to invert traditional cycles, the **F**lipped **C**lassroom **(FC)**. In general terms, it is the idea of activities traditionally conducted in the classroom becoming home activities, and activities normally constituting homework becoming classroom activities (BERGMANN; SAMS, 2012). Instead of merely delivering information, teachers use classroom time to engage in discussions, solving problems proposed by the students, hands-on activities, and offer guidance. While the students are able to govern their learning adjusting it to their own pace (AKÇAYIR; AKÇAYIR, 2018). This model gained attention only recently due to lectures being the main in-class activity, and attending to lectures at home becoming a reality only with the support of ICTs, moodles and MOOCs.

Covid Pandemic. An unfortunate compulsory booster of ICTs and virtualization of education was the Covid-19 pandemic. The outbreak of the novel coronavirus forced various education settings to adapt to mandatory social distancing and sanitary measures. Just in 2020, at least 102 countries around the world closed their schools to stop the virus, making about 900 million children and youngsters stand away from school (SARI; NAYIR, 2020). It put to the test and revealed unforeseen challenges of many promising educational approaches that used to believe that the popularization of technology was the thing keeping them from flourishing, such as: home schooling (AZNAR et al., 2021) and distance education (PREGOWSKA et al., 2021). The long-term impacts of the pandemic are yet to be discovered, but its effects on education could be devastating for some countries, specially for basic education (HUCK; ZHANG, 2021). Higher education suffered with a series of adaptations for distance education, mainly on assessment, but at least most of it wasn't completely interrupted (SARI; NAYIR, 2020). This forced adaptation popularized some expressions, such as: asynchronous activities, assignments given to the students for they to complete at any time up to a due date, usually on their own, e.g. homework; synchronous activities, assignments that students must attend at the same time of others, e.g. live meetings; **E**lectronic **L**earning **(e-learning)**, the online learning based on virtual meetings and digital plat-

forms, where the communication between teacher and learner is through ICTs; and **B**lended **L**earning **(b-learning)**, the hybrid system including traditional face-to-face meetings and online assignments using moodles and other ICTs.

Creative Learning. The world being required to reinvent itself due to tragic events recalls us of one of the 21st-century skills: creativity. No wonder it is the driver of another huge trend in education: creative learning. It relies on the idea of an interdependent relationship between creativity and learning, i.e. creativity impacts learning and learning impacts creativity (BEGHETTO, 2016). Studies have indicated this positive relationship between measures of creativity and academic achievement, yet some point that schools actually suppress or even kill student creativity (BEGHETTO, 2021). A reasonable explanation for this is that schools traditionally avoid uncertainty in the name of assessment reliability, while uncertainty is a must for creativity. If one already knows what to do and how to do it, then there is no room for creativity, they are just rehearsing prior knowledge (BEGHETTO, 2021). Notwithstanding, creative learning is conquering space as something aligned with 21st century skills, Education 4.0 principles, PjBL and ICTs.

Maker Culture. Speaking of creativity, projects and technologies, we are witnessing a rise of hobbies involving crafting, manufacturing, hacking, fabricating and making a wild variety of tools, from toys to furniture. The so-called **D**o-**I**t-**Y**ourself **(DIY)** movement flourished due to the popularization of manufacturing machines and low-cost electronic devices, sensors and systems, as well as the internet connecting communities to share their knowledge with their peers (NASCIMENTO; PÓLVORA, 2018). In educational environments, the DIY phenomenon contributed to reinforce protagonism of students and encourage PjBL inside the classroom (SORMUNEN; JUUTI; LAVONEN, 2020). And beyond it, makerspaces, also known as hackerspaces, FabLabs or creative spaces, are laboratories dedicated to crafting and hands-on learning, often equipped with 3d printers, laser cutters, computers and all kinds of machines to support a variety of fabrications. They are a successful example of education outside of the school that have been increasingly explored in the literature (MERSAND, 2021).

Large Language Models. Another huge impact from computing technologies to education beyond school is the **L**arge **L**anguage **M**odel **(LLM)** boom, such as the **G**enerative **P**re-training **T**ransformers **(GPT)**. With the potential to cause a revolution on autodidatic experiences (FIRAT, 2023), they use **N**atural **L**anguage **P**rocessing **(NLP)** to understand and answer users as they were chatting with another person. LLMs have recently reached important milestones and have shown to excell at writing, mathematics and specialized fields such as medicine, macroeconomics and psychology (OPENAI, 2023). Despite their capabilities as a tutoring tool, their ample availability and indiscriminate use raises reasonable concerns for education (RAHMAN; WATANOBE, 2023). Not only it can be difficult to tell if an assignment was made by the AI alone or

the student, but they are also not always trustworthy, being able to confidently spread misinformation or conform to user's wrong information just to flatter them (LIU et al., 2023; DEMPERE et al., 2023).

Disruptive education. The 21st century proved to have a thing for disruption. Education followed the same path, seeking to break with the established model, pursuing a series of innovations and new perspectives. It learned new ways of learning and took lessons out of tragedies. It advanced to follow technological progress, but the wonders of the digital revolution came with the responsibility of computing education.

1.1.3 Computing and education

Computing education. A field of study regarding teaching and learning of "computing" as defined in this work would include **D**igital **L**iteracy **(DL)**/inclusion, CS, IT and ICT training. Which is comparable to **C**omputer **S**cience **Ed**ucation **(CSEd)**, since computing and CS are often used interchangeably in the literature. It is hard to say if/when the literature use the term CSEd restricting it to CS as defined here. There is also the term **C**omputing **E**ducation **R**esearch **(CER)**, highlighting it is not about the teaching-learning itself, but the underlying theories or resulting discoveries. Therefore, just like CS and IT were differed by a focus on academy and industry, respectively, this work differs CSEd and CER by a focus on practice and theory respectively.

Computational thinking. Specially targeting basic education, CER has adopted the concept of CT since it was defended it would benefit everyone, not just computer scientists (WING, 2006). A seemingly endless debate on its definition has been carried ever since, without reaching a consensus (KALELIOGLU; GULBAHAR; KUKUL, 2016), but producing several frameworks and models (TIKVA; TAMBOURIS, 2021). A **S**ystematic **L**iterature **R**eview **(SLR)** summarizes from 36 general definitions for CT that it "is often depicted as a mental activity to solve problems, ranging from using computers, computation, fundamental concepts or methods of CS, to designing the solution in a way a computer could run it or at least be used to carry out some help" (SILVA JUNIOR, 2020). Which "fundamental concepts of CS" CT considers varies from author to author, being abstraction and algorithmic thinking the most consistent in the literature. What is interesting to note about CT is that its growth is heavily tied to the argument of the benefits of teaching computing to all people, not just to those who will become CS/IT professionals.

Our call. Reasonably, CT can be seen as the very manifestation of the call for a compulsory CSEd this era demands, incorporating the education trends this era cultivated: often understood as an extensive set of abstract skills and attitudes, similar to 21st century skills; directly connected to and surrounded by exciting technologies, similar to Education 4.0; conceptualized around general problem solving, similar to PBL; commonly approached by activities leading to artefact generation, similar to PjBL; en-

couraging open-ended problems and evaluation/reflection upon the solutions, similar to creative learning; and gathering communities to share and remix resources, similar to the maker culture. Consequently, the success and concentration of CT in basic (compulsory) education seems only natural.

Computing training for professionals. Those are just the first steps of CSEd into basic education. But CSEd has been long acting on higher and technical education. In addition to the obvious duty of educating CS and IT professionals, CSEd eventually takes on the challenge to educate workers from all kind of fields about CS subjects. Basically, that generous contribution of CS to all science comes with a liability. The physicists, chemists, biologists, medics, mathematicians, neuroscientists and many other non-CS professionals that could benefit from running some simulations, data processing or calculations won't be able to do so without proper education. Teaching algorithms, programming, computational calculus and simulation to non-CS professionals is also a concern of CSEd.

Computational tools for education. CSEd assuring other fields benefit from CS means and subjects is just part of the contribution. What they benefit the most is from CS products. Part of CER is dedicated to developing computational tools for education as a whole, not just CSEd. These tools include automated assessment (BARANA; MARCHISIO; RABELLINO, 2015), digital educational games and gamified environments (JANTAKUN; JANTAKOON, 2021), tangible interfaces (SCHKOLNE; ISHII; SCHRODER, 2004) and virtual learning platforms (KLIZIENE et al., 2021). Not to mention the ICTs being widely adopted into the classrooms, which are less natural to the teachers from the previous generations than it is to the learners of the newer generations. This intergenerational problem of teacher training is another major concern of CSEd (DEMETRIADIS et al., 2003).

Technology infused education. Computing and education have influenced each other more and more. Now they meet in the middle of a disruptive era of transformations. The CT movement comes embedding several education trends to break the established view of CSEd for CS only, in an attempt to bring it to basic education. The idea that CSEd is not restricted to CS extends to higher education and culminates on its contributions for education as a whole through computational tools. However, the discrepancy in the speed between technology generations and human generations raises additional challenges for CSEd these days.

1.2 Problem Statement

What can I learn here? This section takes the reader from the broad fields and concepts presented in the previous section to the specific problems we are tackling, narrowing down the scope each subsection. Here the reader can understand what the

problem is, where it comes from, why it is important and how it fits a bigger picture. The more familiar with the problematics, the more the reader can skim this section.

1.2.1 The challenges of computing education

Novice programmers. A commonly addressed problem of CSEd in higher education is about retention, high dropout rates in CS courses, despite the great demand for CS professionals (GIANNAKOS et al., 2017). This is perceived to be concentrated on early semesters and introductory subjects such as “CS1”, which might be connected to the first contact of the students with programming (ROBINS, 2019). Therefore, let us focus on the introduction of programming. What is difficult about learning how to program? A SLR (MEDEIROS; RAMALHO; FALCÃO, 2018) captured the main challenges faced by introductory programming students as being: **problem-formulation**, understanding and conceptualizing the problem being addressed; **abstraction**, dealing with concepts that cannot be easily related to a real-life object, such as variables, data types and memory addresses; **algorithmic reasoning**, organizing thoughts in a systematic, well-defined way, for a machine to be able to execute it; **syntax**, transforming an informal solution or even a pseudo-code into a syntactically correct program; **control and data structures**, selecting the best fit for a given problem according to their properties; **motivation and engagement**, being personally interested and willing to learn.

Teacher training. When looking beyond CS, CER is busier than ever with teacher training. With CT slowly becoming a must in basic education, teachers must be trained to support it. And there comes loads of different entrainments, focusing on robotics (SCHINA; ESTEVE-GONZÁLEZ; USART, 2021); Scratch (PAPADAKIS; KALOGIANNAKIS, 2019); mobile educational games (MOLIN, 2017); AR (POMBO; MARQUES, 2021); CT problem solving skills (YADAV et al., 2017). Not to mention the ICT boom (pushed even further by the pandemics) that also requires training (POZO-RICO et al., 2020). Too much to learn, too much to teach. As CSEd fights curriculum overload to introduce computing, teachers fight training overload to keep up with education trends. Perhaps the answer is not proposing new trainings for everything that is being created, but minimizing the training required for such things. How could this training be minimized?

Smarter educational tools. Computing is used to generating powerful tools harnessing the power of automation, which could be a convenient solution. How much of the teacher training could we embed into learning tools and platforms? And how much teacher training is necessary if the students are automatically guided by the tools? We sure should be aware of the risks of mechanizing education, but modern applications are no longer restricted to accepting a fixed single answer as the correct one; and they are more connected/social than ever. We now have the tools to provide flexible,

smart digital environments that can adapt on-demand to different user profiles (MIRAZ; ALI; EXCELL, 2021), offer comprehensive analysis (VIBERG; KHALIL; BAARS, 2020) and insightful feedback (DEEVA et al., 2021). This way, we alleviate teacher training overload by transferring part of their deeds to software.

Automation as an ally, not foe. From teachers' perspective, this should be seen as an expansion of capabilities instead of a loss of space, since personalized assessment of several students per teacher is unfeasible. From students' perspective, this should be seen as putting them on the center of their own learning, a natural alignment with the 21st-century skills (such as adaptability, initiative and self-direction) and the 9 trends of Education 4.0 we mentioned (ubiquity, personalization, flexibility, project, hands-on learning, data Interpretation, dynamic assessment, student participation and role blending). Education may also occur anywhere, anytime too if we offer the students tools and motivation enough.

Only tools rush in. Nevertheless, we must be careful with the development of such tools if their purpose is education and/or science. A SLR (BATTISTELLA; WANGENHEIM, 2016) reveals that from 107 educational games they analyzed, only 6 provided information about the theoretical foundation of their process of creation. Another SLR considering 112 studies concludes that "most evaluations of educational games are performed in an ad-hoc manner in terms of research design, measurement and data collection, and analysis. Most evaluations of educational games lack scientific rigor" (PETRI; WANGENHEIM, 2017).

Computing, not just programming. The current state of computing education shows computing being introduced mostly through imperative programming, with basic education relying more on VPL and higher on TPL. It is evident that programming and machines are essential for CSEd and have been the center of it for a long time now (DENNING; TEDRE, 2021). However, it is a recurring concern not to equate CS and CT to programming or coding (ARMONI, 2016). Yet, these terms have all been used interchangeably even inside CER. And it is hard to argue to the enthusiasts out of CS that those are different while keeping coding massively as the proper, main and sometimes only approach to introduce computing. If every time one sees computing being introduced they see coding in a programming course and nothing more, the synonymization is just a natural consequence. Offering alternatives nurtures the view that computing covers more, is more.

Programming, not just coding. This work has already defined CS and CT, leaving other two terms that are often equated left to be differed here: **coding** refers to the making of code, a collection of information represented (coded) in a certain way, which will be used here as synonymous to implementing/implementation; while **programming** refers to the making of programs, executable abstractions of processes, therefore it will be used here as synonymous to the full (software) development process needed

to make a program, which includes specifying, coding, testing and debugging. After these definitions, it is worth noting that most of the programming used for introducing computing is primarily coding-centred, testing and debugging are secondary and specifications are given or taken for granted. It does not represent what programming means in real-world scenarios, where it will be necessary to gather requirements, specify the solution, translate it into an executable implementation, evaluate if it is working properly and fix the errors if necessary.

Coding for all? The main argument that puts CT in such a strong enthusiasm for education is that there is a number of things in CS that are increasingly useful for all, not just IT and CS professionals. While it is reasonable to state the same, at a smaller scale, for coding, CT arguments are as ambitious as seeing it side-by-side to the 3 Rs (**R**eading, w**R**iting, a**R**ithmetics) (WING, 2006). I find it difficult to see coding being that useful for any citizen, CS might be, and computing certainly is, at least in a technological society. Furthermore, the farther technology goes, the less we need to worry about coding. Our programming languages are using higher and higher levels of abstractions, machine learning is progressing in auto-completing our solutions (SVYATKOVSKIY et al., 2019) and AI is on its way to code for us from natural language textual input (BECKER; GOTTSCHLICH, 2021; CRUZ-BENITO et al., 2021; OPENAI, 2023). In some years coding may not be that necessary, or be drastically changed. On the other hand, programming as the wider concept of full development of executable solutions, CS fundamental concepts, methods and strategies for problem-solving are much more likely to stick around for some time.

Coding for teachers. It was presented that programming (coding-centred) faces such big challenges with novices in the field, just imagine those out of the field then? It was also presented that a big issue in education is that there is always something new and exciting being presented and fighting to be included, which often implies teachers to be trained. Ideally, teachers would receive an infinite amount of training on everything. In the real world, we need to work within the limits of human capabilities, administrative viability, constraints of time and resources. Computing has enough proof of its impact and importance to require teachers' training, but specifically "coding" might be pushing it too far. Can CSEd afford to assume teaching coding in the very limited context of teacher training? If there are struggles in much more controlled environments full of soon-to-be specialists on full-time dedication (CS undergraduate students), what is the viability of bringing it to professionals from other fields that already have full-time jobs? Besides, would teachers even make good use of it, out of eventual lab classes? Is it worth the effort?

Coding as the core. Coding may be the functional hearth that gives life to CS solutions, but that may actually be an argument in favor of NOT using it as an introduction. When you are introduced to something, you are expected to meet the most

superficial layers first, not delve deep into the core straight ahead. Coding is not by any means a bad approach for computing education, it just might not be an ideal first step. That leads us to the question of what should come first then. Well, CT would be a good answer if it wasn't already meaning "coding with VPL" most of time. Fighting for the perfect CT definition would be pointless here because in theory, CT is about problem-solving and abstraction, it is in practice that little is said about problems and much is said about coding. Thus, CT becomes a good answer for what should come first if, and only if, CSEd manages to make it truly about the problem-solving skills its conceptualizations claim it to be.

1.2.2 Computational Thinking as a mental process

The concept of CT. Despite CT appearing in many national curricula already, it keeps appearing in many different forms, without an agreement about a default definition (DENNING; TEDRE, 2019). The conceptualizations, however, generally agree that it is a "mental activity to solve problems, ranging from using computers, computation, fundamental concepts or methods of CS, to designing the solution in a way a computer could run it or at least be used to carry out some help" (SILVA JUNIOR, 2020). After an overview of how CT is conceptualized, the following definition was proposed as an attempt to convey its meaning as a whole: "The ability to critically, consciously and creatively use, create and/or reflect on the products, methods and/or impacts of CS to solve problems" (SILVA JUNIOR, 2020). When frameworks, models or simply deeper meanings are given for CT, authors present a set of problem-solving skills based on CS topics.

Variants of CT. Different influential definitions propose their own subset of skills and attribute their own weight/granularity to each concept. A comprehensive SLR (SILVA JUNIOR, 2020) grouped the most cited terms related to CT from 28 resources pointed as CT definition sources in multiple papers. The result of this SLR is visually summarized by a venn diagram (Figure 1, top), meaning, for instance, that "Abstraction" was found (at least once) in 26 of the 28 sources, while "Decomposition" was found in 16. The three diagrams below it (Figure 1, bottom) are examples of how each source defined CT individually, considering different terms, an additional visual resource to highlight the variability of the definitions. The SLR showed that abstraction, problem-solving and algorithmic thinking are the most mentioned terms related to CT, but there are a multitude of terms mentioned in the majority of the sources.

Critique to the variability. The lack of unified standards forces researchers to pick their favorite definition to work with, resulting in different sets of skills being considered for the same concept, CT. That raises some critical questions: are they all really referring to the same thing? Or are we simply calling different constructs by the same name? Can we measure the same construct (CT) through different skills? Is the CT

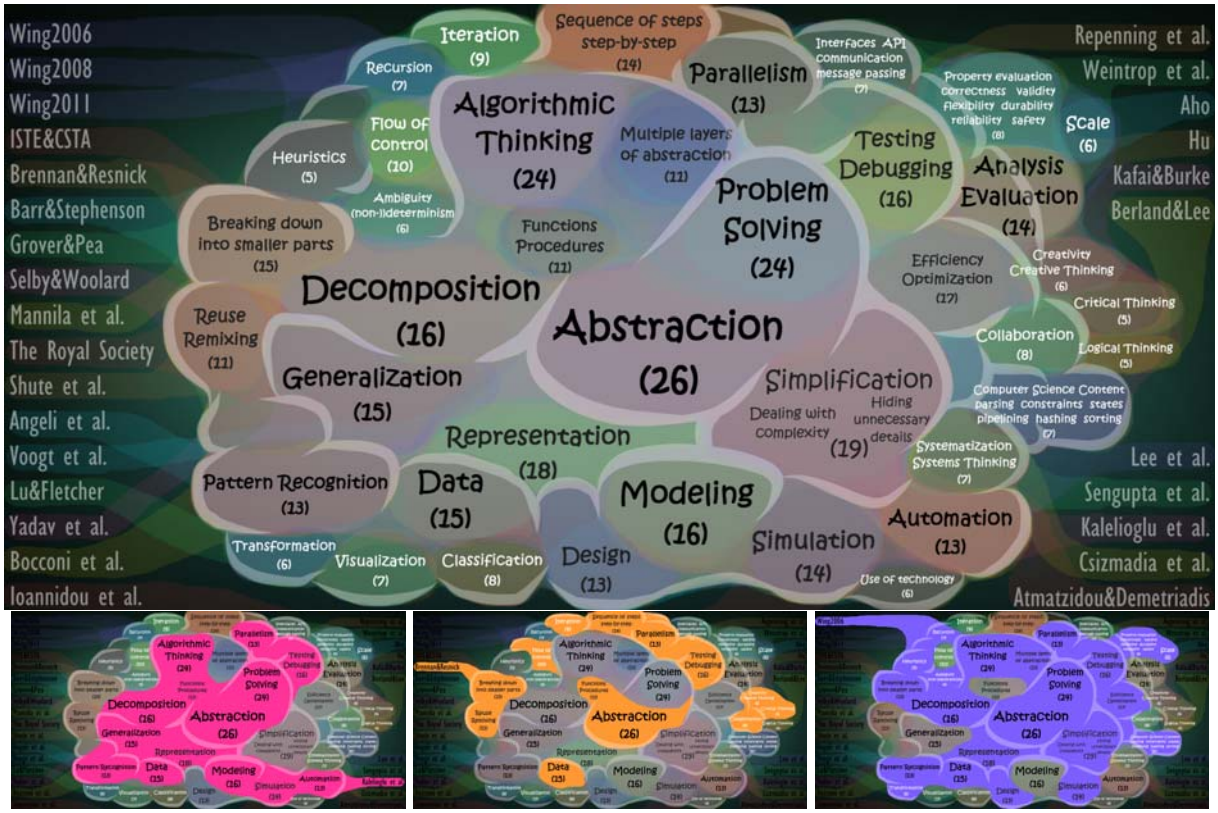


Figure 1 – Venn Diagram of CT-related terms. Top - All authors; Left - (KALELIOGLU; GULBAHAR; KUKUL, 2016); Middle - (BRENNAN; RESNICK, 2012); and Right - (WING, 2006). Source: (SILVA JUNIOR, 2020).

measured through A, B and C skills equivalent or at least comparable to the CT measured through X, Y and Z skills? Are they complementary? Can they be aggregated into a single measurement? Is there a minimum, core set of constituents that can be used to reliably estimate CT? If so, what is the minimum set? Does it vary by definition too? All those questions pose serious threats to the scientific investigation of CT, to treat it as a serious psychometric construct.

Lack of standardization. Possibly due to this chaotic situation with the definition, most interventions assessing CT use their own elaborated pre and post-tests as the main source of assessment, without greater concerns about their validity (LIU; LUO; ISRAEL, 2021; FAGERLUND et al., 2021). As the literature matured, meta-analysis scrutinized diverse aspects of empirical studies, consolidating evidence of the effectiveness of developing CT using: educational games (SUN; GUO; HU, 2023); collaborative problem solving (LAI; WONG, 2022); and unplugged activities (CHEN et al., 2023). Although they consider the use of different assessment strategies amongst the studies they included, we may need a careful review specifically dedicated to identifying whether those personalized tests are targeting the same constructs or not. If CT is to meet the CSEd plans for compulsory education, more research is necessary to properly and reliably assess it.

A process, not a factual knowledge. Although CT is reasonably accepted as

a mental activity in theory, when operationalized, it has been assessed mostly as a learning product, rather than a thought process (TANG et al., 2020). Closed-ended questionnaires (Item Response Theory) and artifact analysis are prevalent assessment instruments for CT, which confirms CT is regarded as a learning outcome (LIU; LUO; ISRAEL, 2021). We think that if we are to understand CT as the problem-solving process the theory tends to agree it is, then its assessment should be directed toward the process itself, not the final product/solution. Initiatives using eye-tracking (KE et al., 2021; PAPAVLASOPOULOU et al., 2017; ARSLANYILMAZ; SHARIF, 2018) and cognitive interviews (LUO et al., 2020; PAN et al., 2023) can be seen as promising steps into that direction, but they remain under-explored (LIU; LUO; ISRAEL, 2021).

1.2.3 The abstract concept of abstraction

A pillar of CT. Conceptually, learning CT should provide a foundational framework for understanding the fundamental principles that underlie CS and harness the power of abstraction as a key mental tool. As one of the core pillars of CT (WING, 2008), abstraction allows learners to create simplified representations of complex systems, enabling them to focus on essential details while suppressing unnecessary intricacies. This cognitive process facilitates the construction of models that capture the essence of a problem, facilitating effective problem-solving across a broad range of domains (MIROLO et al., 2022). Abstraction serves as a bridge between the real world and the digital realm, empowering students to reason about problems and devise solutions in ways that leverage the strengths of computational systems. Let us recall that abstraction was pointed as one of the main difficulties novice programmers have (MEDEIROS; RAMALHO; FALCÃO, 2018). This reinforces that CT (with its pillars) should precede programming education, serving, amongst other purposes, as a preparation for it.

History of abstraction. Early conceptualizations of abstraction can be traced back to ancient Greece with Plato (360 BCE) and Aristotle (384 BCE) through their concepts of forms (qualities) and sensibles (what is perceived through sensations). These were later revisited by several philosophers, such as Locke and Jean Piaget, highlighting the famous abstract-concrete and particular-general distinctions that are common today:

> “Locke proposed two types of ideas: particular and general. Particular ideas are constrained to specific contexts in space and time. General ideas are free from such restraints and thus can be applied to many different situations. In Locke’s view, abstraction is the process in which “ideas taken from particular beings become general representatives of all of the same kind” (Locke 1690/1979).” (SENGUPTA et al., 2013)

The first line of CT. A SLR analyzed the most commonly addressed terms related to CT (SILVA JUNIOR, 2020), producing a model organizing those terms into

six major lines: abstraction, algorithms, decomposition, data, automation and evaluation. Abstraction is defined in that model as “The process of extracting features from a source in a given context and putting them isolated in a new context” (SILVA JUNIOR, 2020) and is related to several concepts, as illustrated by Figure 2: generalization, pattern recognition, simplification (called shallow/deep abstraction, we won’t care to differ here), modeling/designing; and layers of abstraction.

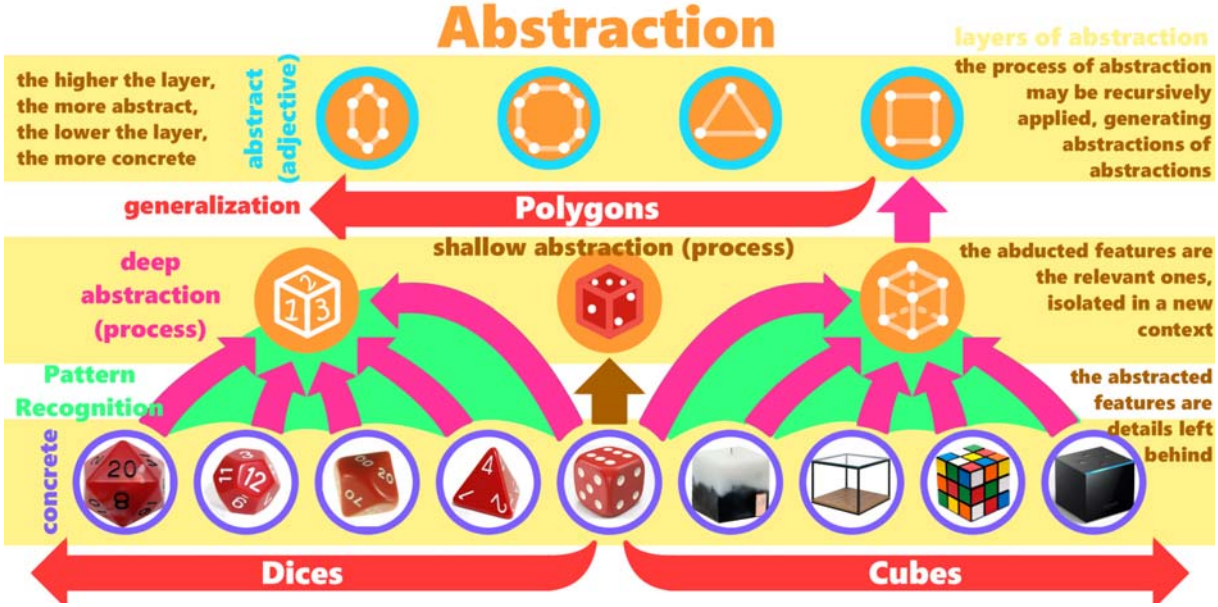


Figure 2 – The Abstraction Line from the Six Lines of CT model.
Source: (SILVA JUNIOR, 2020).

Disentangling abstraction. A SLR on abstraction (EZEAMUZIE; LEUNG; TING, 2022) groups and summarizes the different conceptual definitions of abstraction they found in 56 papers: a form of decomposition; a form of generalization; an intersection between both; problem formulation; data storage and manipulation; and program testing and verification. The vast majority of the studies were in the first three groups. When it comes to operationalization, they summarized approaches from 62 papers as: sophistication of programming concepts; matching patterns; alternative representations/modeling and simulation; transfer of problem solutions; measure of learner activity; identifying and reading program code.

Characterizations of abstraction. An overview of abstraction (MIROLO et al., 2022) summarizes basic definitions of abstraction as: extracting similarities; and ignoring non-essential features. Then discusses developments in the definitions of abstraction, from philosophy to mathematics and computer science. Throughout the discussion, they highlight: the generative power of abstraction (generalizations); abstraction itself as a process and as a product (constructs divorced from reality); structural and operational dichotomy, i.e. viewing something as an object or as a process, respectively; and the perspective of multiple representations. When it comes to abstraction in CS, they shed light on: programming languages and paradigms; generalisation and

parameterisation; procedural and data abstraction; information hiding and abstraction layers; and moving through different abstraction levels.

Layers of Abstraction. In CS, we have a particularly close relationship with abstraction, since we use it recursively, making abstractions of abstractions (WING, 2008). This allows us to manage an insane amount of transistors, giving meaning to their associations (logic gates), then to the association of those (logic circuits), and so on (basic instruction sets $\rightarrow$ machine code $\rightarrow$ assembly $\rightarrow$ high-level programming languages). **L**ayers **o**f **A**bstraction **(LoA)** are not just part of the history of computing and how we got to this point, but still very present in our modern world. The more we advance, the more complex and interconnected our systems become. The more we interact with them, the more important it becomes to realize we are going up and down abstraction. When you can't open a file on your phone: Is the file itself corrupted, or did the software that opens it stop? Maybe the OS got bugged and needs a reset? Maybe the screen froze because of a hardware problem? What if it is not local? There could be a connection problem or the server isn't on?

Recursive world. We are running systems upon systems, more and more. This perception of the multiple LoA around us and how to deal with them is an important responsibility of computing education that feels neglected. Beyond its importance to the common citizen in a technological society of increasing complexity, LoA should be given more attention to the new generations of ICT professionals that will be in charge of creating and maintaining the systems underlying all that, Developing and using **A**pplication **P**rogramming **I**nterfaces **(API)**, which basically create a higher LoA abstracting the internal implementations, has become an essential skill for developers. Just like going down to the implementation of a function in a library when a problem cannot be found or solved externally. The so-called "tech stack" required for job applications just gets taller and taller, but understanding how to navigate the stack is a veiled skill that cannot be ignored.

From context to problem. The disruptive context of a fast-paced field created the challenge of teaching about a whole new world the students were born in, while the teachers did not. Ironically, this new world allows us to craft powerful tools with the potential to support education requiring less from the teachers. These tools, however, must take into account their educational purposes and follow solid theories to succeed in bringing us a technology-supported student protagonism. Regardless of the tools, deciding what and how to teach computing became an educational necessity that quickly embraced the CT movement to gain traction. However, exciting ideas with overarching claims require substantial effort to provide proportional evidence. From theoretical construct consistency to the applicability on real-world scenarios, we started to treat CT with the rigor a reliable and valid psychometric construct deserves. Notwithstanding, it is challenging to answer intricate questions about the collective behavior of

cognitive skills when they are not even individually well established yet. Many of the CT constituent skills are as vague, complex and multi-faceted as CT itself. For instance, abstraction has been subject of study way before CT was popularized. Now, it is revisited under the premise of being one of the pillars of CT. Composed of and related to other skills and topics, abstraction is said to pervade the core of CT. Curiously, what is perhaps its most iconic manifestation in CS, layers of abstraction, rarely appears as an explicit subject or goal in CT experiments.

1.3 Thesis Aims

What can I learn here? This section is meant to objectively state research questions, hypotheses, goals and what this work proposes for pursuing them. They are shown in succinct numbered lists that are coherent between them, e.g. question 3 raised hypothesis 3, leading to goal 3, for which proposal 3 was made. This section ends with a recapitulation of the introduction followed by a graphical summary, briefly describing in advance all content of the thesis. More detailed rationale, arguments and methods for each proposal are given in their respective chapters.

1.3.1 Research Questions

Research **Q**uestions **(RQ)** are the overarching questions driving this work.

- **RQ1:** How can we introduce computing into basic education, viewing CT as a problem-solving process prior to programming, while considering the educational trends of the 21st Century?
- **RQ2:** Can we properly illustrate (considering the target public) and work with different LoA using this approach?
- **RQ3:** How can we accurately model and assess capabilities related to LoA following this approach?
- **RQ4:** Can we design interventions following this approach to present or engage the capabilities modeled?

1.3.2 Research Hypothesis

Research **H**ypothesis **(RH)** are assumptions relying on plausibility to be validated by this work, used as first directions for answering the RQ.

- **RH1:** The process of specification, which precedes programming (meaning implementation), can successfully precede it also in basic education for introducing computing.

- **RH2:** Multiple LoA are supported by at least one specification language, wherein such concept is palatable to teenagers and children.
- **RH3:** A specification language/process and LoA can both fit into a reliable psychometric model.
- **RH4:** An activity based on a specification language is able to introduce concepts of LoA, teaching the basics and creating a solid frame of reference to deepen the learning later.

1.3.3 Research Goals

General Goal: - Build the theoretical, instrumental and methodological foundations to approach Computational Thinking skills in basic education as solid psychometric constructs, contributing to the operationalization of computer science education through the view of a general-purpose computing, focusing problem-solving skills, rather than technical knowledge.

Specific **G**oals **(SG)** are the explicit targets of this work. Although the goals are listed with generic words (CT skills) to emphasize we had expansibility and generalization ability in mind during the development, the work reported here treats, specifically, how to introduce **layers of abstraction**, a fundamental concept of a pillar of CT.

- **SG1:** Offer an alternative approach based on a specification language to approach introductory concepts of CT preceding programming.
- **SG2:** Make available tools that support dealing with LoA within an engaging educational environment where young students could exercise CT skills beyond class time.
- **SG3:** Develop a psychometric model to assess a CT skill.
- **SG4:** Design an intervention where the alternative approach can make use of the educational environment to foster the CT skill and to be assessed through the psychometric model.

1.3.4 Proposed Approach

Proposed **S**olutions **(PS)** refer to how we pursued our goals in this work.

- **PS1:** The graming approach, playing and making games specified as graph grammars.
- **PS2:** Wrappers, an abstract hierarchical graph grammars feature in GrameStation, an educational game engine based on graph grammars.

- **PS3:** ECD[GS]: Layers of Abstraction, an evidence-centred design psychometric model for LoA.
- **PS4:** Abstraction Land[GS], an educational game made in GrameStation to introduce concepts of LoA to children.

1.3.5 Graphical Summary

Recap. Throughout this introduction we saw how computing and education intertwined themselves as they evolved, reaching a disruptive point where CSEd got into basic education and should reach every citizen. We followed the CT journey to demystify CSEd, distancing it from a technical knowledge and suffering the consequences of being protagonized by vague general-purpose problem-solving skills. Taking sides in this quest, we faced the challenge of pursuing one of those skills: abstraction. Understanding its importance, we went from its ancient origins to a contemporary perspective centering LoA. Albeit this one being a reasonable scope, first we had to consider building the tools and environments to support the new directions we envisioned for CT. Which were objectively stated through the research questions, hypothesis, goals and solutions of the thesis.

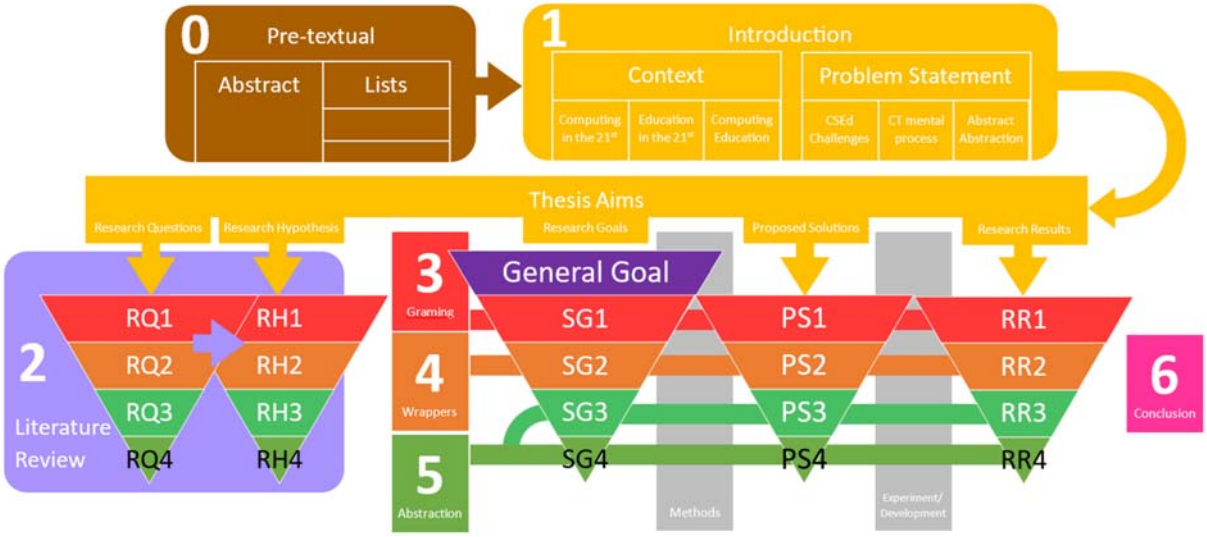


Figure 3 – A graphical summary of this work.
Source: Elaborated by the author.

Now I see. The Figure 3 visually summarizes this work and its content distribution, highlighting how the whole work was guided by the thesis aims and the iterative narrowing down of the scope. During chapter 2, it is shown how the existing literature led us from each research question to a respective hypothesis. During chapters 3, 4 and 5, it is shown how the goals motivated by the hypothesis were concretized by employing the described methods and produced research results (RR) after the developments/experiments.

2 LITERATURE REVIEW

Content Distribution. This chapter presents what we can learn from the current literature. The sections are after each of our research questions, discussing what the literature tells us regarding: how we can approach CT as problem-solving considering the educational trends in the first section; what we can use to approach LoA in the second section; and how we can assess this in the third section.

2.1 Computational Thinking

What can I learn here? This section reviews what the literature can teach us about the following research question, effectively leading us to the hypothesis that concludes the section:

RQ1: How can we introduce computing into basic education, viewing CT as a problem-solving process prior to programming, while considering the educational trends of the 21st Century?

Computing without coding. When the problem was stated, it was elaborated why not to use coding to introduce computing. Here, it will be presented and discussed how computing has been approached without coding. Many of those approaches are indeed used for introducing CSEd, but they tend to have a character rather eventual, being considered secondary, complementary or situational.

Unplugged activities. The first example of codeless CSEd are the unplugged activities, popularly known as **C**omputer **S**cience **U**nplugged **(CSU)**, the organization and movement to promote the pedagogical approach of allowing students to explore CS ideas before working with a computer (ARANDA; FERGUSON, 2018). CSU origins are credited to the Computer Science Education Research Group, at the University of Canterbury, New Zealand, often represented by a book of unplugged activities (BELL et al., 2009). CSU movement is aligned with the view that students shouldn't be "encumbered by the technical expertise required to code" to explore the fundamental ideas of CS (ARANDA; FERGUSON, 2018). It is also regularly compared to coding and instead of positioning itself against coding, it is rather seen as a "priming step to help

students understand algorithmic steps before they write code" (HUANG; LOOI, 2021). The critical discussion sounds really familiar, with the strengths of: sustaining potential to integrate non-computing subjects with CT (interdisciplinary approaches); and being less intimidating to teachers without a background in CS; and the following weaknesses: lack of construct validity, confounding factors and what constitutes evidence of CT with those of programming; and the claims for expanding understanding of CT in the theory does not reflect on practice, where CSU is used mostly to teach the very same concepts of programming (HUANG; LOOI, 2021).

Automation. These discussions are so aligned with all that was argued in here for our approach that one might think that we are talking the same talk, wearing the same shoes. If there is already an approach with such similarity in many aspects, what is it that makes graming worth introducing, what distinguishes it? The main, obvious difference is automation. The very definition of CSU is the lack of machine usage for CSEd. However, in addition to automation being often seen as one of the main aspects of CT (SILVA JUNIOR, 2020), it is about borrowing the power of CS, a power with high speed of growth. Using this power in favor of education, such as automating assessment and providing meaningful calculations on data gathering is such a big deal. Diving into the fast-paced social behaviors of the digital world to instantly communicate, personalize feedback, promote content creation/sharing among active, diverse communities and grant ubiquity for the learning process are contributions of automation beyond measure.

Robotics. Speaking of automation, another very common approach for CSEd is robotics (TEKDAL, 2021). In theory, robotics starts with assembling electronics and develops around how to make it perceive the environment and react to it. In practice, despite coupling hardware and software, most introductory robotics activities focus on programming or the creation of algorithms (LÓPEZ-BELMONTE et al., 2021; IOANNOU; MAKRIDOU, 2018), using pre-defined environments where all obstacles are known (and intentional, when existing). They then turn up to be really similar to regular coding activities (YANG; LIU; CHEN, 2020), including the predominance of Scratch (SANTOS; ARAUJO; BITTENCOURT, 2018). The main difference being that a physical agent runs your solution, making the output an interaction with the real world, instead of an avatar confined to its virtual world. It seems much more like a "coding+" approach than an alternative to it. Thus, there is not much to relate to robotics in the scope of this work.

Tangible user interfaces. Inverting this logic, the **T**angible **U**ser **I**nterfaces **(TUI)** gather inputs from a physical agent to interact with the digital world (XU, 2005). Bringing a hands-on alternative to **G**raphical **U**ser **I**nterfaces **(GUI)**, TUI are used the most to work with small children and in special/inclusive education to develop visuospatial and motor skills (O'MALLEY; FRASER, 2004). They aim to transform activities into

more expressive and exploratory experiences, being much more of a support technology to be applied for something else, than an approach on its own. Conversely, all approaches being discussed can be applied to something else, but students do not learn many topics on TUI itself, as they do for programming or robotics for instance. In this sense, the main difference between TUI and graming for introducing computing would be the depth of the approach. Although it can offer benefits, there is not that much involved in TUI to explore as an approach when isolated.

Block Programming. As the main approach being massively used to introduce computing, Scratch and similar languages have earned space in this discussion. But only to make it clear that although they are considered event-driven VPL (RESNICK et al., 2009), they ultimately follow the logics of a partially pre-written TPL under the imperative paradigm. Because, in the end, most activities are targeting solutions and directing students' thinking towards an ordered series of textual commands passing the instructions concerning how to do something. This means that besides being highlighted as an alternative to the traditional TPLs, it is about coding just as much, and arguably following the same imperative paradigm given the scope of most activities being restricted to a single character completing an ordered series of actions.

Other paradigms. Paradigms, by the way, have long been center of introductory programming discussion (BRILLIANT; WISEMAN, 1996). However, imperative and OOP had never dropped their status of main approaches in CSEd (KRISHNAMURTHI; FISLER, 2019), as all other approaches have been marginally approached in CSEd (SAMUEL, 2017). Most programming languages are multiparadigm, since programming paradigms are loosely defined, not mutually exclusive and much more about a focus than a restrictive, rigorous set of features (KRISHNAMURTHI; FISLER, 2019). This work cares to differ the following, which are all supported by graming: imperative languages are concerned with "how", passing instructions detailing the steps to reach a goal, standing at a lower level of abstraction; declarative languages are concerned with "what", informing just inputs and expecting the output to be reached through the language's rules, standing at a higher level of abstraction; procedural languages are concerned with describing and ordering procedures, sequencing instructions into groups to be called; OOP is concerned with an analogy to real world objects, describing classes with features and behaviors to interact with each other; and event-driven languages are concerned with events, meaning the program does not flow sequentially from the start to the finish, it waits for events to happen to appoint responses for them.

Model/design-first. Finally, specification has been proposed for introductory programming. The model-driven programming approaches, often relying on **U**nified **M**odeling **L**anguage **(UML)** notations and strong reliance on OOP, propose going from the problem domain to modeling, and only after, from modeling to code (BENNEDSEN; CASPERSEN, 2008). More specifically, after modeling, first using the UML model to

code the skeleton of the program using coding patterns; then specify properties and distribute responsibilities; implement the classes; and then their methods. However, this approach is pretty scarce in the literature, which makes it difficult to discuss its characteristics. But theoretically, the main difference is that in the graming approach we are using specification for effectively delivering executable solutions. In model-first programming, the specification is nothing but the first step in a process dominated by several coding processes. This as a first contact could unconsciously contribute to seeing the specification as a negligible part of programming, which is precisely what we are trying to prove wrong. We argue that, if we are to educate on specification, it should be the focus, it should clearly show power and results. Then, once the learner has matured the concepts and importance of specification, it should be presented to coding and the rest of the development flow.

Importance of programming. This work understands the greatest triumph of imperative programming as being its generalness of purpose and freedom to create exciting things due to their "autonomy" of running processes on their own and answering inputs. It almost feels like creating a living thing, a dynamic artefact. Let alone the (nowadays) instant feedback, immediately seeing and interacting with the creation. That is what CSU misses abandoning automation; what robotics struggles with when partially leaving the digital world (restricting the freedom to laws of physics); what block programming preserves while dealing with syntax annoyance; what other paradigms seem to turn harder or less intuitive; what specification in model-first approaches fail to achieve without coding; and what specifying in graming is expected to reproduce while turning the process much more problem-centred.

Hypothesis. Therefore, we were led to the following research hypothesis:

RH1: The process of specification, which precedes programming (meaning implementation), can successfully precede it also in basic education for introducing computing.

2.2 Abstraction

What can I learn here? This section reviews what the literature can teach us about the following research question, effectively leading us to the hypothesis that concludes the section:

RQ2: Can we properly illustrate (considering the target public) and work with different LoA using this approach?

LoA literature. The concept of LoA is often not mentioned as a knowledge or skill to foster considering the creation, modification or navigation of new/custom LoA. The literature on CT mentions the importance of LoA always considering some predefined

layers whereupon one can approach problem-solving. For instance, it is highlighted that working with LoA is essential for maintaining clarity in computational problem-solving (HOPPE; WERNEBURG, 2019). The authors endorse that each pair of layers requires well-defined relationships, often represented through abstraction functions or simulation relations. They recognize this is critical for ensuring students can manage complexity in learning environments. But they are referring to LoA in a very broad and generic way, like most authors. A book (SENTANCE et al., 2023) explores abstraction at multiple levels in computing, such as the method, class, and module levels. It is asserted that teaching multiple layers of abstraction is fundamental for preparing students for complex problem-solving. A framework emphasizes three LoA for developing CT (QIAN; CHOI, 2023): data abstraction (simplifying complex data sets into manageable formats); control abstraction (simplifying control structures like loops and conditionals); and problem abstraction (breaking down real-world problems into solvable computational steps). Beyond theory, the studies in the following paragraphs discuss how tasks are designed to foster the specific skill of navigating through multiple LoA.

While programming. The operationalization of LoA is observed when students are engaged in programming tasks that explicitly require them to think in terms of abstraction functions (HOPPE; WERNEBURG, 2019). Here, the students focus on maintaining relationships between high-level ideas (e.g., user requirements or algorithms) and their lower-level implementations (e.g., code or machine instructions). This two-layer operational structure fosters an understanding of both conceptual abstraction and its detailed execution in code. Students engage in modeling exercises where they start by defining high-level requirements, then break them down into smaller, implementable steps that correspond to different layers (design, implementation). For that, they have to learn how to manage dependencies between layers and how to maintain the overall integrity of the system by working through different abstraction functions.

On problem-solving. The three-layered abstraction framework of data, control and problem abstraction (QIAN; CHOI, 2023) was used in an activity where students were assigned tasks where they needed to move between these three layers to construct computational models or algorithms. In a problem-solving activity, it is expected a student first works on problem abstraction by identifying the main computational challenge, then develops a high-level algorithm (control abstraction) and finally manages the data involved (data abstraction). There, students iteratively refine their understanding of how data flows between these layers and how different control structures handle that data. Exercises often involve writing pseudo-code for high-level concepts and then implementing lower-level code to solve the problem. It is reported that students developed the ability to work across these interconnected layers, understanding how each layer contributes to the final solution without being overwhelmed by details.

On projects. The method, class and module LoA system (SENTANCE et al., 2023)

proposes that the operationalization happens when students build projects by progressively working through these layers: at the method level, students focus on writing individual functions; at the class level, students work on combining methods to form more cohesive units of functionality; and at the module level, students integrate multiple classes into larger systems or projects. The programming exercises are structured so that students must think in terms of these layers. For instance, they might start by writing simple functions (method-level abstraction), combine them into classes (class-level abstraction), and finally integrate them into a module or system (module-level abstraction). It is reported that students learn modular thinking and the importance of separating concerns at each layer while maintaining interoperability.

On system design. When it comes to specification languages, authors (STRIUK; SEMERIKOV, 2019) discuss the importance of abstraction in software engineering, noting how specification languages are instrumental in teaching students to move between different layers of system design, from high-level requirements to low-level implementation. A more concrete work used a Model-View-Controller (MVC) framework (DORODCHI et al., 2021) in an activity where students designed and visualized complex systems while distinguishing between the view layer (user interface), controller layer (logic handling), and model layer (data). This division directly introduces students to practical LoA in software development. It is reported that students develop the ability to visualize how each layer interacts with others without needing to delve into implementation specifics at every layer. An older work (BUNKER; GOPALAKRISHNAN; MCKEE, 2004) puts students to interact with several formal specification languages, such as Objective VHDL and UML, that inherently operate at various LoA. For instance, students use specification languages to design high-level system behaviors and verify low-level hardware functionality. This teaches them to manage how the abstract system design layer interacts with more concrete hardware implementation.

Hypothesis. Therefore, we were led to the following research hypothesis:

RH2: Multiple LoA are supported by at least one specification language, wherein such concept is palatable to teenagers and children.

2.3 Assessment

What can I learn here? This section reviews what the literature can teach us about the following research questions, effectively leading us to the hypotheses that conclude the section:

RQ3: How can we accurately model and assess capabilities related to LoA following this approach?

RQ4: Can we design interventions following this approach to present or engage the capabilities modeled?

Need for more research. A SLR (LIU; LUO; ISRAEL, 2021; TANG et al., 2020) makes explicit suggestions for future research to look for evidences of validity and reliability of CT assessment instruments.

Validity. The extent to which an instrument measures what it is intended to measure. That is, the accuracy and appropriateness of the instrument for the study's purpose. In our case, we are interested in finding out how much a CT assessment is estimating CT skills, and not something else. Validity should not be neglected because it determines the credibility of the findings. Without validity, the results and conclusions drawn from the assessment may be misleading.

Reliability. The consistency and stability over time. A reliable instrument will always give us the same (or similar) results under the same (or similar) conditions. Reliability distances the results from the chances of being random or due to noise. In our case, we are interested in finding out if a CT assessment would give the same (or similar) result for the same (or similar) student across multiple administrations (assuming the student's proficiency remains constant). Techniques to assess reliability include test-retest reliability, inter-rater reliability, and internal consistency.

Internal validity. The extent to which a cause-and-effect relationship between the independent and dependent variables can be trusted. That is, how well isolated from other factors it is, how likely the given explanation is to be the real/only one. In our case, we are interested in finding out if we can trust that the task performed by the student was enabled by a CT skill, not any other. For instance, a study (ARAUJO et al., 2019) on the internal validity of the Bebras tasks (DAGIENE; FUTSCHEK, 2008) for CT warns that the experts' mappings of which CT skills are assessed by each question mismatch the results given by factor analysis, presenting a threat to their validity. Techniques to enhance internal validity include randomization, control groups, and blinding.

External validity The extent to which the findings can be generalized to other settings. That is, how much of it holds true when applied more broadly. In our case, we are interested in finding out how much of a CT teaching method applied in one school can be generalized to other schools and educational environments. Techniques to improve external validity include using representative samples, replicating studies in different settings, and considering sociocultural differences.

Convergent validity. The degree to which two measures that theoretically should be related are actually related. Establishing the validity of a new measurement tool by comparing it with an established instrument measuring the same construct. In our case, as there are no well-established instruments on CT assessment yet, the best we can do is compare if our new tools seem to agree between them (ROMÁN-GONZÁLEZ; MORENO-LEÓN; ROBLES, 2019) or compare with other, related constructs. Addressing that, some efforts (ROMÁN-GONZÁLEZ, 2015; ROMÁN-GONZÁLEZ; PÉREZ-GONZÁLEZ; JIMÉNEZ-FERNÁNDEZ, 2017) took the direction of comparing their CT-

test to a golden standard of assessment on intelligence, the CHC Model (SCHNEIDER; MCGREW, 2022).

Psychometrics. The scientific field concerned with the theory and technique of psychological measurement. It involves the development, validation, and refinement of measurement instruments such as tests, questionnaires, and scales designed to assess cognitive abilities, personality traits, attitudes, and other psychological constructs. Central to psychometrics are concepts like reliability and validity. Psychometricians utilize statistical models, particularly those based on classical test theory and item response theory, to analyze data and improve the precision of psychological assessments. This discipline plays a vital role in various fields, including education, clinical psychology, and organizational behavior, by enabling the quantification of complex mental phenomena.

Psychometric frameworks. An example of a psychometric framework that can be used to guide the development of a robust assessment instrument is the **E**vidence-**C**entered **D**esign **(ECD)** (MISLEVY; ALMOND; LUKAS, 2003). It focuses on ensuring that the design aligns with the specific inferences and claims about learners' knowledge or abilities that the assessment aims to measure. By explicitly linking each task to the inferences being made about the learner, ECD strengthens the validity of the assessment instrument. It ensures that the tasks are meaningful representations of the underlying constructs and that the evidence collected from responses directly supports the intended claims. This structure reduces ambiguity and enhances the reliability of the conclusions drawn, contributing to a more valid and coherent assessment process. It is a very generic framework, allowing its application in various different areas, as previous work have proven (SHUTE; TORRES, 2012; ARIELI-ATTALI; CAYTON-HODGES, 2014; PHELPS et al., 2020).

Hypothesis. Therefore, we were led to the following research hypothesis:

RH3: A specification language/process and LoA can both fit into a reliable psychometric model.

RH4: An activity based on a specification language is able to introduce concepts of LoA, teaching the basics and creating a solid frame of reference to deepen the learning later.

3 GRAMING: SPECIFYING GAMES WITH GRAPHS

SG1: Offer an alternative approach based on a specification language to approach introductory concepts of CT preceding programming.

Content Distribution. In this chapter you will learn how we approached the research specific goal above, understanding how we reached the proposed solution that concludes the chapter. In this regard, the first section introduces what is and why we choose such approach. The second section reveals the methodological apparatus used. The third section showcases the results reached. And the fourth section concludes the chapter discussing implications and future work.

3.1 Introduction

What can I learn here? This section clarifies how games allow us to gather together several educational trends; how GG allow us to approach computing under a different perspective that differs it from programming; and offers the necessary notions of GG for those who don't know it.

3.1.1 Why Games?

Skill matters. Educational gaming has been shown to foster 21st-century skills, particularly in enhancing learning skills such as creativity, critical thinking, communication, collaboration, and problem-solving. For example, game-making activities promote creative problem-solving and encourage students to work together, thereby improving both collaboration and communication skills (BERMINGHAM et al., 2013). Games also provide environments that engage critical thinking and innovation by presenting players with complex, open-ended problems requiring thoughtful strategies and experimentation (NAVARRETE; MINNIGERODE, 2013).

Literate gamers. In addition, educational games contribute to the development of literacy skills, such as information literacy, media literacy, and ICT literacy, by encouraging students to evaluate and apply information critically in game-based tasks. Game-based environments can be effectively designed to engage students in media

analysis, information processing, and the use of digital tools, helping to foster ICT literacy (THORNHILL-MILLER et al., 2023). Games like Minecraft and other creation-based environments are particularly effective at cultivating ICT literacy through their design-based learning approaches (QIAN; CLARK, 2016). Finally, educational games promote life skills such as flexibility, adaptability, initiative, and self-direction. By working in diverse, dynamic gaming environments, learners are often required to lead teams, adapt to new challenges, and demonstrate accountability in team-based contexts (YANG, 2015).

Gaming 4.0. Educational gaming has emerged as a powerful tool in the context of Education 4.0, promoting key pillars like ubiquity, personalization, flexibility, and hands-on learning. In this new educational paradigm, ubiquity refers to the availability of learning anywhere and anytime, enabled by digital gaming platforms. Ubiquitous learning environments allow students to engage in game-based activities that promote continuous learning, accessible through mobile and online platforms, fostering engagement across various settings (IQBAL; MANGINA; CAMPBELL, 2022). Similarly, personalization is a core element in educational gaming, where learning experiences are tailored to individual needs and preferences, providing unique learning paths for each student. Educational games may offer personalized learning environments that adapt to student abilities, enabling more effective and individualized learning experiences (BONTCHEV; ANTONOVA; DANKOV, 2020). In terms of flexibility, educational games support adaptable learning processes by allowing students to explore various solutions at their own pace.

Power of interactivity. Games inherently encourage project-based and hands-on learning, crucial for Education 4.0, where students take an active role in constructing knowledge by completing in-game tasks and solving real-world problems. Games offer dynamic project-based environments where students practice data interpretation, analyzing in-game information to make strategic decisions, an essential skill in Education 4.0 (HUANG et al., 2022). This approach allows students to engage with content actively, leading to more meaningful learning outcomes. Moreover, dynamic assessment is integral to gaming within Education 4.0. Unlike traditional static assessments, educational games may provide real-time feedback and continuous evaluation of student performance, as demonstrated in a work on collaborative game-based environments for mathematics (AHMED, 2023). Student participation and role blending also play vital roles, as gaming environments often require learners to collaborate, shift roles, and take responsibility for collective outcomes. This bridges both ends of the teaching-learning process, fostering greater engagement and accountability. Educational gaming, therefore, serves as a multifaceted tool in the transformation of education towards the goals of Education 4.0.

Industry impact. Educational games are uniquely positioned to connect with to-

day's students due to the massive growth of the gaming industry, which has become an integral part of youth culture. With the global gaming market now exceeding $300 billion and engaging over 3 billion players worldwide, gaming has become one of the most pervasive entertainment mediums, especially among younger generations. This deep familiarity with gaming mechanics, interfaces, and interactive experiences makes educational games a natural fit for modern students, who are already comfortable navigating complex game environments. By leveraging game-based learning, educators can meet students where they are—incorporating elements of fun, challenge, and competition that resonate with their everyday digital experiences. Educational games not only tap into students' intrinsic motivation and engagement but also align with their preferences for interactivity, immediate feedback, and self-directed exploration. This synergy between students' recreational habits and educational methods helps bridge the gap between formal education and the digital world students inhabit, making learning more relevant and engaging.

Problem Based Gaming. Educational games offer a dynamic and interactive platform for PBL, effectively bridging theoretical concepts with practical applications. In PBL, students engage in complex, real-world problems that require critical thinking, problem-solving, and collaborative skills. Educational games enhance this process by immersing students in simulated environments where they must apply knowledge and strategies to overcome challenges. This immersive experience fosters deeper engagement and motivation, as students see the immediate consequences of their decisions and actions. Research indicates that educational games promote active learning by providing iterative feedback and opportunities for experimentation (ZENG; PARKS; SHANG, 2020), which are crucial components of PBL. By integrating educational games into PBL frameworks, educators can create rich, interactive learning experiences that not only reinforce academic content but also develop essential problem-solving skills and collaborative abilities.

Creative Game Makers. In PjBL, students work on extended projects that require them to solve real-world problems and produce tangible outcomes. Educational games offer unique opportunities for students to explore and manage complex scenarios within virtual environments. Examples of games that succeeded in this regard and are used in educational settings for PjBL are: Minecraft, whose open-world sandbox allows students to design and build projects collaboratively (BAR-EL; E. RINGLAND, 2020); Roblox, whose user-generated content and game development features enable students to create and iterate on their own projects (HAN; LIU; GAO, 2023); and the use of "mods" upon various games, which also extends the learning experience, allowing students to customize and adapt existing projects in response to evolving needs and constraints, or repurpose the originals (MCARTHUR; TEATHER, 2015).

Collaborative DIY. In the context of educational gaming, maker culture encourages

learners to engage in the creation of their own games, mods, and educational tools. This participatory approach allows students to design, build, refine and share gaming experiences. For example, platforms like Scratch or Unity enable students to develop their own games, integrating educational content into their designs and experiencing firsthand the iterative process of game development. The so-called Game jams further enhance educational gaming by providing structured, time-limited challenges that simulate real-world development scenarios. These events, where participants create games within tight deadlines, promote rapid prototyping, creative thinking, and teamwork. Studies on game jam participation have highlighted that such events improve learners' coding skills, project management abilities, adaptability, proficiency in game design principles and collaborative problem-solving (KOLEK; MOCHOCKI; GEMROT, 2022).

Flow State. The Flow theory (CSIKSZENTMIHALYI, 1991) describes a state of deep immersion and optimal performance where individuals experience effortless involvement in an activity, driven by a balance between challenge and skill. In education, flow theory guides the design of learning experiences that maximize student engagement and motivation. When tasks are appropriately challenging and aligned with students' skill levels, they are more likely to enter a state of flow, leading to heightened focus, intrinsic motivation, and improved performance. This state not only enhances learning by providing clear goals and immediate feedback but also fosters personal growth and satisfaction, contributing to greater academic achievement and a more rewarding educational experience. The flow state has been demonstrated to be massively approached using educational games (PERTTULA et al., 2017).

3.1.2 Why Graph Grammars?

Origins. Graph **G**rammars **(GG)** can be found among formal methods, i.e. "Formal notations, tools and techniques with a mathematical basis, often drawing on theoretical computer science fundamentals such as logic calculi, formal languages and automata theory, that are used to unambiguously specify the requirements of a system. Those notations and tools shall support the creation of formal specifications, the proof of properties for them and proofs of correctness of an eventual implementation with respect to the specification. It provides frameworks to specify, develop and verify systems in a systematic rather than ad hoc manner." (SILVA JUNIOR, 2020). Bowen; Stavridou (1993) defines correctness as the delivery of proper service, i.e. service which adheres to specified requirements. He highlights that a correct system is not necessarily a dependable system, the latter would be a system with properties such as safety, availability and reliability. Roughly speaking, correctness is about doing what it was designed to do. It sheds light on usually neglected and problematic discrepancies: what one meant and what he/she said, what one wanted to do and what was done,

or how a system is intended to behave and how its design makes it behave. The less difference between those, the greater the correctness.

Formal Specification. Hall (1990) says that formal methods are all about specification, it is predominantly a process of: writing a formal specification; proving properties about the specification; constructing a program by mathematically manipulating the specification; and verifying a program by mathematical argument. He refers to formal methods as "the use of mathematics in software development" (HALL, 1990). Terms are discussed by Hinchey; Bowen; Rouff (2006), that highlights the term "formal methods" is misleading, because despite originating from formal logic, they do not adequately incorporate many of the methodological aspects of traditional development methods. It is now used in computing to refer to a plethora of mathematically based activities. There is also a definition for formal specification: "A specification written in a formal notation, often for use in proof of correctness" (HINCHEY; BOWEN; ROUFF, 2006). Kossak et al. (2014) states that formal methods have several success stories in high-assurance systems. Wing (1990) lists examples of formal methods applications in system design, verification, validation, documentation, analysis and evaluation, in addition to requirement analysis. Bowen; Hinchey (1995) argues that formal methods may feel like an "overkill" (and indeed be one in some cases), but its use is recommended in any system where correctness is of concern, and even required when dealing with safety-critical and security-critical systems.

The CT in specification. Specifying something with a mathematical basis is all about defining, restricting, delimiting objective straight-forward meanings. It is, intrinsically and heavily, a process of abstraction, abducting relevant pieces of the uncontrolled complexity of the real world to an absolutely controlled environment, where we have knowledge about everything that might exist (the universe of the model) or happen (all events are ruled by predefined structures). Systems are almost always compound structures made of several different components, to specify them, good decomposition and the related skills are required. The behavioral, event side of the model must consider algorithm skills, since the lack of ambiguity is a pillar here. Data skills are all around the formalization process, after all, it is about putting into specific data structures that support mathematical techniques, enabling systematic visualization and then analyzing it. Once specified, the system should have described all its features and behaviors, even if not implemented or simulated, it is an automation challenge to describe in such a self-contained manner. At last, probably the most heavily used skills in formal methods are the evaluation ones, because our main targets are correctness of specifications and property checks.

Problem-solving. The process where an abstract solution to a problem is modeled and evaluated against key properties can notably be said aligned with CT conceptualizations. In comparison to the current dominant approach, specification is much more

problem-focused, much closer to the theoretical foundations of CS. While coding is much closer the the applications of CS. It is specification that actually confronts the problem, it is where the problem is conceived, modeled and abstractly solved. Coding comes after, it receives the problem already modeled from the specification and then turns it into a concrete solution for the real world. The specification stage has been neglected in CS education, students try to understand and model the problem on the fly while already programming. As a result, the CT movement rises and shouts for problem-solving skills to fill this gap. The GrameStation is an effort to revive the specification stage and automatically bring specifications to the real world (final product execution).

Well-defined nightmare. Unfortunately, despite their benefits and success, formal methods have been underused in real industrial and commercial solutions (KOSSAK et al., 2014; BOWEN; STAVRIDOU, 1993). There are at least 14 misconceptions revolving around the formal methods that might be the blamed (HALL, 1990; BOWEN; HINCHEY, 1995), but the main stigma carried by them is their spooky heavy math appearance. Finney (1996) points out that pioneers of programming used to have a solid mathematical education and used to see programming essentially as an application of math, which is not the case for current software engineers. With the advance of programming languages, we left the context where programmers were essentially mathematicians. Consequently, the use of formal methods, which are grounded in math, starts to have a hard time.

The best of both worlds. What if we could give our ugly scarecrow a nice-looking outfit? Could we get totally or at least partially rid of the spooky Greek letters and huge theorems while still using formal methods? We should try to leave the heavy notations for theoretical logicians and mathematicians but translate their models and findings into something more friendly to use them without being mathematicians ourselves. What if there was a formal method that could look as cute and innocent as comics? If the issue relies on textual annoyance, such as logic notation, advanced math operators, unusual symbols and Greek letters, our best bet is on visual alternatives, like graphs.

Graph Grammars. Generative grammars (CHOMSKY, 2013) and Petri Nets (PETRI; REISIG, 2008) are two common formal language topics often taught in CS courses for their relevance for compilers and overall TCS. If you replace the strings in a generative grammar with graphs, what you get is the basic notion of a **G**raph **G**rammar **(GG)** (EHRIG et al., 1997). Or yet, a Petri Net with dynamic changes over the system topology and references between tokens (RIBEIRO, 2000). GGs do have a robust underlying mathematical theory supporting them, but they have a quite unique power to hide their own complexity behind their visuals. Get some fancy looks on the vertices, add a little color on the edges (graph theory pun intended) and *voilá*: you have a formalism that no child will turn up the nose.

3.1.3 Graph Grammar Intuitions

Graph Grammar. A GG describes a system modeling a state of it as a graph (vertices and edges), and events that may alter the current state as a set of graph transformation **rules**. A GG must define how its **state graph** starts, in what is called the **initial graph**. Additionally, a GG may distinguish and restrict its elements by declaring them in a **type graph**. For instance, Figure 4 models the Pacman game as a GG. There is a type graph T declaring the existence of pacmans, ghosts, places (black dots), fruits, counters (white dot) and their relations in this GG. Then, there is an initial graph Ini that shows a pacman, two ghosts and a fruit in a 3x3 grid of places, while a counter at the bottom shows that a fruit has been eaten. Then, there are four rules, $moveP$, $moveG$, eat and $kill$, each being represented by a pair of graphs linked by an arrow carrying the name of the rule.

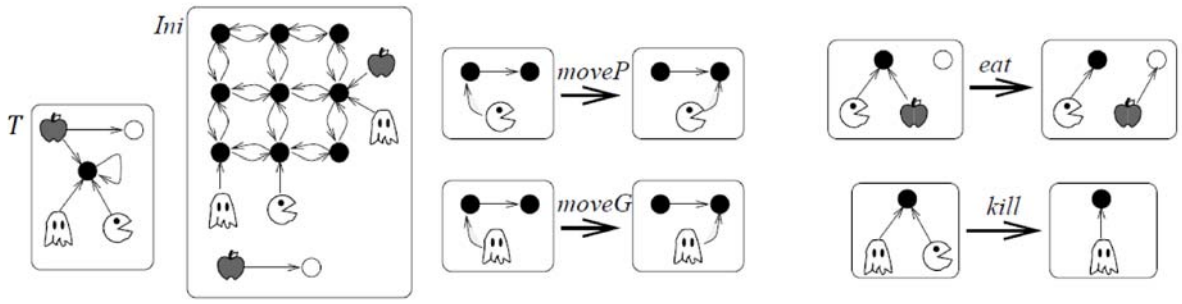


Figure 4 – A Pacman graph grammar.
Source: (RIBEIRO, 2000).

Rules. The pair of graphs representing rules are the **Left-Hand Side (LHS)**, expressing a condition for applying the rule, and the **Right-Hand Side (RHS)**, expressing a consequence of applying the rule. As in $moveP$ (Figure 4), the LHS defines the condition of having a pacman in a place that has a way to another, while the RHS defines the consequence of removing the pacman from the initial place and putting it into the other. This representation also implies element-wise mappings between graphs (**morphisms**). That is, for each element in one graph, we have to say which element (if one) in the other graph corresponds to it. For instance, saying if the pacman in the LHS is the same pacman in the RHS. If an element is successfully mapped (has a correspondent), it means the rule **preserves** it, as the pacman of $moveP$. If an element is left unmapped (has no correspondent) and is in the LHS, then the rule **deletes** it, as the pacman of $kill$. If unmapped and in the RHS, the rule **creates** it, as the edge from the fruit to the counter of eat.

Matches. This pacman is played by applying rules, which is a process that depends on another morphism: the **match**. The match is a total morphism from the LHS to the state graph, meaning that we must find a correspondence for each element. In Ini we could apply $moveP$ mapping any of the three adjacent places and $moveG$ mapping any

of the ghosts and their adjacent places. But not eat and $kill$, because there is not a pacman and a fruit/ghost in the same place, so we could not complete a match, since they must respect source and target of the edges.

Types. The base theory for GG makes no distinction between vertices and edges, i.e. it is not sufficient to specify that a vertex is a pacman and another is a ghost, they are all simply vertices. Distinguishing between elements is possible through labeling or typing, which is mapping every element into a label or a type. For typing, an additional component enters the GG: the **type graph**, which is a special graph where each element is considered distinct from the other. Then, every other graph of the GG must map its elements into the type graph (typing morphism) and respect the source/target restrictions their types impose. For instance, an edge from a black dot can point to another black dot in Ini (Figure 4), but not to a white dot, because in T there is an edge with the black dot type as its source and target (a loop), but no edge with the black dot as source and white dot as target. Again, in the visual representations, the morphisms are implied by the look of the elements, corresponding looks in different graphs imply they are mapped.

Definitions. We avoided exposing the underlying math in this paper, but the full set of formal definitions that we are following is shown in previous work (SILVA JUNIOR, 2020) and comes from an algebraic approach for GG (CORRADINI et al., 1997), using Double Pushout for graph transformations, injective matches and the category of typed attributed graphs and total morphisms (CAVALHEIRO; FOSS; RIBEIRO, 2017).

3.2 Methods

What can I learn here? This section presents the methodologies and tools we used in order to propose our approach and build the platform that powers it. That is, how we merged GBL and GG, as well as how and why we built a GG game engine from scratch.

3.2.1 Graph Game Engine

Graming. For all the reasons discussed in the introduction of this chapter, we had merged gaming and GG in previous work, turning GG into educational games, from physical boardgames (SILVA JUNIOR; CAVALHEIRO; FOSS, 2017) to digital games (SILVA JUNIOR; CAVALHEIRO; FOSS, 2019). These previous experiences matured into the notion of graph games (grames). Then, the quest for an adaptive grame that modified its own rules to adapt itself to the student's CT skills got us wondering if it was worth the extra cost of making a GG game engine instead. The thought that invaded our minds was: "Since we are already pursuing the flexibility of the underlying GG, we might as well allow any user to create any GG". Quite a leap, indeed. But

the endless possibilities it seemed to bring outweighed the perceived additional effort at the time.

Requirements gathering. So, our next step was to ask ourselves: “What a GG game engine would need in order to be useful in basic computing education?”. First and foremost, manipulating GG, that is, creating, editing and running GG, like other GG tools already did. Second, being able to create games, that is, interactive systems with their own mechanics, rules and goals. Third, it should be educational, guiding new users, prioritizing being user-friendly and supporting educational content. Fourth, it should be all of the above at the same time, those features should be integrated. So, our starting point was to review the existing GG tools and verify if they properly met these requirements.

GG Tools. The GrameStation is a game engine that allows the user to specify and run GG, but it does not offer tools for formal specification and analysis as other tools does (AZZI et al., 2018; TAENTZER, 2003; RENSINK, 2003). The core distinction is that GrameStation is an educational tool targeting non-expert users (in particular, children), rather than a professional formal specification tool. Additionally, GrameStation approaches GG centring decision-making through a dynamic environment, where the users respond on the fly to the state transitions caused by others. While the existing GG tools approach GG centring formal specification through a single user environment, where users analyze properties that emerge mostly from the automatic execution of its rules and the exploration of the generated state space. Also, it is specifically designed to create games out of GGs, bringing some adaptations and tools aiming at that goal.

Grame. In order to meet the requirements of manipulating GGs and creating games, at the same time, we introduce the concept of **Grames**, short for graph games. A grame is essentially a GG that contains special elements (**gears**) hidden from players (but not from creators) to control gaming aspects, such as which player is able to play at a given time (managing turns) and which rules each are available for each player. During the execution of grames, the engine offers players a subset of rules (controlled by gears), so the player can select one of them to start mapping a match to apply the rule. Gears can also be created or deleted by rules, which means this control is in the grame creator’s hands. A grame is played by selecting individual rules from the rule set and matching their LHS into the board (state graph).

Decision Trees. In order to meet the requirement of guiding new users, we designed a system based on decision trees to take all important modeling steps, that is, to create, edit or delete any element or component. This way, we kept the specification process relatively loyal to the formalities, since the user had to objectively define each single feature of everything they were modeling. While limiting their error proneness, since the set and order of features to be defined was given by the engine, and it also restricted the user answers to those that are correct with respect to the GG theory.

For instance, when a user would create a new edge, the engine would offer them a decision tree to pick its type. With the type set, the tree would calculate and offer only vertices of the same type as the source and target of the edge's type to fulfill these roles, respectively. That is, once an edge type isAt from an Animal to a Land is defined, the user can only create edges between Animals and Lands when the type isAt is selected.

3.2.2 Implementation

Unity. Unity (TECHNOLOGIES, 2020) is an excellent environment for developing an educational game engine due to its versatility, user-friendly interface, and extensive support resources. As a highly flexible and widely used game development platform, Unity offers a robust set of tools and features that facilitate the creation of interactive and engaging educational content. Its visual editor simplifies the design and development process, enabling educators and developers to prototype and refine their ideas efficiently. Additionally, Unity's extensive asset store and large community provide access to a wealth of pre-made assets, scripts, and plugins, which can significantly accelerate development. The platform's support for cross-platform deployment ensures that the educational game engine could reach a broad audience across various devices, including PC, mobile, and VR/AR systems. Therefore, we chose to use Unity due to its comprehensive ecosystem and supportive community.

Project design. We first structured the engine as shown in Figure 5: a builder module, where GG could be specified creating grame files, it also could import external files like images and encapsulate them into files of our own formats to share resources to be used in grames; an explorer module, where users could navigate through existing grames and packs to choose which they would play or edit; and a player module, where grames would be run.

Command center and parser. The engine works upon a command center responsible for delivering all relevant actions, which are mostly requested through a parser entity that reads command lines. The command lines are structured series of commands, IDs and variables, such as "[NEW] [TYPE] [VERTEX] [<Typer ID>] [<Alias>] [] [] [] [<Coord X>] [<Coord Y>] [<Coord Z>] [<Anchor ID>] [<Package ID>] [<Pocket ID>] [<Look ID>] [<Color Hex>] [<Size X>] [<Size Y>] [<Rotation X>] [<Rotation Y>] [<Rotation Z>]". These commands however, aren't typed, they are generated by the decision trees.

Tracker, logs and reports. Everything that happens in GrameStation is registered by a tracker module in a structured "Update Report" containing some classifications to identify its nature, a timestamp, the user registering it, and codes that identify exactly what the report is about. In order to occupy less space on the drive, the reports themselves only contain codes and variables, which are interpreted during execution

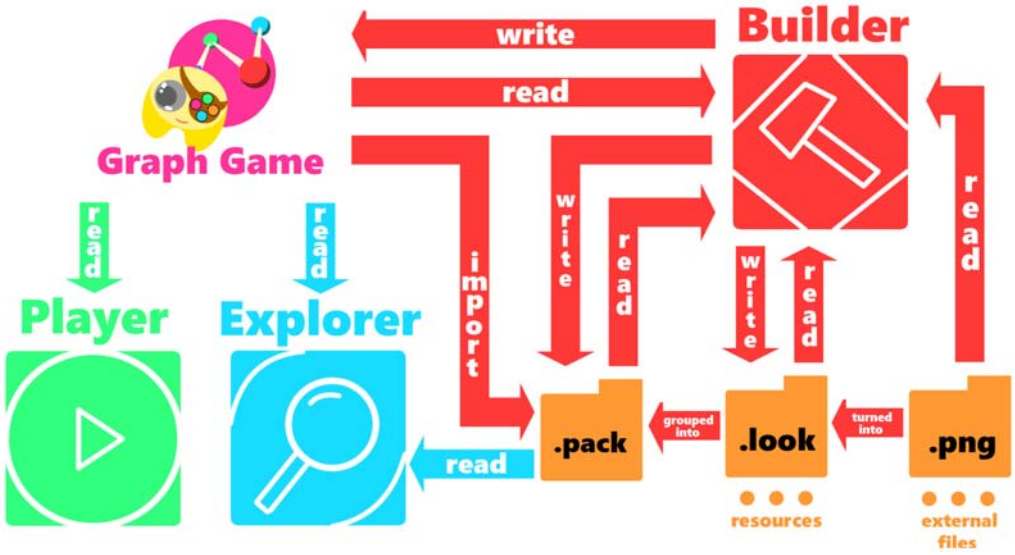


Figure 5 – GrameStation modules and overall organization.

.

to create a readable message. Some of them use a set of dynamically created details that couldn't be foreseen and coded, which include variables and custom messages.

File management. All files managed by GrameStation are stored in specific **Extensible Markup Language (XML)** files defined by **XML Schema Definition (XSD)** files we specified. They have been extended and modified every version, which is what we expected and the reason we refused to use XMLs obeying pre-existing XSDs.

3.2.3 User Experience

Grame Play. Preliminary published results (SILVA et al., 2022). A total of 17 people with different levels of knowledge about GG were invited to play two grames (The Last Tree - single player and Pacman) and complete a questionnaire about their experience. The activity was remote, asynchronous, and individually completed within 5 days. They were divided into three groups: group 1, composed of those who had previous experiences with GG; group 2 of those who didn't, but receive a video about GG's basic notions; and group 3, of those left clueless. The questionnaire was an adaptation of the MEEGA+ model (PETRI; WANGENHEIM; BORGATTO, 2016). The study concludes that participants did not report difficulty in using GrameStation, but in understanding how the grames should be played, i.e. they know how to use the platform to do things, they just don't know which things they are supposed to do. This indicates that GrameStation will have to do more than being easy to use, it will have to explain the behavior of GGs somehow.

Grame Design. Preliminary unpublished results (submitted). The same settings of the previous experiment were repeated, but now they had two weeks to complete and the questionnaires used were the System Usability Scale (GRIER et al., 2013) to

measure usability, and the AttrakDiff (HASSENZAHL; BURMESTER; KOLLER, 2003) questionnaire to check the user experience. Regarding usability, on a scale between 0-100, it had an average of 75.4 according to group 1; 41.5 according to group 2; and 43.5 according to group 3. Group 1 found GrameStation easy to use and agreed that most people would learn to use it quickly. While Groups 2 and 3 found GrameStation unnecessarily complex, that they would need help to be able to use the platform and they would need to learn many things before starting. Regarding UX, the results can be seen in Figure 6.

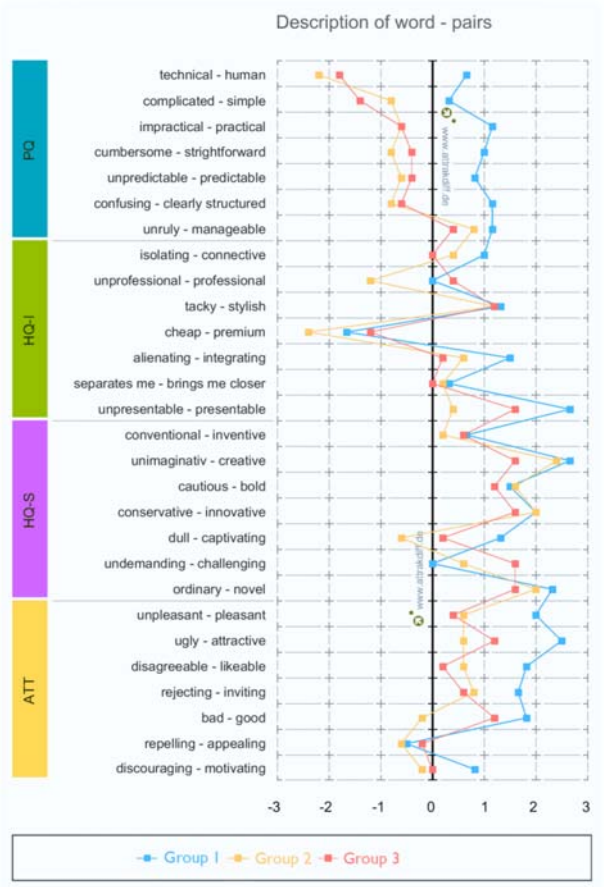


Figure 6 – GrameStation's User Experience chart of AttrakDiff

.

Pedagogical Agents. These results led us to look for alternatives to enhance user experience and usability. Which we found on pedagogical agents, virtual characters or avatars designed to facilitate and enhance the learning experience in educational tools. These agents can embody various roles, such as tutors, mentors, or facilitators, and interact with learners to provide guidance, feedback, and support. Their relevance in educational tools stems from their ability to create personalized, engaging, and interactive learning environments. By simulating human-like interactions, pedagogical agents can adapt to individual learning styles, offer tailored explanations, and maintain learner motivation. They also help bridge the gap between learners and educational content

by making complex concepts more accessible through dynamic and relatable presentations. As a result, pedagogical agents can significantly enhance the effectiveness of educational tools by fostering a more immersive and responsive learning experience.

Figure 7 – Early conceptualizations of gramers.

Gramers. Our take on pedagogical agents for GrameStation was to delve into an immersive universe for the platform, where the agents could be found. We designed the background fantasy of the Graph Universe, a fictional universe crowded with endless parallel realms with their own rules, the grames. In this universe exists small creatures called Gramers (Figure 7), who could connect themselves to Grames, entering the realm in full immersion as someone “entering the matrix”. Gramers were conceived as living technological gadgets, such as game consoles, VR headsets and 3-D printing machines. Our intention was to create a different Gramer for each purpose inside the platform, but keep them all under the same artistical identity.

3.3 Results

What can I learn here? This section shows the products of the operationalization of graming. That is, what was reached through the methods presented in the previous section. Here the GrameStation platform is showcased along with the pedagogical agents.

Guiding game modeling. We designed GrameStation with the goal of inducing an easy game design workflow that would carry the user through modeling stages guided by critical questions. The stages aren’t an explicit, static, rigid temporal series that once finished you can’t go back. They are simply an implicitly suggested workflow for an initial development and then further iterations, as users will be switching between them all the time.

What will it look like? In addition to the GG, grames count with external files such as images and audio effects to be used as **resources**, e.g. for the appearance of the vertices and edge. GrameStation organizes resources in collections (**packs**), and currently supports images using **Looks**, which are XMLs encapsulating a png or

a jpg file. If we want to make a grame, the very first thing we shall do is to import pre-available packs or to create a new one. For instance, Figure 8 shows a pack of looks for making a pacman grame.

Figure 8 – Looks for the pacman grame.

.

What will it be about? The next step in designing a grame is to define the types that we are going to use. Thus, the type graph fits the role of a declaration area. All grames begin with an empty type graph and an empty initial graph. GrameStation allows the users to create new types by requesting them a kind (vertex or edge), a name, a look and a color (source and target are also requested if its an edge). As shown in Figure 9, for a pacman grame we filled the type graph declaring: a Pacman, a Ghost, a Fruit and a Place. Additionally, we added two vertices to represent Victory and Loss. As a grame is a GG, relations stand out, so we also declared that: Pacman isAt a Place; Ghost isHaunting a Place; Place may have a wayTo another Place; and a Place hasA a Fruit.

Figure 9 – Type graph for the pacman grame.

.

How does it all begin? With everything declared, we proceed to fill the initial graph. GrameStation allows the users to create new elements in the initial graph by instantiating the ones defined in the type graph, requesting only a type and a name (source and target are also requested if its an edge). Continuing our example, Figure 10 shows the initial graph of the pacman grame. There are our hero pacman, our enemy ghost and

our goal fruit at (isAt, isHaunting, hasA, respectively) different linked (P01, P02, P12, P23) places (P0, P1, P2, P3).

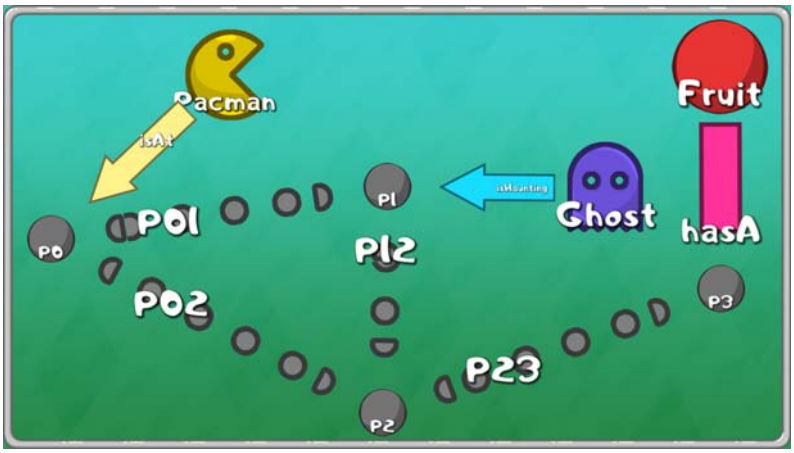


Figure 10 – Initial graph for the pacman grame.

.

What happens in there? At last, we specify the rules. GrameStation allows the creator to add as many rules as they want. And to add an element to a rule, it will request if it is deleted, preserved or deleted by the rule, as well as type and name. In our example (shown in Figure 11), the same $moveP$ from the pacman GG we introduced in section 2 (Figure 4) is modeled in the GrameStation. This rule preserves a pacman, two places (CurPos,FutPos) and the way between them (wayTo), while replacing its positioning edge that target a place (delete wasAt) to one that targets the other (create isGoingTo), effectively moving the pacman. All the other rules can be defined just like this one.

Figure 11 – Rule $moveP$ for the pacman grame.

.

How do I play? The specified rules can be selected on the top of the screen in the Grame Player during the execution. If so, their LHS and RHS will be shown and a match can be set by clicking the LHS elements and their corresponding elements on the board (state graph) in sequence. If any illegal mapping occurs, such as mapping a pacman into a ghost, a cross will appear signalizing the error and the current match will be

cancelled. Grames are designed to support parallel, asynchronous rule applications, result of multiple players playing at the same time. But, as GrameStation does not offer support for online play yet, an adaptation to allow multiplayer grames is temporarily held as default: once a rule is applied by a player in a computer, this computer will become the next player (they switch turns). This is all controlled by gears and could be changed by expert users. Gears bring a whole new level of freedom to model games, being possible to make real-time grames, rules that pass the turn backwards or changes the player order and much more. But for beginners, they should be completely ignored, as we ignored in our pacman grame. By default, all is set up to be a simple turn-based that passes the turn when any rule is applied.

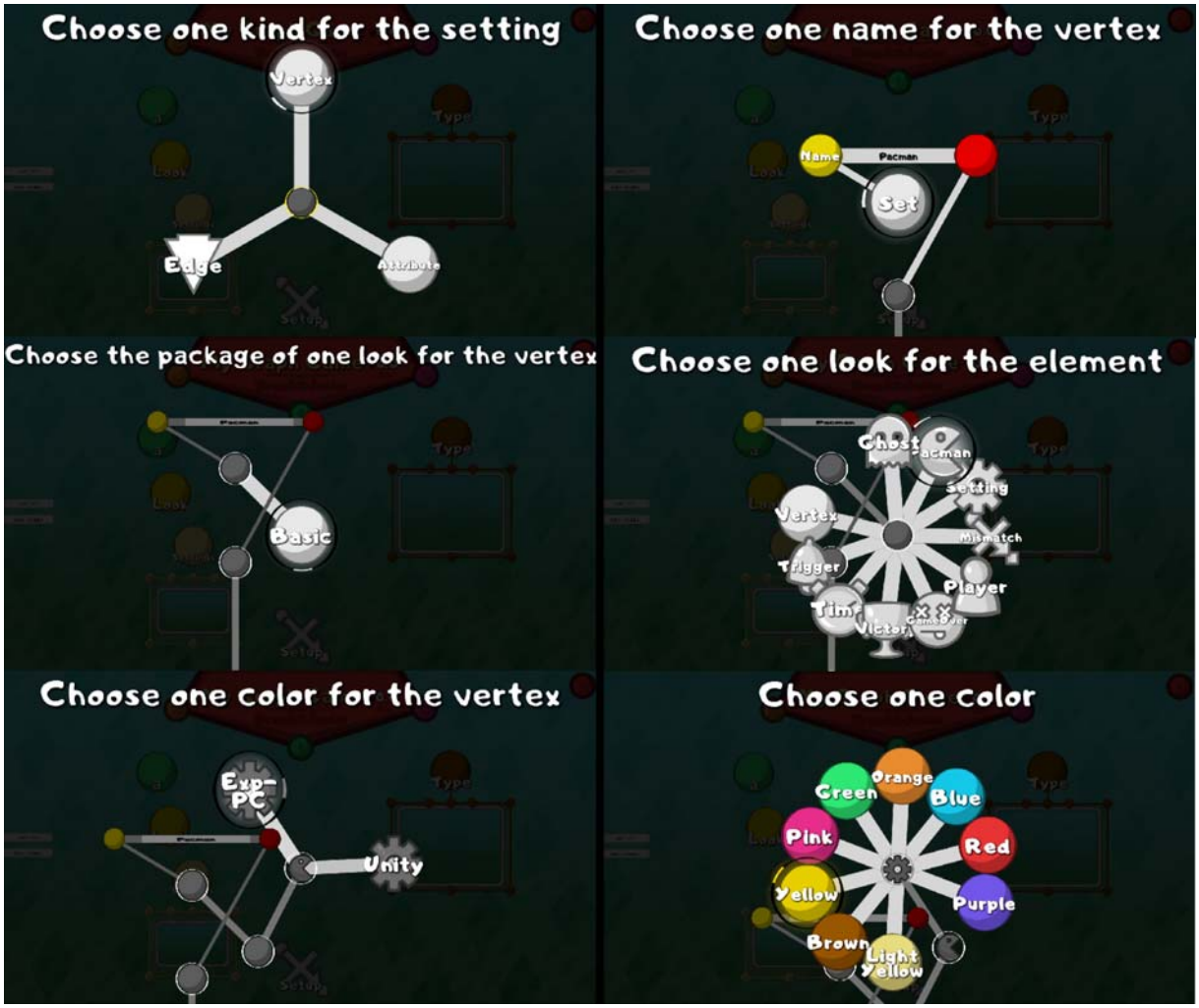


Figure 12 – GrameStation's definer sequence for creating a pacman type.

.

How do I specify? Internally, Grame Builder works through a parser reading and executing command lines, but a grame designer is not supposed to code. A decision tree system, named the **definer**, is called every time someone wants to build or edit anything in the module. It asks the user for consecutive choices that build up a complete definition (a command line), as a result, this constructs a graph linking the new

choices with the previous. An example of the definer is shown in Figure 12: a sequence of expansions in a decision tree to build the type for pacman. From left to right, top to bottom, first we select that we are building a vertex, then we enter its name, choose the package of the look (there is only the basic set available), choose the look, choose the package of the color, then the color.

3.3.1 Gramers

Gruly. The UI PA, Gruly, the hint scanner, was designed as a living VR glasses with the superpower to scan and project information in her vision, like the robotic suit of the famous superhero Ironman. She is meant to be the tourist guide of the graph universe GrameStation enables the user to enter. Gruly will lend the user her power to scan the screen for hints, being an always-available resource to understand what is on the screen.

Figure 13 – Gruly, the Hint Scanner

.

Grimone The GG PA, Grimone, the pocket advisor, was designed figuratively as a living handbook, and literally as a living controller with the superpower of popping out of anyone's pocket at any time. She is meant to be the engineer who supervises the construction of grames in the graph universe. Grimone switches her helmet and eyes according to your grame, giving success, warning and error messages about your project. She is also the one in charge of explaining errors while the user is playing grames.

Dr. Grafoss. The CT PA, Dr. Grafoss, the cartridge model checker, was designed as a living "game boy color" console with the superpower to analyse any data by making a cartridge of it and plugging it into her backpack. She is meant to be the scientist who promotes experiments to check if everything in the graph universe is meeting their specifications. Grafoss makes and plugs cartridges of your project, revealing all kinds of data about it, including the Grame CTScore and the CompaCT log.

Master Graz The TA PA, Master Graz, the silent therapist, was designed as a living kindle with the superpower to know and understand everything that is written, no matter the language, how confusing or complex. He is meant to be the librarian who tells the

Figure 14 – Grimone, the Pocket Advisor

.

Figure 15 – Dr Grafoss, the Cartridge Model Checker

.

stories about the creations of all things of the graph universe. But in his spare time, he is the therapist who graphs visit when their life isn't colorful. As a therapist, he is a good librarian: with a no-speakers policy and always remembering he is not a doctor, he is a master! Graz asks you to take notes on your feelings about new things and changes in your life, or at least, in your grame. So he can take notes on your notes.

Figure 16 – Master Graz, the Silent Therapist

.

Graphony. The abstraction PA, Graphony, the gift wrapper, was designed as a living 3D printer with the superpower to wrap anything, physical or abstract, no matter size or complexity. He is meant to be the Santa Claus of the graph universe, who distributes all kinds of mysterious presents hidden inside the wrappers he prints. Graphony helps

the user with the creation of wrappers and notify any problems with them or opportunities to use them.

Figure 17 – Graphony, the Gift Wrapper

.

3.4 Conclusion

A platform for specification. The GrameStation is an effort to revive the specification stage and automatically bring specifications to the real world (final product execution). More than that, some of the skills often claimed by the CT movement are related to GG aspects: they raise the level of abstraction by being able to get completely rid of texts; and by being declarative, where you just specify key graphs (a condition and a consequence), rather than sequencing orders to describe how to achieve them; the idea of matching is one of the meanings of abstraction – selecting only the necessary elements for an event of interest and temporarily ignoring the rest; matching is also pretty straight forward a pattern recognition process; rules are implications, conditionals – if LHS, then RHS; types are generalizations of elements, while rules are generalizations of behaviors of subgraphs; it is naturally a parallel model; it is based on well-known data structures – graphs; and the match being required for rule application implies that GG execution is compulsorily an evaluation task, you cannot disassociate an operation from examining its context.

Aligning to trends. GrameStation is a platform to use graphs for creating games, merging the benefits of GBL with GG, such as: harnessing 21st century skills; following the innovative standards of Education 4.0; being part of modern culture and a huge industry part of our daily lives; allowing PBL and PjBL while flirting with the maker movement; inducing the state of flow; shedding a light on specification and the roots of computing; and being visual and intuitive. At last, we cannot neglect CT assessment, this area still needs further research and GrameStation might be very helpful by automatizing data gathering and allowing CT assessment through the association with GG concept.

PS1: The graming approach, playing and making games specified as graph

grammars.

Recap. This chapter presented how we reached the conclusion that the proposed solution above would be a reasonable approach aiming at our goals. In this regard, we explored why we should harness the power of gaming in education; what GG are and could offer us; how we brought those two together with the development of an educational game engine based on GG; and how additional features, such as pedagogical agents, could be added to enhance its capabilities.

4 WRAPPERS: BRINGING LAYERS OF ABSTRACTION TO GRAPH GRAMMARS

SG2: Make available tools that support dealing with LoA within an engaging educational environment where young students could exercise CT skills beyond class time.

Content Distribution. In this chapter you will learn how we approached the research specific goal above, understanding how we reached the the proposed solution that concludes the chapter. In this regard, the first section introduces what is and why we choose such approach. The second section reveals the methodological apparatus used. The third section showcases the results reached. The fourth section concludes the chapter by discussing implications and future work.

4.1 Introduction

What can I Learn Here? This section situates the context of abstraction within CSEd and CT, defines LoA and discusses why we developed a tool for approaching LoA.

Abstraction for CT. The importance of abstraction in CT cannot be overstated, as mentioned in a SLR on abstraction, “it has been the most significant component of CT in empirical studies measuring learners’ CT development” (EZEAMUZIE; LEUNG; TING, 2022). However, it is safe to assume that developing this skill requires deliberate and strategic efforts in educational settings. As students grapple with increasingly intricate problems, they must be equipped with proper tools and resources to cultivate their ability to construct and navigate multiple layers of abstraction. In this regard, we shed light on the critical skill of abstraction in CT and present an effort to respond to the pressing need for specialized tools and resources to foster its development (MIROLO et al., 2022). We present a feature that enables and emphasizes modeling layers of abstraction in an educational game-based learning platform targeting k-12 education, but suitable for all ages. Throughout the paper we explore the following research questions:

1. What features or structures could support the modeling and management of layers of abstraction in GrameStation?

2. How could these features foster CT? That is, how could these features induce or influence learners to solve problems using strategies and abilities aligned with CT?

What the literature says. A recent overview (MIROLO et al., 2022) and a SLR (EZEAMUZIE; LEUNG; TING, 2022) on abstraction in CSE showed that the term "abstraction" has multiple characterizations and is tied to various topics in CS, as: problem formulation; extracting similarities; ignoring non-essential features; decomposition; computational abstractions; programming languages and paradigms; generalisation and parametrization; procedural and data abstraction; information hiding; and abstraction layers. The overview also mentions that abstraction is rarely explicitly approached, as expected from a very broad term. Instead, it is expected to emerge from various tasks, mainly revolving around modeling and programming, which is how it has been mainly approached (KAKAVAS; UGOLINI, 2019; MIROLO et al., 2022). Frameworks explicitly targeting abstraction have been presented in the literature (ARMONI, 2013; QIAN; CHOI, 2023). But they are theoretical guidances for educators to create tools/conduct activities in a way abstraction is favoured, rather than practical tools to be used by the learners.

LoA definition. On top of that, here we are focusing on a specific aspect of abstraction: **layers of abstraction**, which refers to the hierarchical organization of complex systems or problems into multiple levels of representation, with each level hiding unnecessary details and exposing only the essential elements needed for a particular purpose. This approach helps manage the inherent complexity of systems and enables problem solvers to reason at various levels of granularity.

Working with LoA. To the best of our knowledge, tools directed toward working with layers of abstraction in K-12 education are scarce. Arguably, visual programming languages and other CSE modeling/programming environments are able to handle some form of layered abstraction, but as a minor or implicit feature that does not receive much attention. Beyond programming, activities introducing notions of layers of abstraction approach it by showing multiple forms of representation of the same problem. For instance, in a chemistry class on the natural carbon cycle, students are shown a real-world macroscopic phenomenon; the scientific notation of the chemical reactions happening there; a computer-generated simulation of the phenomenon; and the simulation's source code. (GAUTAM; BORTZ; TATAR, 2020).

4.2 Abstract Hierarchical Graph Grammars

What can I learn here? This section presents the theoretical foundation that supports bringing LoA to GG. It explains the theory of hierarchical GG and provides the basic notions.

Hierarchical GG. In order to bring LoA to this GG environment, we developed the concept of "wrappers", that behave similarly to a subgraph. The underlying theoretical framework used comes from the abstract **Hierarchical Graph Grammars (HGG)** (BUSATTO; KREOWSKI; KUSKE, 2005), which provides a hierarchical organization on top of a base graph by using two additional graphs. One graph defines the relationship between packages, which are elements that will organize the elements of the base graph. And the other graph indicates in which package each base element is contained. For instance, Figure 18 shows a hierarchical graph: the base graph on the left defines a snake, a bear and a bird in a grassy land adjacent to a beach and a mountain; the hierarchy graph on the middle defines a package of animals and another of lands; and the coupling graph on the right defines the containment relations between base elements and hierarchy packages.

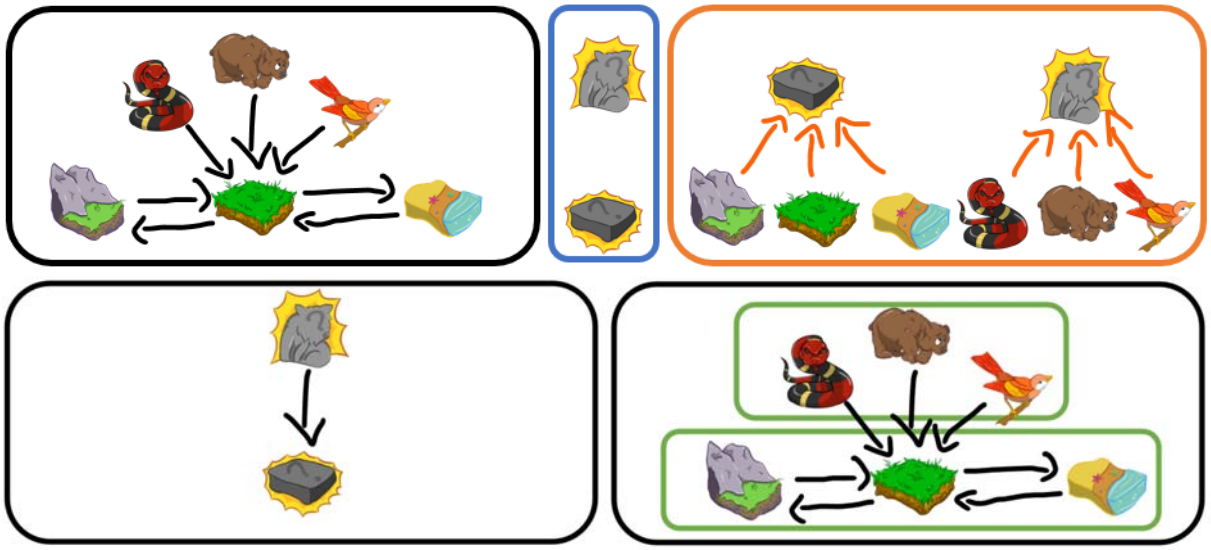


Figure 18 – Example of hierarchical graph as three graphs (top) and their simplified representation (bottom) as one, either collapsed (left) or expanded (right).

Object-oriented GG. Throughout the paper we will discuss some desired topics on object orientation. While we acknowledge the existence of **Object-Oriented Graph Grammars** (FERREIRA, 2005; FERREIRA; FOSS; RIBEIRO, 2007), we didn't want to bound all activities to object orientation nor rework the whole engine, adapting it to a different theory. The HGG approach was chosen because it allowed us to add the grouping information without compromising all definitions from the theoretical framework GrameStation already relied on. However, it felt overwhelming to always show the user two additional graphs just for that. Therefore, we hid them, embedding the

information provided by the HGG into the visualization of the base graph. We implemented the “packages” from the theory as “wrappers” (see Figure 18), which are shown as regular vertices while collapsed (left) and regions behind their content while expanded (right).

4.3 Wrappers

What can I learn here? This section introduces how HGG were incorporated into GrameStation using wrappers and the abstractometer.

Abstractometer. To deliver control over the layers in a fun and engaging way, we used the “Abstractometer”, which shows how many layers of abstraction are present in your project and can be used to navigate through them by clicking in the respective section. Figure 19 shows the usual interface of the game builder module of GrameStation, used to design the games as GG, featuring: (1) a component area to switch between rules, type and initial graphs; (2) a main display that shows relevant info as name, icon and ID of the object currently selected, and action buttons to add, edit or erase elements; (3) the work area, where the graphs/rules are displayed; and (4) the new addition, the abstractometer.

Figure 19 – Interface of GrameStation with the Abstractometer.

Fostering abstraction skills Wrappers may be used to foster abstraction skills, to model using different LoA and to facilitate the visualization of all that. As an explanatory support, we will use an educational game to offer practical examples. The game was chosen due to the convenience of being designed as a GG, we refer to (SILVA JUNIOR; CAVALHEIRO; FOSS, 2017) for readers wanting further details, here we will

summarize the essentials. It is a turn-based strategy game where animals are trying to revitalize a recently deforested jungle. The players may apply simple rules representing the growth cycle of plants and their dispersion being carried by the animals. There are multiple goal states randomly drawn and assigned to each player at the start of each game. The player that manages to make the jungle look like their goal first wins the game.

Limitations. The game features four animals: a bird, a snake, a bear and a wolf. They do not have different behaviors in the game, which means they are functionally the same kind of thing. That is why they are modeled as a single type, animal. The different visual representations and names of the instances are just conventions to make the game more visually diverse, they are not a real distinction for the GG. If they were modeled as different things, each of their own type, we would quickly realize some inconvenience trying to model common behavior between them. For instance, the rule that moves an animal from one square to another would have to be designed four times, each using one of the animals. That is the reason why modeling different types for things that do not behave differently is considered bad design, but that is a common mistake our intuition leads us to do while modeling. Additionally, this inconvenience is inescapable if we are after things that are supposed to behave differently but also to have some common behavior.

Generalizing. To deal with that problem, we shall generalize the elements. Wrappers can be used to do that in a very intuitive way (see Figure 20): an Animal would be a wrapper containing either a Bird, a Bear, a Snake or a Wolf (1). Then, instead of four rules (2), we would need to have only one, using the Animal wrapper (3). The important thing to note here is that we are still able to make specific rules for each type of animal, which wouldn't be achievable if they were all the same type.

Differentiating. For instance, let's say the bird should be able to Fly (see Figure 21-1), which would be a special move allowing it to go from a square to another regardless of the connections between them (the original movement rule requires connected squares). If the animals were all of the same type, we would have no way to refer specifically to the Bird, then either all would be able to fly, or none would. With the wrappers, we can let only the Bird fly, and all animals (including the Bird) move. So, here we properly differentiated and generalized the original four animals of the game, allowing them to show both, general behavior to share amongst all of them, and specific behavior to further characterize each.

Refining. The other way round, breaking an abstract concept into more concrete ones, is also possible and useful. We will harness the opportunity to illustrate how relations (edges) can also be approached in different layers of abstraction. We created a specialized behavior for the Bird, but not a corresponding specialized relation. A Bear may not be able to reach all squares a Bird can, but if they are at the same square,

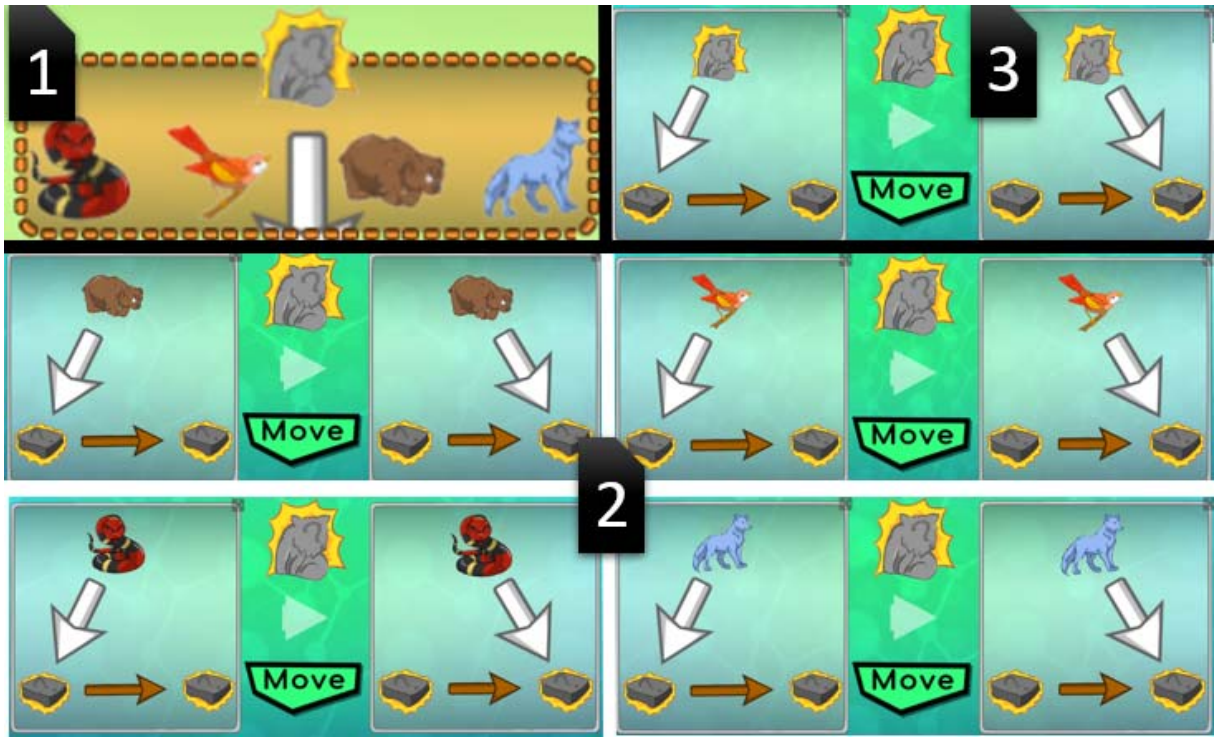


Figure 20 – Generalization from Bird/Bear/Snake/Wolf to Animal.

they stand the same kind of relation to that square (IsAt). Differentiating them could further enrich (and increase the complexity of) the game. For instance, the original rule to Gather fruits from trees could be made more strict, allowing only animals that are above the ground to reach and gather the fruits.

Specializing. This effect is reached in Figure 21 (see 2 and 3), turning the IsAt (white) edge into a wrapper containing edges as OnTheGround (green) and AboveTheGround (blue). Then Fly and Gather could use the specific edge AboveTheGround relation; while Move uses the generic wrapper IsAt (see Figure 20-3). This way Move could still be applied on both, those OnTheGround and those AboveTheGround, as nothing had changed, since it would require just the wrapper to match. So, here we properly broke down a generic relation into two specifics, allowing it to have some of its behaviors specialized, while preserving the rest as no changes were made. This is one of the most useful features of working with layers of abstractions, we can make changes in one without compromising the others.

Object Orientation. Those generalizations and refinements may superficially simulate some characteristics of **Object Orientation Programming (OOP)**. There isn't a real inheritance or polymorphism, since an element must always be mapped to another of the exact same type (not to any descendant type/child class as in OOP). However, if we see a wrapper and its content as a form of "composite" type, they are able to produce similar behavior in GGs. That takes advantage of the fact that matches require some elements, but have no say on the context of them[1]. That is, a general rule can

[1]That may vary for each Graph Grammar approach. In the one we have been using, the algebraic

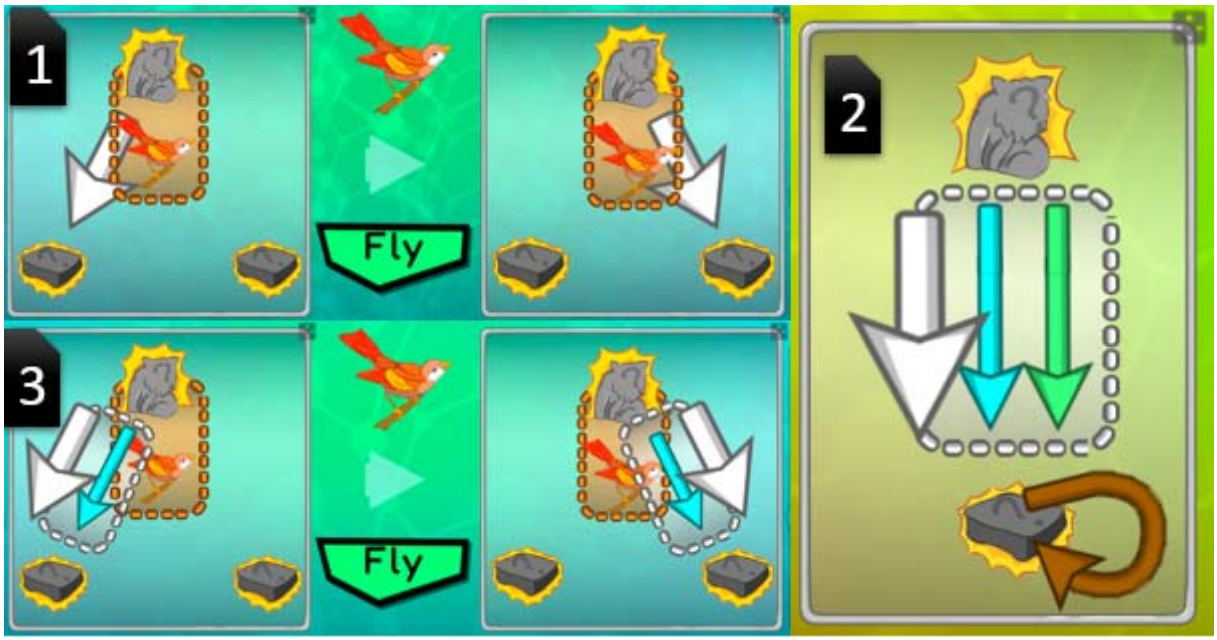


Figure 21 – Refining of the edge IsAt in the Fly rule.

require a wrapper regardless of its content.

Coherence. Simulating that in GGs require some attention to avoid faulty behaviors like deleting the content of the wrapper, but not the wrapper, or the other way round. This would be analogous to stripping a child class off its superclass properties or somehow instantiating an abstract superclass.

Modularizing. Another use for wrappers is as containers, which can be used to modularize the project, either just for organization and better visualization or being functional elements themselves. For instance, we could organize the squares together with their seeds, plants and trees into places (see Figure 23-1 and 2). That would allow not just a better visualization, especially in big projects, but using the place as an element itself, regardless of its content. For instance, we could add the sun to the game, illuminating one place at a time, and adding this as a requirement for the rule to make plants Grow into trees. The difference of using places instead of squares for this is that we could allow some places to have different content, or even change their content mid game, while keeping the sun mechanic intact.

Nesting. Wrappers can also be nested as long as they don't form a cycle. That is, there is no limit of how many layers of abstraction you could create with them. Nesting is generally only justified if the system gets too big or complex, which is not the case of our little game, here we forced nesting just to exemplify. As Figure 22 depicts, if we considered each different representation of the original game as a type and then generalized them (as we did for the animals), we would have a wrapper for

approach using Double-Pushout for rule applications, it is not entirely true. There is a context-based restriction of not being able to delete elements that would leave dangling edges (without a source or target).

each element. On top of that, we could consider the “Place” suggestion we mentioned, which would be a wrapper containing multiple wrappers (see also Figure 23-1).

Figure 22 – Type graph using wrappers for each element.

Navigating. The greatest advantage of wrappers is the clear visualization they offer to the modeler over the designed abstractions and their levels. In the platform, the default is a **free view**, where you can collapse or expand any wrapper at will, regardless of its layer. But we decided to empower and highlight the use of layers with the **layered view**, allowing the user to navigate through them seeing only the respective representations on each level (expanded wrappers are hidden, showing their content only). That can be seen in Figure 23, where: (1) shows the free view with all wrappers expanded; and the rest show layered views, (2) with the highest layer of abstraction; (3) with the middle layer; (4) with the lowest; and (5) shows how the rule Fly (Figure 21-3) is much cleaner under the layered view. Noteworthy, this Fly rule seen through the highest layer looks exactly like Move (Figure 20-3), since one is a specialization of the other.

Reasons to wrap. Theoretically, everything achievable with the wrappers is also achievable somehow using basic GG resources. For instance, instead of the wrapper, we could model Animal as a type of vertex and Bird/Bear/Snake/Wolf as types of edges, or other vertices connected to Animal. But **wrappers** help to ensure consistency and **provide a intuitive**, organized and explicit **way to model** those **abstractions**. What is appealing about wrappers is that they are not just an additional resource that would be approached only when layers of abstraction are the subject or learning objective of the activity. They actually solve an inconvenience that is quite often found when modeling GG: common behaviors among different types. Because GG requires strictly same-type matches[2], when a novice models a GG, they are likely to go through the

[2]This can vary depending on the GG approached, but holds true for the one used in the platform.

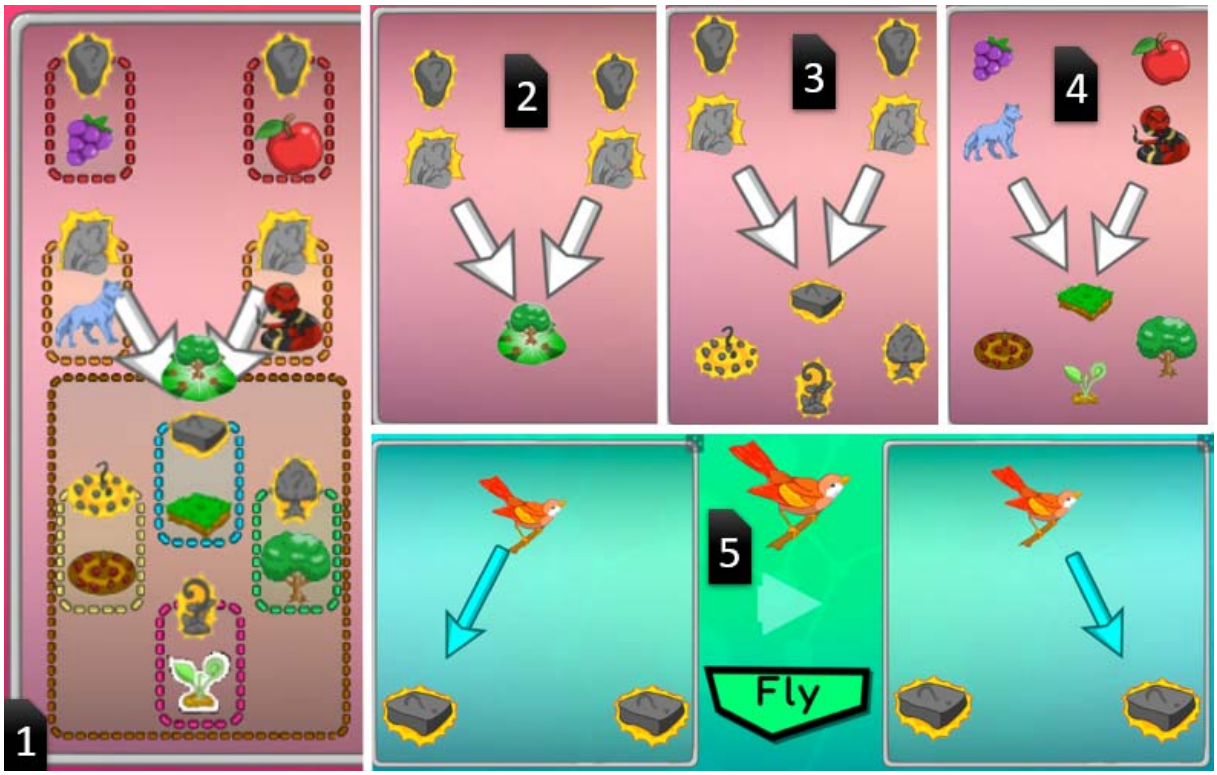


Figure 23 – Free and layered views of a graph and rule.

following path: making different types; wanting them all to do the same thing; then realizing that would require several similar rules, one for each type. That we see as **a natural way** to induce the learner **to understand** the reason and importance of **generalization**. Furthermore, as information in a graph is represented by shapes in a space, we naturally try to manage their positions to fit the screen. As a graph grows, fitting all elements in the screen becomes harder and harder, eventually impossible without overlaps. Wrappers allow us to compact (hide) multiple elements into a single container, simplifying the view of the whole. That we see as **a visual way to induce the learner to** understand the reason and importance of **abstraction of data** and **modularization**.

Conclusions. We discussed the capabilities of wrappers to foster abstraction layers and related skills (generalization, refinement, modularization, visualization and navigation). However, we recall it is a tool within a game engine, not a fully fetched activity on its own. It enables the creation of a wide range of activities around those topics: from putting the students to model entirely new games relying on the support of wrappers to organize and modularize it; to making them play strategically pre-made games where navigating between LoA is crucial. Creating such activities and making them available is part of our future work, but also possible for anyone that downloads the platform[3].

Future Work. Once these activities are at hand, the primary focus will be on empir-

[3] `https://wp.ufpel.edu.br/pensamentocomputacional/gramestation-pt/`

ically validating this proposal. We plan to conduct experiments in different educational settings to determine its effectiveness in fostering the ability: to model considering different LoA; refine; generalize; and modularize.

PS2: Wrappers, an abstract hierarchical graph grammars feature in GrameStation, an educational game engine based on graph grammars.

Recap. This chapter presented how we reached the conclusion that the proposed solution above would be a reasonable approach aiming at our goals. In this regard, we explored alternatives to bring LoA to GG, namely HGG and OOGG; presented the theory and main notions of HGG; how wrappers incorporate them into GrameStation in a way that is easy to visualize and operate; and the modeling skills that could be fostered with them, such as generalization, refinement, modularization, visualization and navigation.

5 ABSTRACTION LAND

SG3: Develop a psychometric model to assess a CT skill.
SG4: Design an intervention where the alternative approach can make use of the educational environment to foster the CT skill and to be assessed through the psychometric model.

Content Distribution. In this chapter you will learn how we approached the research specific goals above, understanding how we reached the proposed solutions that conclude the chapter. In this regard, the first section introduces what is and why we choose such approach. The second, third, fourth and fifth sections reveal the respective methodological apparatus, results and discussions for each model designed within the psychometric framework followed.

5.1 Introduction

What can I learn here? This section situates the role of psychometrics in the context of computing education and CT, discusses the rationale for using a psychometric model and introduces the one used for the activity, briefly presenting the framework that guided our work.

Computing education. As computing education evolves in compulsory education, the development of abstraction skills becomes essential for preparing students for a digital future. Abstraction, the ability to simplify complex systems by focusing on core elements, is crucial for CT, which lies at the heart of modern problem-solving. However, introducing abstraction to young learners is challenging due to its intangible nature. Traditional educational methods often struggle to convey such abstract concepts, making it difficult for children and teenagers to grasp and apply them effectively.

Importance of CT. All CT skills, especially abstraction, equip students with the ability to break down problems, identify patterns, and devise creative solutions, key to thriving in the digital era. However, the abstract nature of these skills contrasts with the more concrete and tangible learning experiences common in K-12 education. Traditional teaching methods, which emphasize memorization and factual knowledge, fall

short of nurturing the abstract reasoning needed to recognize patterns, create algorithms, and develop computational solutions.

Operationalizing abstraction. Assessing abstraction skills adds another layer of complexity to computing education. Unlike factual knowledge, which can be evaluated with standard tests, abstraction requires assessing students' ability to apply knowledge in new situations, decompose problems, and synthesize solutions. Conventional assessment methods struggle to capture these nuanced cognitive processes. This calls for a more adaptive and holistic framework that reflects the unique demands of abstract thinking in computational education, enabling educators to effectively evaluate and foster these critical skills in students.

Ad-hoc development. In today's rapidly evolving educational landscape, assessment is a critical tool for evaluating students' knowledge, skills, and abilities. This systematic evaluation not only influences learning outcomes but also helps educators and policymakers measure the effectiveness of educational systems. As individualized learning and technological advancements reshape education, the role of assessment methodologies has grown in importance, with a particular emphasis on evidentiary reasoning—collecting and analyzing data to provide deeper insights into students' learning processes and critical thinking abilities.

Principled approach. Traditional assessments, primarily based on standardized testing, often fail to capture the diverse talents and skills of students. In response, there has been a shift toward evidence-based assessment design. A principled approach to assessment emphasizes the importance of using valid, reliable, and fair methodologies to create instruments that accurately reflect student learning. In contrast to ad-hoc assessments, which are often designed in response to immediate instructional needs and lack rigorous foundations, principled methods ensure assessments are both coherent and consistent, preventing biases and ensuring a more comprehensive evaluation of student abilities.

Evidentiary reasoning. Evidentiary reasoning is crucial for enhancing the validity and reliability of assessments, offering a holistic understanding of student learning. This approach allows educators to consider a variety of evidence, including portfolios and performance-based assessments, fostering inclusivity and equity. By aligning assessments with the complex, multifaceted nature of learning in the 21st century, evidentiary reasoning supports the evaluation of critical skills like collaboration and creativity, which are often overlooked by traditional methods. This paper explores the advantages of principled approaches and evidentiary reasoning, advocating for their integration to ensure educational assessments reflect the true scope of student learning.

Evidence-centered design. "As powerful as it is in organizing thinking, simply having this conceptual point of view isn't as helpful as it could be in carrying out the actual work of designing and implementing assessments" (MISLEVY; ALMOND; LUKAS,

2003). The **E**vidence-**C**entered **D**esign **(ECD)** is a structured framework addressing this need, turning explicit the evidentiary reasoning underlying the design process of assessment instruments. (MISLEVY; ALMOND; LUKAS, 2003)

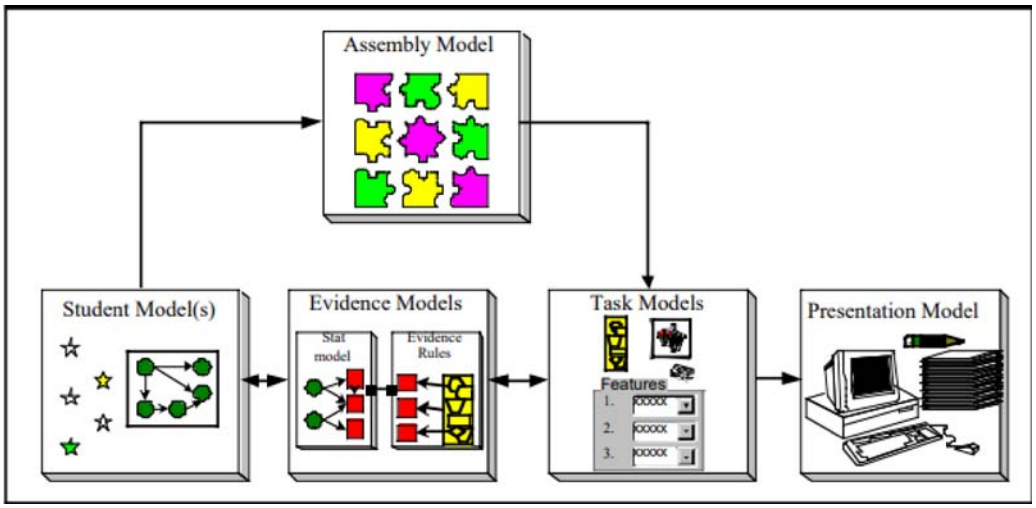


Figure 24 – The ECD model.
Source: (MISLEVY; ALMOND; LUKAS, 2003).

Conceptual overview. Generic and composite, e-ECD is a psychometric framework that guides the creation of a **C**onceptual **A**ssessment **F**ramework **(CAF)** organized in 5 models (see Figure 24): **proficiency**, about what we are measuring; **task**, about what we can observe; **evidence**, about how observables estimate measured constructs; **assembly**, about what from each of those previous three should be used for a specific intervention; and **presentation**, about how it will look and which means will be used to deliver it.

Missing model. In this work, we will not cover the Assembly model of ECD, since it is directed towards broader instruments or narrower interventions. For instance, if an exam uses a huge bank of questions, it would have an Assembly model responsible for designing which questions from the bank should be selected for a given intervention considering a given student. Each question from the bank is considered a different task, and therefore, this instrument has a gigantic Task model that must be consulted by the Assembly to make a feasible activity (applying all questions on every intervention is impracticable). This is not the case of our system, our Task model was designed to offer a single task targeting each KSA subskill. Our intention is to have each intervention using all tasks. Therefore, there is no need for an Assembly model.

Text Organization. Given the size, importance and independence of each model, we decided to avoid the classic Methods, Results and Discussion setup. Each of those sections would have to bring subsections respective to each model and it would break the continuity of the text. Therefore, we prepared a self-contained section for each model, wherein their specific methods, results and discussions are presented.

5.2 Proficiency Model

Introduction

What to measure. The proficiency model (also called student model) defines unobservable proficiency variables representing target **K**nowledge, **S**kills, and **A**bilities **(KSA)**[1], denoted by θ, which can be seen as a proportion of a domain of tasks the student is likely to answer/perform correctly. It is considered that θ can never be known with certainty because we don't get to observe it directly. Therefore, the knowledge about its value (at a given point in time) is expressed by a probability distribution across the range of values it might take (MISLEVY; ALMOND; LUKAS, 2003). On a conceptual level, we must clearly define the KSA each θ represents and their eventual relations.

KSA models. There are various ways to model KSA, they can be independent, whole variables like the **G**raduate **R**ecord **E**xamination **(GRE)** parameters: verbal reasoning; quantitative reasoning; and analytical writing. Or they can be a complex arrangement of subskills connected in a map or net, like Figure 25. In general, these variables result from experts conducting cognitive task analysis; content domain analysis; relying on a theory of knowledge; or research findings (ARIELI-ATTALI et al., 2019). Given our approach, we will conveniently represent the variables and their interconnections as a graph.

KSA design. The starting point to create our KSA model for layers of abstraction was a CT model that organizes the most commonly addressed terms related to CT into six major lines: abstraction, algorithms, decomposition, data, automation and evaluation (SILVA JUNIOR, 2020). That model was already based on a SLR on 28 sources defining CT, but due to its broad nature, we decided to complement it with two other reviews that specifically targeted Abstraction (EZEAMUZIE; LEUNG; TING, 2022; MIROLO et al., 2022). An initial draft drawing from them was then presented to a panel of CT experts for further discussion and polishment.

Big ideas underlying abstraction. Combining the abstraction part of a SLR-based CT model (SILVA JUNIOR, 2020) to the insights of an overview (MIROLO et al., 2022) and the results of a SLR (EZEAMUZIE; LEUNG; TING, 2022), both specifically targeting abstraction in the context of CSEd, we elected 6 topics permeating abstraction addressed by the literature. Table 2 organizes them by name, definition, importance to CS and how they have been approached on practice. From this compilation of topics, we understand they are all intertwined manifestations of abstraction. Therefore, it could be difficult or even wrong trying to isolate these concepts. However, in an effort to reach a more precise assessment (and learning objectives) for education, we proceeded by conceiving KSAs that are clearly distinct. Our hypothesis was that ad-

[1] The concept of KSA receives various names in different works, such as Focal Knowledge, Skills, and other Attributes (FKSAs) (MISLEVY; ALMOND; LUKAS, 2003); and Focal Concepts and Practices (FCP) (GROVER et al., 2017)

Figure 25 – CBAL middle school mathematics competency model.
Source: (ARIELI-ATTALI; CAYTON-HODGES, 2014).

dressing each of them individually could facilitate assessment, in the sense we would know more precisely what we are measuring.

Layers of Abstraction. It called our attention that the apparent perceived/recognized importance of **L**ayers **o**f **A**bstraction **(LoA)** contrasts with the lack of activities directly targeting it. Arguably, LoA are implicit on all other topics, as it was a byproduct of them. But we emphasize that LoA is one of the most iconic forms abstraction reveals itself in CS. Computing relied on solid LoA to reach unparalleled complexity reduced to manageable representations. A simple function written in a high-level programming language translates to several parts in the layer below. In turn, the same thing happens to each part. This happens again and again through several layers until we reach a humongous set of transistor switchings that would be completely impracticable to design directly. Thus, we built our KSA model through the lens of LoA, rescuing

its importance to CS.

LoA KSA model. Therefore, we had LoA as the focus of the KSA model, but we also intended to use it as the means to reach the rest of the topics found amongst the conceptualizations and operationalizations of abstraction. Our first draft (drawing from Table 2) resulted in a KSA with subskills all somehow related to modeling. However, modeling is arguably an advanced KSA we would expect as a final goal, synthesizing and employing everything that was learned, rather than an introductory KSA to be learned in a first contact. That led us to the creation of 3 other KSA focusing in the acquisition and consolidation of the comprehension of LoA, as well as their analysis. Those 3 are not necessarily “easier” than modeling, but they refer to understanding and operating upon pre-existing LoA, which makes them more suitable for establishing foundational knowledge. Neither they are prerequisites for modeling, but might exert some influence.

Table 2 – Topics permeating abstraction in CT literature.

Topic	Theory (Definitions in the literature), from (SILVA JUNIOR, 2020)	Rationale to teach (Importance to CS), from the authors, considering (MIROLO et al., 2022)	Practice (Operationalization), from (EZEAMUZIE; LEUNG; TING, 2022)
Simplification	"deciding what details we need to highlight and what details we can ignore" (WING, 2008); "creating something simple from something complicated" (ATMATZIDOU; DEMETRIADIS, 2016); "reducing the unnecessary detail" (CSIZMADIA et al., 2015)	Vital for managing complexity. It breaks down intricate problems into manageable parts in algorithm design, improves readability and maintainability in software engineering, and creates user-friendly interfaces in HCI. Simplification also aids in understanding and classifying problems in complexity theory, leading to more efficient, scalable, and user-friendly systems.	It is identified that "extracting relevant features" is one of the main ways abstraction has been operationalized
Pattern Recognition	"Identify patterns/rules underlying the data/information structure" (SHUTE; SUN; ASBELL-CLARKE, 2017); "identify common patterns" (ANGELI et al., 2016); "identifying and making use of patterns" (CSIZMADIA et al., 2015)	Vital for interpreting data and making predictions. It enables machine learning algorithms to identify trends, improves cybersecurity by detecting anomalies, and enhances natural language processing by understanding human language. Overall, pattern recognition is crucial for developing intelligent, secure, and efficient systems.	"discovering underlying patterns, creating classification rules and assembling the parts together are the core actions of learners in abstraction".
Generalization	"transferring this problem solving process to a wide variety of problems" (MANNILA et al., 2014); "The skill to formulate a solution in generic terms so that it can be applied to different problems" (ANGELI et al., 2016); "the step of recognising how small pieces may be reused and reapplied to similar or unique situations. (...) the ability to move from specific to broader applicability" (SELBY; WOOLLARD, 2014)	Crucial for creating adaptable solutions. In machine learning, it enables models to perform well on new, unseen data, enhancing predictive accuracy. In software development, writing generalized code promotes reusability and flexibility, making systems easier to maintain and extend. Generalization also aids in developing algorithms that can solve a wide range of problems, leading to more robust and versatile systems.	In th sense of transfer of problem solutions, abstraction appears in some studies being operationalized through "the ability to use solutions from past experience to solve new sets of problems with similar structures".
Decomposition	"Breaking down tasks into smaller, manageable parts" (ISTE; CSTA, 2011); "Dissect a complex problem/system into manageable parts. The divided parts are not random pieces, but functional elements that collectively comprise the whole system/problem." (SHUTE; SUN; ASBELL-CLARKE, 2017); "a way of thinking about artefacts in terms of their component parts. The parts can then be understood, solved, developed and evaluated separately." (BOCCONI et al., 2016)	Essential for managing complex systems. It involves breaking down a large problem into smaller, more manageable sub-problems, simplifying design and implementation. In software engineering, decomposition improves code organization and maintainability. In algorithm design, it facilitates efficient problem-solving through techniques like divide and conquer. Overall, decomposition leads to clearer, more efficient, and easier-to-manage systems.	Despite abstraction being conceptually defined mostly as a form of generalization or decomposition, the activities targeting it "often did not fit neatly" those categories. Decomposition, however, is often considered a separated CT component and operationalized as such, unrelated to abstraction.
Modeling/ Formulation	"Build models or simulations to represent how a system operates, and/or how a system will function in the future." (SHUTE; SUN; ASBELL-CLARKE, 2017); "choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable." (WING, 2006); "formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent." (WING, 2008)	Crucial for abstracting and solving complex problems. It involves creating representations of systems or processes, which aids in understanding, analysis, and prediction. In software engineering, modeling helps design and visualize system architecture. In machine learning, it enables the creation of models that can predict outcomes from data. Overall, modeling leads to better problem-solving, improved system design, and more accurate predictions.	The sophistication of programming concepts is a common way to assess abstraction, measuring how many/which elements (keywords, functions, VPL blocks) were used to model a solution. Additionally, the creation of alternative representations is another approach being used to pursue abstraction.
Layers of Abstraction	"The core CT concepts include, abstractions (the mental tools of computing, necessary to solve the problem), layers (problems need to be solved on different levels) and relationships between layers and abstractions" (WING, 2008); "Thinking in levels — Systems can be understood and analyzed from different perspectives, ranging from a micro-level view that considers the smallest elements of the system to a macrolevel view that considers the system as a whole. (...) insights about how micro-level behaviors lead to emergent macro-level patterns. (...) being able to black box the details of the underlying systematic interactions and focus on the system as a whole" (WEINTROP et al., 2016)	Fundamental for managing complexity and simplifying design. They involve organizing systems into hierarchical levels, where each layer provides a different level of detail and functionality. In software development, abstraction allows programmers to focus on high-level concepts without getting bogged down in implementation details. In computer architecture, layers of abstraction enable hardware and software to interact efficiently. Overall, layers of abstraction promote clarity, modularity, and scalability in designing and understanding complex systems.	It is recognized that activites work in different levels of abstraction, namely program and problem-level, but they rarely approached multiple at the same time. Noteworthy, none were about the levels themselves.

Model

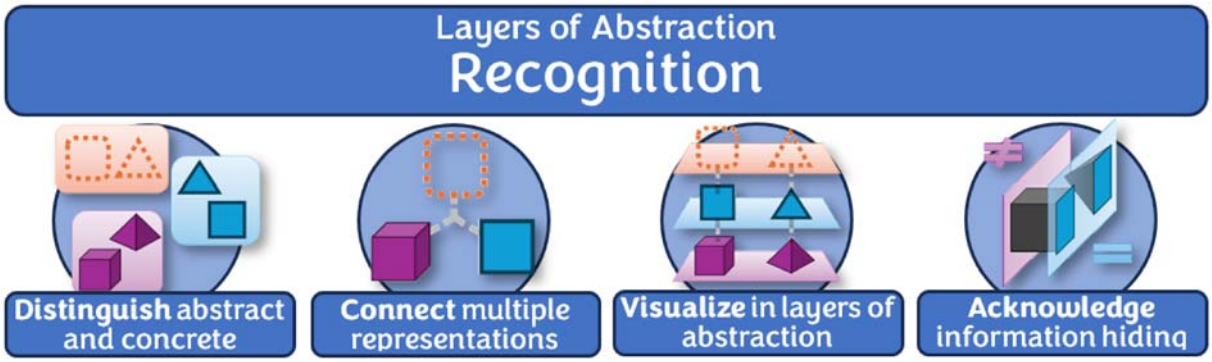


Figure 26 – LoA Recognition KSA.
Source: Elaborated by the author.

LoA Recognition. Let us consider the following definition for LoA: “the hierarchical organization of complex systems or problems into multiple levels of representation, with each level hiding unnecessary details and exposing only the essential elements needed for a particular purpose” (SILVA JUNIOR et al., 2023). The first KSA (Figure 26) then refers to the general comprehension of what is a LoA and the acknowledgment of their existence (or absence) in a given system/problem. Students with this introductory KSA are able to:

- **Distinguish abstract and concrete.** Realizing that things may have representations with different amounts of details. Being able to tell if something is closer to an idea (more abstract) or the reality (more concrete). Grouping things with similar levels of detail, or classifying which items belong to each group. Examples: Distinguishing between software (abstract) and hardware (concrete); or a map (abstract) and the physical terrain it represents (concrete).

- **Connect multiple representations.** Understanding that a single thing can be represented in different ways, that is, have multiple representations. Linking multiple representations to a single thing. Identifying all representations that can refer to the same thing. Examples: Connecting a flowchart, a pseudo-code and a visual programming language to the same algorithm; or a real-world phenomenon, an equation, and a computer simulation of the same chemical reaction.

- **Visualize in LoA.** Stratify a system into different views upon the same elements, establishing a hierarchy over the abstraction of their various representations. The higher levels are those abstracting more, thus with fewer details. The lower we go, the more details there will be. Examples: Visualizing living beings through kingdoms, classes, families, species and food-chain roles, with each individual consistently dispersed across the levels of organization.

- **Acknowledge information hiding** Demonstrating awareness about the fact that we hide/lose information when abstracting. Comprehending that things may be different at lower levels while being the same at higher levels. Examples: Acknowledging that the summary of a book can tell me the story very quickly, but there will be lots of details I would not know just by reading it; or that, when seeing two games as just "racing games", they may still be different despite having the same general goals and mechanics.

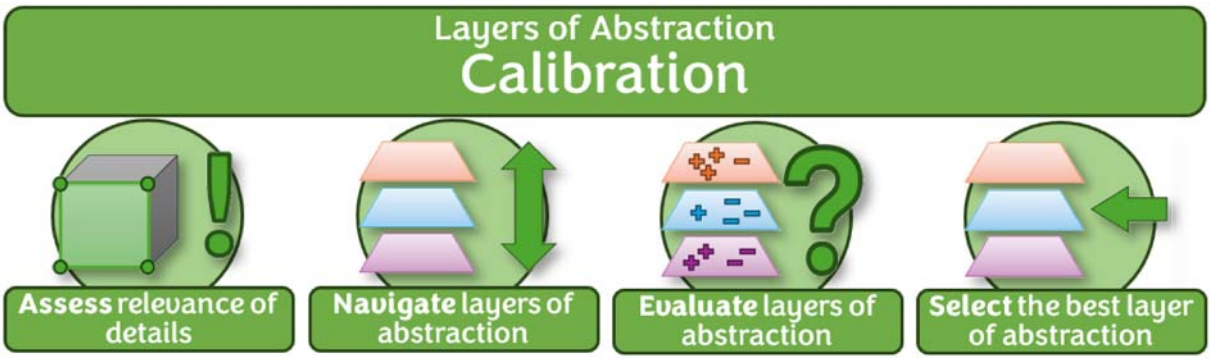


Figure 27 – LoA Calibration KSA.
Source: Elaborated by the author.

LoA Calibration. Once their existence is acknowledged, it is important to know how to navigate and select the appropriate layer for each task. Thus, the second KSA (Figure 27) refers to identifying in which level of abstraction one should work in order to reach a given solution. Alongside this, knowing how, when and why to operate in a given LoA allows us to deal with complexity, often considered the main point of abstraction. Students with this critical KSA are able to:

- **Assess the relevance of details.** Judging whether a detail or part is necessary or important for a given purpose. Deciding what could be hidden/forgotten and what must be shown/considered. Measuring the impact of a given detail or part on the solution. Examples: Assessing which sensors of a robot are necessary to simply follow a painted line when walking; or which historical events are crucial for understanding a particular era.

- **Navigate through LoA.** Switching between LoA without losing track of what is being represented. That means recognizing the connection between multiple representations and staying consistent to the object being analyzed when switching layers. Mapping the representations of a LoA into another. Examples: Navigating back and forth a flowchart of an algorithm and the functions or commands of its implementation; or a zoom in a GPS during a travel, triggering changes on the elements shown, such as the city names and main roads, nearby restaurants and touristic places, or accident reports and speed limits.

- **Evaluate LoA.** Identifying what a LoA offers and what it demands. Recognizing if a problem is solvable in a given LoA. Understanding in which LoA the requirements for the solution can be found. Measuring the limits of what can be done and what cannot in a certain LoA. Calculating the costs and benefits of a LoA. Reasoning about characteristics of a LoA and how problems are impacted by being approached in them. Examples: Evaluating the advantages and disadvantages of representing data in tabular form versus graphical form; or of using the predefined functions of the microwave versus setting everything manually.
- **Select the best LoA** Choosing fitting criteria to decide in which LoA to operate. Reducing a set of attributes/features of each LoA to a comparable suitability rate for a specific problem. Weighting the pros and cons of each LoA. Comprehending that even when solvable in a given LoA, another LoA might suit the problem better. Justifying the choice of a given LoA for a given solution. Examples: Selecting between using a simple simulation with toy cars, a set of equations ignoring air resistance and friction or a complex physics simulation with thousands of calculations to find out if two cars would collide when crossing roads.

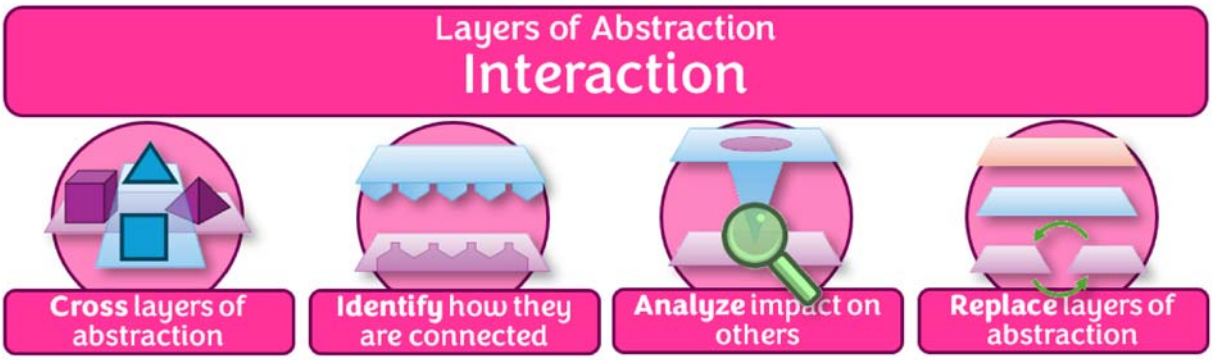


Figure 28 – LoA Interaction KSA.
Source: Elaborated by the author.

LoA Interaction. Up to this point, LoA can be considered individually, isolated from the others. We considered navigating between them, but we were just jumping from one to the other without looking at the bridges between them. We also didn't consider yet the eventual situations where they cannot be isolated from each other. Thus, the third KSA (Figure 28) refers to dealing with multiple LoA at the same time and the interfaces between them. Students with this KSA are able to:

- **Cross LoA.** Considering multiple LoA at the same time, that is, working simultaneously with constructs from different LoA. We may need more details from a part while another may keep its intricacies hidden. Examples: Crossing native operations, such as variable attributions, with function calls (that each may involve thousands of other operations); or variables (unknown expressions) together with constants when solving equations.

- **Identify how LoA are connected.** Recognizing the interfaces between LoA, what makes a set of constructs suitable to represent another in a different layer. Examples: Identifying that what connects a logic gate to a lower LoA is processing an input into an output according to its truth table, regardless if it is an association of relays, valves, transitions or other technologies (even humans following strict instructions).
- **Analyze the impact on other LoA.** Foreseeing the consequences of replacing what is in a LoA on the rest of the solution. Identifying the influence of the content of a LoA into the content of another. Spotting independence between layers, where replacing what is in a LoA does not compromise what is in another. Examples: Analyzing how the food chain would be impacted if instead of a shrub we had a coniferous tree as the producer. To wonder if, for instance, we would end up with the same number or variety of predators; or how a change in the type of exercises you make impacts the schedule you have for other tasks (beyond exercising) in your overall daily routine.
- **Replace the content of a LoA without modifying the rest.** Comprehend that part or the whole content of a LoA might be substituted for another without having the change the whole solution. Evaluating which of the candidates for each LoA is best suited for a given problem. Examples: Replacing a JavaScript code with a Python one to implement the algorithm you have specified; or a bag with marbles instead of a balloon with sand to simulate/represent a fluid.

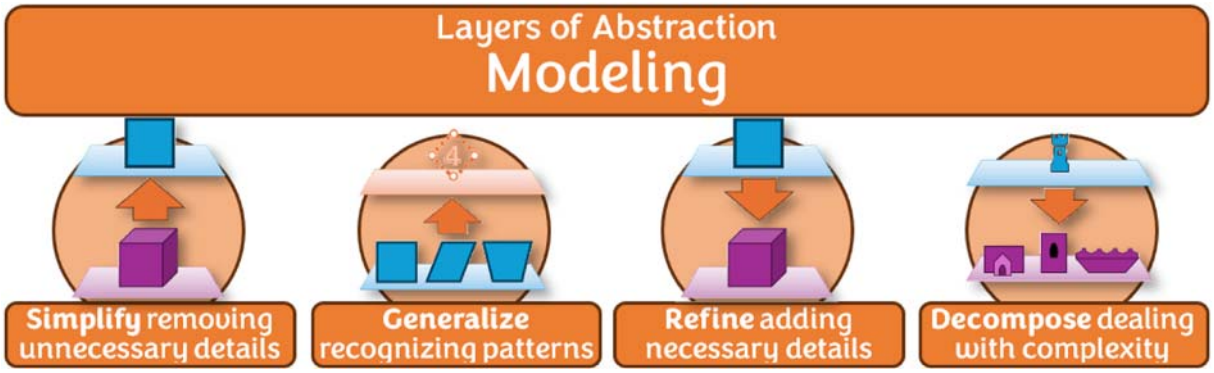


Figure 29 – LoA Modeling KSA.
Source: Elaborated by the author.

LoA Modeling. The previous KSAs operate upon LoA that are given, pre-existing or implicit. So, the fourth KSA (Figure 29) refers to the creation of LoA and their constructs, to design constructs that represent another in a different LoA. Students with this creative KSA are able to:

- **Simplify removing unnecessary details.** Devising new constructs by extracting only the important features from others. Making a new construct, that is some-

thing on its own, out of the main characteristics of another. Examples: Simplifying a car, which is a very complex machine with several parts in the real world, as a rectangle that can accelerate and rotate changing its direction in a racing game; or a volcano eruption, a complex natural phenomenon involving various geological, physical and chemical processes, as baking soda and vinegar.

- **Generalize recognizing patterns.** Finding a new (more abstract) construct that maintains a set of essential features while being able to exist in other contexts, represented by (more concrete) constructs (that share those features as a pattern). This can be reached by either recognizing a pattern among multiple constructs or by defining which features of a given construct should be constant (and which could vary), proposing a new pattern to be instantiated. Examples: Generalizing a function to sum any two numbers a and b, instead of 3 and 5; or the process of making lemon juice to make juice of any fruit.

- **Refine adding necessary details.** Devising new constructs by adding new relevant information into another. The opposite of simplification, detailing can be necessary to reach concrete solutions by iteratively pushing it closer to the real world. Examples: Refining a grocery list, where you put all the dishes you want to make, into the ingredients they require; or a kinematic equation, where you described the actors in a given scenario, into the mathematical description of the vectors of all forces acting upon them.

- **Decompose dealing with complexity.** Breaking down a construct into smaller, more manageable parts. Despite technically being a refinement instead of a simplification, decomposition has the goal of reaching simpler units that are easier to deal with, individually. This benefit may outweigh the fact that you end up with more information than before, not just because each piece is more treatable, but because you could save a lot of effort by reusing pieces. Examples: Decomposing a movie into each pixel of each frame to make it black and white, then reusing the solution for the first pixel to all pixels in all frames; or a complex choreography into short dance moves that, once you learn and train individually, you won't forget.

KSA Graph. The resulting KSA graph depicted in Figure 30 have the following vertices: LoA Recognition, understanding granularity, LoA structure and similarity in LoA; LoA Calibration, navigating, assessing and selecting appropriate LoA; LoA Interaction, considering multiple LoA, their independence and interrelationship; and LoA Modeling, simplifying, generalizing, refining and decomposing.

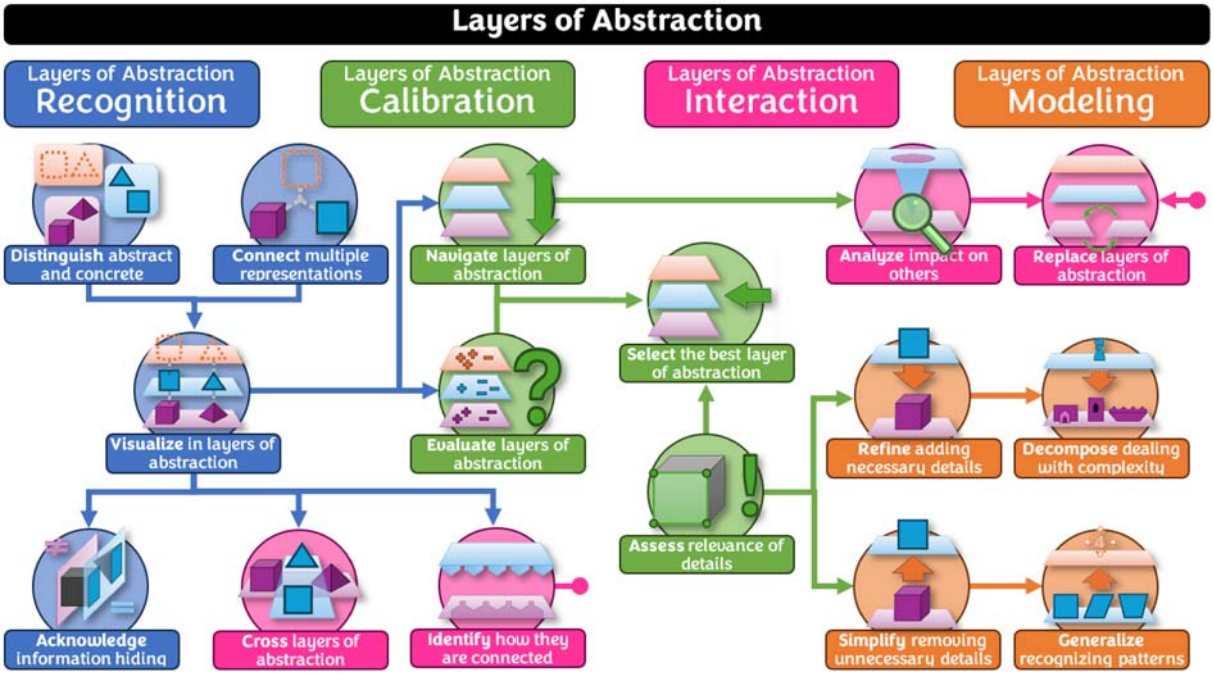


Figure 30 – The KSA graph of LoA.
Source: Elaborated by the author.

Validation

Advisory Panel. This draft was then reviewed by an advisory panel of 3 professors from different universities: an expert in CS education with a background in psychology and linguistics; an expert in artificial intelligence and educational games with a background in electrical engineering; and an expert in fuzzy logic, quantum computing and CT, with a background in mathematics. They received the KSA graph and were asked to assess its quality by leaving commentaries, suggestions and classifying it as: **Unacceptable (1)**, meaning it would do more harm than good to use it without adjustments; **Limited (2)**, meaning it is acceptable to use without adjustments, but it would still lack something to properly fit its purpose; **Adequate (3)**, meaning it already suffices, but there is room for improvement; and **Ideal (4)**, meaning it should be used as is, no further adjustments are suggested or necessary; for each of the following criteria:

- **Clarity and Structure**: How clearly are the competencies defined? Is the structure of the KSA graph logical and intuitive? Are the relationships between different competencies evident?
- **Coverage**: Does the KSA graph cover a broad range of essential skills and abilities in computing related to abstraction? Are there any major gaps or redundancies in the coverage?
- **Granularity**: Are the competencies appropriately distributed in terms of Granularity? Should one be broken down into more, or various merged into one?

- **Relevance and Alignment**: How well do the competencies align with the learning objectives for students in the computing domain? To what extent does the graph reflect the KSA required in industry, education and research settings? Are there opportunities for students to apply these skills in real-world projects?
- **Interdisciplinarity**: Does the KSA graph give space to incorporate relevant interdisciplinary connections with other fields? Are there opportunities for students to apply computing skills in diverse contexts?

Setup. The assessment was completed asynchronously using an online form (Appendix A), where the advisory board rated the graph according to the rubrics above, gave their Informed Consent (IC) and chose how to be acknowledged for their contributions. Beyond those, we collected pertinent information, such as: their field of expertise; years of experience with computing education theory (research); and practice (teaching); and the ISCED levels of education they have acted. All three members of the board declared to have computer science as their field of expertise and more than 10 years of experience teaching CS. Two of them also have more than 10 years of experience CSEd research and work with tertiary education only, the other one has between 5 and 10 years of experience with research, but has acted across all ISCED levels of education.

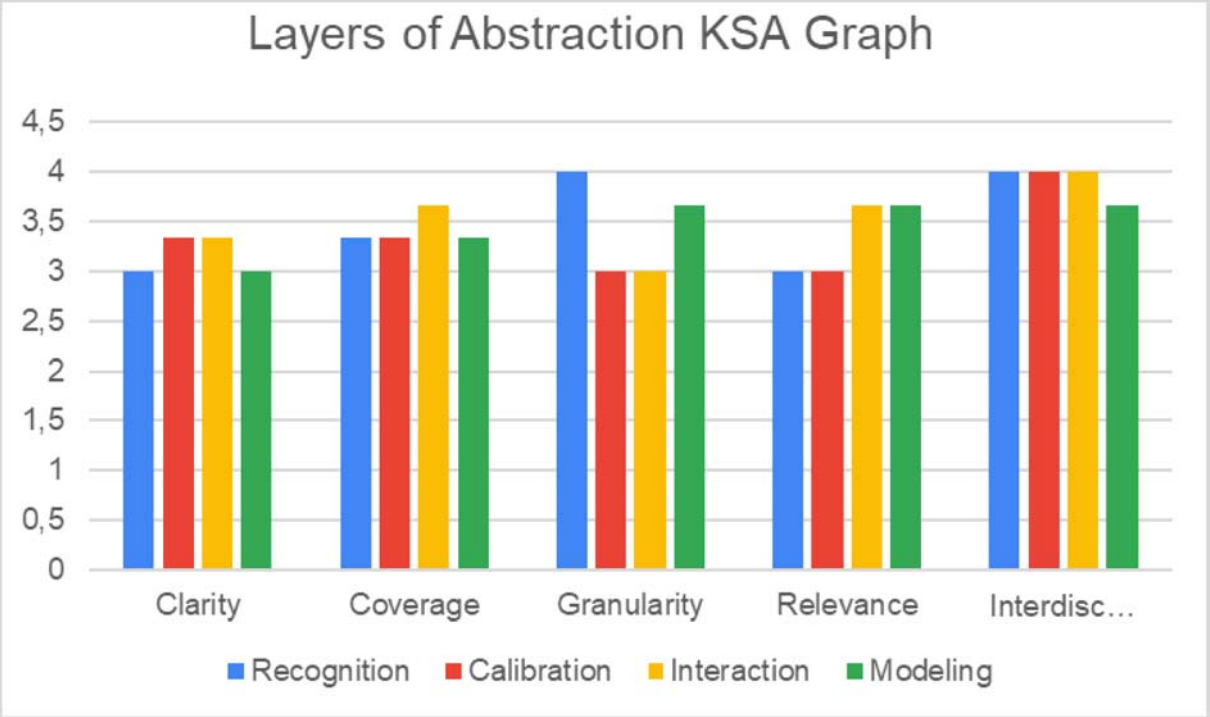


Figure 31 – Average ratings of the KSA graph of LoA.
Source: Elaborated by the author.

Results. Considering the classifications mentioned above, from 1 (unacceptable) to 4 (ideal), Figure 31 shows the average rating of each KSA for each criterion. By

the average of all reviewers, no KSA was deemed less than 3 (adequate). We can say by those ratings that the strength of the graph is interdisciplinarity with an average of 3.917, while the weakness is clarity with 3.167. Relevance got 3.33, while Coverage and Granularity tied with 3.417. Considering the average of all criteria, all KSAs performed very similarly. There was a difference of 0.2 between the best performing (Interaction, with 3.53) and worst (Calibration, with 3.33) while the two others (Recognition and Modeling) tied with 3.46.

Inter-rater reliability. Preserving the reviewers' anonymity, Figure 32 displays how each of them, named A, B and C for convenience, rated each KSA for each criterion. Raters A and B had an average difference of rating of 0.3 (out of a maximum difference of 3), and an absolute agreement of 70% (14/20). Raters B and C had an average difference of rating of 0.95 and absolute agreement of 35% (7/20). Raters A and C had an average difference of rating of 0.65 and absolute agreement of 60% (12/20). Raters A, B and C together (the sum of the previous) had an average difference of rating of 1.9 (out of a maximum difference of 6), standard deviation of 0.54, and absolute agreement of 35% (7/20).

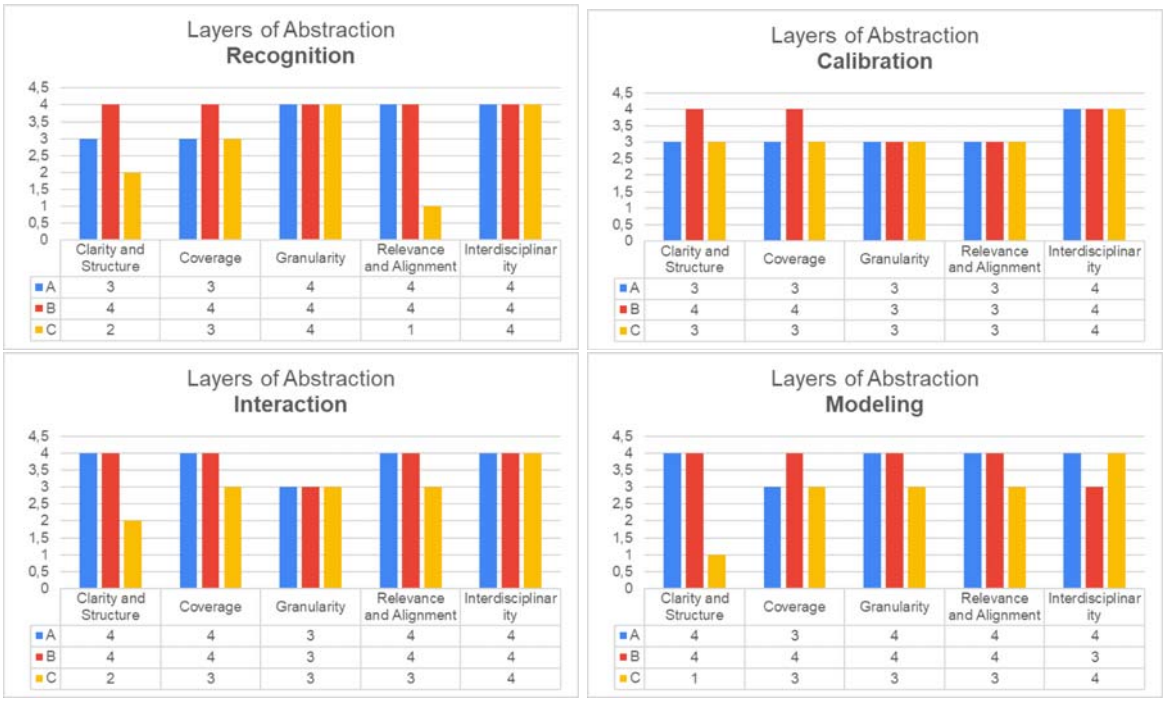


Figure 32 – Evaluation of the KSA graph of LoA according to reviewers A, B and C.
Source: Elaborated by the author.

Interpretation. Clarity being the worst rated criterion was expected due to abstraction being a soft conceptual idea that, "in its broad sense does not lend itself to a full comprehensive and concrete definition, nor are there any specific rules concerned with the application of abstraction" (KRAMER, 2006). That is one of the biggest challenges while working with abstraction in a more formal environment. However, being able to clearly define and explain KSAs related to abstraction is crucial to the purposes of the

KSA graph. Thus, we will deepen the discussion on clarity with reviewer C, who rated it below 3 (adequate) for LoA recognition, interaction and modeling. On the other hand, interdisciplinarity was rated really close to 4 (ideal), because “The competencies are so broad that they are easily interdisciplinary”, as put by a reviewer. The numbers on inter-rater reliability indicated that, despite having a low absolute agreement, the difference of rating and standard deviation were not high, meaning they often disagreed, but by little. Noteworthy, all exceptions to that were from reviewer C disagreeing with the others on the clarity of LoA modeling and relevance of LoA recognition, which will also be discussed ahead.

Discussion. The feedback provided by the reviewers included open-ended answers, which allowed us to better understand the rates and start relevant discussions. For instance, the format of the evaluation itself was questioned. Some comments suggested that criteria such as coverage, granularity, relevance and alignment should be asked only in the end, about the graph as a whole, not for each KSA. For instance, the following comment was made about coverage of LoA recognition (first KSA): “It is very difficult to assess this in isolation. Things that I might think are missing could be covered in other parts of the KSA graph”. Several similar comments were found along the form. Other comments mentioned that the rubrics might have been a little demanding, as expressed by the following one after rating 3 (adequate): “Nothing major seems wrong. (You have set the "Ideal" bar very high!)”.

Similarity. Some of the subskills were pointed as being too similar, enough for the raters to understand them as the same “Are ‘Select the best LoA’ and ‘Evaluate LoA’ really different? Don’t we need to evaluate to select the best?”. We actually have only four KSA, subskills are an additional effort to isolate certain aspects or even steps of each KSA. So, it is natural and expected that subskills feel similar, they are closely related. They are an attempt to facilitate isolating the targets/goals of tasks, minimizing noise. In this case, selecting the best layer refers to the processing (comparison) of available information for decision-making, which can be done intuitively (different than randomly); while evaluation refers to the measurement/calculation/demonstration of the meaningful information used to compare. Therefore, we usually need to evaluate to select the best, but not necessarily.

Broad range of application. Another question that was raised in the comments was “Do these questions apply to a novice or an expert? (. . .) A novice will be doing some of these things slowly and explicitly, an expert will be doing them quickly and intuitively”. The KSA graph is supposed to cover the KSAs in a broad sense, without restricting them to any level of expertise. Naturally, there are KSAs that are more superficial (tending to happen at a first contact with the problem, such as LoA recognition) and those that are deeper (tending to happen after an initial assessment of the problem, such as LoA modeling). But each of them can be explored and manifested across

various levels of expertise, as will be discussed in the Progress Model.

Skill requirements. Another doubt that reviewers exposed was about the nature of the tasks students would be performing. “Do these questions apply to a situation where someone is given a set of LoA to work with, or do they apply to a situation where someone is creating LoA in the process of some task?”. The KSA graph is not supposed to verse over tasks, which are left for the Task Model. However, tasks are a concrete visualization of KSAs, which are very abstract descriptions by nature. So, it is understandable that reviewers wanted to know about tasks. The dependency relationship between each subskill is described by the graph (Figure 30), where each arrow indicates that the target employs the source, which does not mean requirement in a strict sense, but rather influence. That is, you may be able to perform well the target without performing well the source, but it is very likely that performing well the source will make you perform well the target. At last, tasks might approach multiple KSAs at the same time, if the student will be given the LoAs or they shall create them will depend on the purposes of the task. It is advisable to give the LoAs when the goal is to isolate a specific subskill.

Abstract and concrete. Referring to our second example – “a map (abstract) and the physical terrain it represents (concrete)” – the following comment was the only explicit disagreement with the graph that reviewers mentioned: “A map is concrete, I can hold it in my hand (like hardware). The relationship between things (terrain) and representations of things (maps) is very complicated, a big topic in philosophy, it isn’t just abstract vs concrete. I can have a concrete representation (e.g. a statue) of a thing that is abstract (e.g. a dragon)”. We agree that this is far more complex than the dichotomy of abstract–concrete and recall that abstraction is a soft concept really problematic to thoroughly describe and approach as an objective construct. However, this work is an effort to revisit the core ideas underlying abstraction in a way we can properly approach them and facilitate their assessment. Thus, we present the following argument to justify why we considered “map” as abstract. It is also important to highlight that we are not considering a binary classification, but a spectrum where the map was presented in opposition of the physical terrain because it is merely MORE abstract.

Software/hardware analogy. When one says a map is concrete, that they can hold it in their hands, we can assume that by “map” the person thought of a drawn/painted/printed piece of paper. But what if it is a digital map I access through my smartphone? By the hardware/software logic, the phone is concrete, the map is not. The map is a dataset that, once interpreted by specific software, is rendered on the screen as an understandable representation of a place/space highlighting regions or points of interest. I would argue that this abstraction can be "embedded" into a paper via painting, but the painted paper itself is not the map. Following the hardware/software analogy, the paper is just a screen, the painting is just a pixel array that

goes to the screen as output of the software execution (the painting process) and the map is the data the software (painter) interprets to visually represent. Surely this is too technical and deep into semantics, so we don't make the distinction in our daily lives. But ultimately, my argument is that, if I show you two painted papers (same materials, height, width and physical properties) with representations of different places and ask you if they are the same map, you would answer "no". But if I show you a painted paper and a smartphone with representations of the same place and ask you again, you would tell me "yes". Therefore, "map" is not the concrete thing embedding it, but the abstract concept reached through it.

Unidiomatic. Given the feedback from the native English speaker reviewer "Is 'Dosing' the right label here? I'm not sure what you mean?", we decided to rename the second KSA, which was called "LoA Dosing". Words like "calibration", "tuning" and "balancing" seem to convey the meaning of the KSA better. It was first called "Dosing" because its literal translation in Portuguese *"Dosagem"* is commonly applied figuratively to address the weighting of abstract concepts, analogous to how you dose medicines. However, the same does not hold true in English.

Conclusion. From both, the ratings and the written feedback of the advisory board we concluded that the KSA graph was adequate. It was identified that the major flaws highlighted by the board come from the isolation of the model from the others that are supposed to integrate the instrument into the ECD framework, namely the Task and Progress models. This may hinder the independent use of the Proficiency model in other contexts, but it would still be acceptable if alternative Task and Progress models (other than ours) are provided or built to complement it. The one relevant change that the review revealed necessary was the name of the second KSA for a more idiomatic term. Additionally, for future iterations or similar developments, we suggest requesting ratings collectively (upon the graph as a whole), instead of individually (for each KSA), with the exception of the "clarity and structure" criterion.

5.3 Task Model

Where to measure. The task model specifies tasks that are presumed to elicit observables that allow inferences on KSAs. The principled design of tasks considers the connection with their target KSAs from the start, turning explicit such inferences. This effort towards validity and reliability of the tasks guides the development to target only the respective observables, considering and minimizing "noise" (ARIELI-ATTALI et al., 2019). In ECD implementations (ARIELI-ATTALI; CAYTON-HODGES, 2014; GROVER et al., 2017), the task model is often domain-specific and incorporates characteristics of the activity/experiment. Here, we tried to maximize the reusability of the models. So, we described generic observables that are not tied to GG, GS or Graming, then refined

them into the ones from our approach, which could be seen as examples for educators who may want to approach it in a different setting.

Optimal observables. Unfortunately, designing observables to infer KSAs is a process far from optimal. Observable behaviors or achievements may result from a variety of KSAs and there hardly is any way of proving a task was accomplished due to a certain KSA instead of another (including unknown ones). A priori, task design heavily relies on the experience of the designers on the subject and the plausibility of the inferences. A posteriori, their reliability can be iteratively improved through revisions based on the comparison of the results of experiments. So, we turned the inference explicit and identified possible "noise" to the assessment, i.e. alternative explanations for the achievement that could invalidate or distort the inference.

Universal noise. There are several intrinsic factors that can influence the performance of the learner during almost any kind of task. We cannot prevent them completely, but they should not be neglected:

- **Prior Knowledge and Experience**: Learners with varying levels of familiarity with similar tasks might perform differently. It is obvious that those with more experience tend to find the task easier, while others who have never done anything like it before are likely to struggle. But, this difference in performance might come simply from memory and the mechanization of the task, not from a better-developed KSA.

- **Cognitive Load**: Here we have two types of loads to consider: Internal, the complexity of the task itself could be a source of noise. If the system is too complicated or has too many layers to consider, the cognitive load might hinder the learner's performance; and External, if the context in which the task happens is convoluted and competes with many other tasks, demanding too much from the learner, they are more likely to underachieve.

- **Instruction Clarity**: How well the task instructions are communicated can impact the learner's ability to complete the task. Vague or overly complex instructions might lead to misunderstandings about what is required.

- **Learners' Profile**: Learners' differences in spatial reasoning, memory, and analytical skills can influence their ability to complete tasks. Furthermore, their personal preferences, interests, personality and habits might make them more (or less) suitable to some approaches than others, beyond interfering with ther motivation.

- **Motivation and Engagement**: The learner's interest and motivation in the task could impact their performance. Low motivation might lead to a lack of effort and attention, affecting the quality of their performances and explanations.

- **Distractions**: Environmental factors, such as literal noise, interruptions or uncomfortable settings, can distract the learner and reduce their ability to focus on the task. That also includes pleasant distractions, such as fun environments, technological devices and toys with several other attractions disputing the learners' attention.

- **Task Fatigue**: If the task is too lengthy or tedious, or the learner has been in a row of demanding tasks for too long, fatigue might set in, affecting their ability to maintain high performance throughout the task.

- **Cultural Differences**: Culture can influence how learners perceive and prioritize information, potentially leading to different approaches in how they comprehend the task and organize themselves to solve it. The cultural background will define which references to popular culture (characters, animations, music, lullabies, shows, sayings) and daily life examples each learner has. Missing such references (or mismatching them) can compromise their understanding of the task.

- **Cheating**: There is also the possibility that the learners exploit something or get any kind of forbidden help, completing the task in ways they were not supposed to. Note that some tasks are easier to cheat out than others. In general, the more objective and based on the final product/answer the assessment of the task is, the more prone to cheating.

- **Randomness**: Most tasks involve some kind of decision-making at some point, which opens the possibility of non-informed, lucky guesses. Others explicitly rely on random generations for replayability, shuffling, dice throws and coin tosses. It is important to assess how the possible outcomes of those random events can influence the difficulty of the task.

Presentation of the model. We organized the task model in Tables 3–6, where each row refers to a subskill from the KSA graph (Figure 30). The first column identifies the target KSA, the second describes the task, the third lists the elicited observables and the inferences upon what should be observed, and the fourth identifies eventual noise interfering with the assessment. Those contain generic tasks with a greater potential of reusability in other contexts, but their generalness keeps them vague and abstract. In this regard, we refer interested readers to more concrete examples in the Presentation Model, where this Task Model was instantiated through the graming approach (Table 7).

Table 3 – LoA Recognition Task Model.

Target KSA	**Tasks** What should be asked from the student?	**Elicited observables and explicit inferences** What should be observed? And what does that mean?	**Possible noise** What else could explain it?
Distinguish between abstract and concrete	Provide a set of various descriptions, images, and scenarios. Ask the learner to: **Classify each item**: Determine whether each item is more abstract or more concrete. **Group items**: Organize items into groups based on their level of detail (abstractness vs. concreteness). **Justify classifications**: Explain the reasoning behind their classifications and groupings.	**Groupings**: When items are grouped/classified, observe the groups they are put into. How similar the groups are to their level of abstraction or concreteness shall indicate the ability to make such distinction and use it as criterion for the classification. **Representational Levels**: The identification of more than two coherent groups shall indicate a refined ability to distinguish abstraction/concreteness beyond the dichotomy. **Gray Zone**: When there is hesitation or second thoughts on the classifications, observe the items causing them. Hesitation should be proportional to how difficult/unclear to assess their abstraction/concreteness, which shall indicate if this is the underlying criterion. Hesitating on a clearly abstract item while not on an edgy item is not a good sign. **Conceptual Understanding**: When the classifications are justified, observe the reasons. The mention of amount of details or proximity to ideas/real world shall indicate they are grasping the concept of abstract-concrete differentiation.	**Classification Bias**: If the items share other features, like colors or shapes, the criterion used to put them together could be one of those, instead of the abstraction/concreteness distinction.
Connect multiple representations	Provide a set of different representations, being some of the same concept or object. Ask the learner to: **Link Representations**: Explicitly link each representation to the others of the same underlying concept. **Identify Commonality**: Recognize what is the concept underlying each set of linked representations. **Justify Links**: Explain how and why each representation relates to the same concept.	**Recognition of Multiple Representations**: When links are drawn, observe the items they connect. How far from representing the same underlying concept shall indicate the ability to perceive them as different representations of one, even if the concept itself is not clear. **Ambiguity**: An item might represent more than one underlying concept, which would allow it to be linked to a separate set of linked items. Observe learners' reaction (or negligence) to such cases. This shall indicate a refined ability to see a connection between representations. **Underlying Concepts**: When links are justified, observe the reasons. The mention of the concept they represent shall indicate they know there are multiple ways to represent it and they can spot the connection to the source.	**Linking Bias**: If the items share other features, like nature or theme, the criterion used to connect them could be one of those, instead of the reference to the same underlying concept.
Visualize in layers of abstraction	Provide an unorganized system with items represented at various levels of abstraction. Ask the learner to: **Stratify Representations**: Establish a hierarchy of different views of the system, organizing the representations into different layers, from most abstract to most concrete. This will involve distinguishing abstraction levels and connecting corresponding representations across layers. **Explain Utility**: Explain how understanding these layers of abstraction helps in dealing with the complexity of the system.	**Distinction and Connection**: Observables and inferences above also apply here. **Coherence**: When the system is stratified, observe the consistency of the layers. The distribution of representations across the layers shall indicate they are visualizing the system as a whole, not just individually classifying and connecting. **Strategy**: While the items are being put into the layers, observe the order. Different strategies can lead to correct solutions, but well-defined ones, such as breadth-first or depth-first shall indicate a clear understanding of what is being done. **Abstract Hierarchical Reasoning**: When the utility is explained, observe if some keywords are used. Terms such as simplification, dealing with complexity, organization, cleaning and visualization shall indicate acknowledgment of purposes and advantages of visualizing systems in hierarchies of abstraction.	**Quality of the Means to Provide the Solution**: This is a universal noise, but this KSA might be considerably harder (especially for novices) without proper support. On one extreme, doing this purely in our minds can easily be overwhelming. On the other hand, fully automated tools could do it entirely for us.
Acknowledge information hiding	Provide a complex system with representations at various layers of abstraction, a problem in a given layer and the same problem again in a lower layer with more information. Ask the learner to: **Spot the Differences**: Explain what changes in the problem when approaching them in different layers. **Identify Information Loss**: Recognize and describe the information that is hidden at higher layers. **Explain Implications**: Discuss what could happen if information from lower layers are neglected because they are hidden in higher layers.	**Attitude Towards Higher Layers**: When the problem is presented in a layer higher than the ideal to solve it, observe the reaction. Hesitation or confusion shall indicate intuition towards lack of necessary information. **Attitude Towards Lower Layers**: Anxious or spontaneous initiatives to use lower layers shall indicate awareness that the necessary information could still be found somewhere, it was just lost/hidden by the abstraction. **Comparison**: When the differences are explained, observe what is spotted. Bringing up the density of details while noticing they still refer to the same problem shall indicate understanding that details in lower layers may disappear when seen in higher layers. **Declaration of Lost Information**: When the information loss is declared, observe what is identified. The presence/absence of the information in the higher layer shall indicate grasping the concept of information loss/visibility.	**Pattern Recognition** If the access to different layers is simultaneous and facilitated, learners might complete most of the task by comparing patterns alone, without considering both sides in the comparison as parts of a system layered by abstraction, i.e. without engaging with the concept of information loss.

Table 4 – LoA Calibration Task Model.

Target KSA	**Tasks** What should be asked from the student?	**Elicited observables and explicit inferences** What should be observed? And what does that mean?	**Possible noise** What else could explain it?
Assess relevance of details	Provide a complex problem or system with various details and parts. Ask the learner to: **Evaluate Details**: Assess the necessity and importance of different details and parts for achieving a specific purpose. **Decide on Inclusion**: Decide which details or parts can be omitted and which must be included. **Measure Impact**: Justify the selection of relevant details, analyzing and measuring the impact of including or excluding specific details on the overall solution.	**Underestimations and Overestimations**: When details are selected, observe which are included and omitted. The amount of omitted details that are critical to solving the problem and included unnecessary details shall indicate inability to assess relevance. **Prioritization**: While details are being selected, observe the order. Selecting from the most relevant to the least is an efficient approach that requires an advanced ability to assess relevance. The order of selection might indicate if this strategy was adopted. **Impact Analysis**: When learners are asked why they included/omitted each detail, observe the answers. The justifications shall indicate if their selection criteria were indeed versing about the relevance.	**Problem Interpretation**: Also a universal noise, but here it occupies a critical role since the relevance of details for a given purpose is directly dependent of how such purpose is interpreted. One could correctly assess the relevance of the details, but for a purpose that is not the intended one.
Navigate layers of abstraction	Provide a complex system or concept with representations at various levels of abstraction. Ask the learner to: **Solve a Problem from a Different LoA**: The problem is presented in a LoA where it can't be solved. **Change LoA**: Enter another LoA, acknowledging it is a different view of the same system, not an entirely new one. **Switch Between LoA**: Navigate back and forth while maintaining consistency in understanding what is being represented.	**Reaction to the Task**: When the problem is presented, observe the reaction. Confusion, questioning the current LoA or asking permission to change it shall indicate awareness of the navigability of the system. **Post-change Disorientation**: When a LoA is changed, observe the learner's familiarity. Disorientation or estrangement shall indicate inability to switch LoAs without losing track of the represented object. **Consults to other LoA**: While the problem is being solved, observe additional LoA switches. Purposely changing LoA shall indicate confidence in navigating without losing elements or getting lost.	**Drifting**: The environment may turn changing LoA into an easy, attractive task. Purposeless navigation may happen due to visual feedback (something engaging happens when the LoA is changed) or mechanization (e.g. pressing the only button the learner knows that does something).
Evaluate layers of abstraction	Provide a complex problem and multiple representations of the system at various LoA and a specific problem. Ask the learner to: **Locate the Requirements**: Point out which elements are necessary to solve the problem. **Check Solvability**: Answer whether a given problem can be solved at a certain LoA or not. **Identify Limitations**: Spotting other problems that could not be solved at the LoA. **Make a List of Pros and Cons**: Defining what are the costs and benefits of operating at the LoA.	**Requirement as Criterion**: When a problem is said solvable in a given LoA, observe the elements previously identified as requirements. Selecting the LoA with the most requirements or the uppermost (most abstract) LoA containing at least one requirement shall indicate, respectively, acknowledgment of the connection between required elements and solvability, and understanding of the hierarchy of abstraction. **Costs and Benefits**: When pros and cons are being listed for a given LoA, observe the items. Listing advantages like agility, simplicity and readability on higher LoA and robustness, richness and depth on lower LoA shall indicate a good grasp of the benefits of LoA. As well as listing disadvantages like information hiding/loss on higher LoA and complexity on lower LoA shall indicate a good grasp of the costs.	**Requirement Analysis**: If learners fail to understand the requirements of the problem, they are likely to fail the task. But that does not necessarily mean they underperform at evaluating LoA for a given set of requirements, just that they underperform at identifying them.
Select the best layer of abstraction	Provide a system and multiple representations at various LoA and a specific problem. Ask the learner to: **Compare LoAs**: Compare the suitability of different LoAs for solving the problem. Consider the costs and benefits of each LoA in the context of the problem. **Choose One**: Select one of the LoA to solve the problem. Justify and explain how the choice was made.	**Comparative Analysis**: When LoA suitabilities are compared, observe which characteristics of the problem and of each LoA used. Including irrelevant characteristics (or not including relevant ones) in the comparison shall indicate unawareness of the constraints and requirements the problem imposes. **Justification**: When the LoA is chosen as the best fit, observe the explanation. Clear and coherent reasons shall indicate informed decision-making.	**Articulation**: Typically, the number of LoA in a system will be low. This hinders the choice of the best prone to random hits. Assessing if this was not just a lucky guess relies on the learner's ability to justify their choice. Some may fail at doing so merely because they can't express themselves well.

Table 5 – LoA Interaction Task Model.

Target KSA	**Tasks** What should be asked from the student?	**Elicited observables and explicit inferences** What should be observed? And what does that mean?	**Possible noise** What else could explain it?
Cross layers of abstraction	Provide a complex system including representations across various LoA and a present a problem featuring elements from different LoA. Ask the learner to: **Identify the LoA**: Identify to which LoA the problem belongs. **Work Simultaneously**: Develop a solution using constructs from different LoAs, ensuring coherence and integration. **Manage Detail Levels**: Adjust the level of detail for specific parts as needed, maintaining consistency and avoiding loss of critical information.	**LoA Identification**: When the LoA of the problem is being identified, observe which one(s) are picked. While picking the lowest LoA needed for the problem shall indicate understanding hierarchical abstraction, pointing or questioning about the other(s) involved shall indicate capability to consider multiple LoA simultaneously. **Detail Management**: While the solution is being built or the problem analyzed, observe how the learner navigates the LoA. Switching LoA with clear goals shall indicate ability operate on appropriate LoA for specific parts of the system without losing track of the whole. **Integration**: When the solution is presented, observe how elements from different LoA were integrated. Coherent use of elements obeying their respective LoA relationships and limitations shall indicate ability to keep track of LoA even when they are blended together.	**Discreet LoA**: If the LoA of the system used are not very well defined, or not important at all, it is possible that the learner will be able to perform very well. Not because they are able to cross LoA, but because there is not much to do in order to cross the LoA of the problem.
Identify how they are connected	Provide a complex system or concept with representations at different LoA. Ask the learner to: **Identify Connections**: Identify how constructs from different LoA are connected or related to each other. **Suitability Assessment**: Assess what characteristics or properties make a set of constructs suitable to be represented by another in a different LoA. **Explain Representation**: Explain why constructs from a LoA can accurately represent other(s) in another LoA.	**Connection Recognition**: When connections are identified, observe the reasoning that is presented. Intuitive answers stating they are the same thing or that one is a class/instance of the other shall indicate ability to recognize common connections between different LoA. **Representation Requirements**: When the learner is explaining what makes a construct suitable to be represented by another, observe the generability. Listing generic features beyond the obvious specific ones shall indicate a grasp on what makes any set of constructs suitable for representation in another LoA, rather than just their direct link.	**Loose Representations**: If abstract constructs are vague, they may open up multiple interpretations to what is required from concrete ones to be represented by them. An alternative interpretation may be mistaken as lack of ability to indentify connections.
Analyze impact on others	Provide a complex system or concept with representations at different LoA. Ask the learner to: **Check Dependence**: Identify if specific changes in elements at one LoA compromise or affect elements at another LoA. **Foresee Consequences**: Predict how creating, deleting or modifying elements within a LoA affects others. **Identify Influence**: Explain how/why content from one LoA influences content in another LoA.	**Dependence Assessment**: When the learner answers whether each change will affect other LoA or not, observe the answers. The amount of right answers shall indicate the ability to identify dependency between elements of different LoA. **Systems Thinking**: When the consequences are predicted, observe the impact described. Predictions affecting multiple LoA or upon different parts of the system shall indicate ability to consider the interconnectedness and dependencies across various LoA. **Influence Identification**: When influence is being explained, observe the connection with the spotted consequences. Translating specific consequences into generic impacts shall indicate a clearer visualization of the influence of a part of the system upon the rest.	**Previous Knowledge**: If the system being analyzed is easily linked to something previously known, the dependencies, consequences and influences declared by the learner might come directly from their reference, rather than the analysis of the system.
Replace layers of abstraction	Provide a complex system or concept with representations at different LoA. Ask the learner to: **Identify Replaceability**: Identify if the contents of an entire LoA could be replaced without compromising the rest of the solution. **Evaluate Candidates**: Evaluate multiple candidate LoA and determine which ones are suited for replacing the one in question. **Propose Candidates**: Present entirely new content for a LoA to substitute the one being replaced. **Reintegrate**: Map the content of the new LoA into the other constructs of the unaltered LoA.	**Replaceability Assessment**: When the learner is asked if an entire LoA could be replaced, observe reasoning. Seeing room for replacement shall indicate implicit abstraction and generalization, that they considered the LoA as a black box that could contain something else. **Candidate Evaluation**: When the content of candidate LoA is proposed or evaluated, observe if they could replace the previous. False positives/negatives shall indicate poor ability to identify the connection between LoA, or to apply such ability. See second row. **Reintegration**: When the content is replaced, observe if it fits the rest of the system. This shall indicate ability to connect representations.	**Fundamental Skills**: Replacing LoA is much more an application of other subskills together with a goal than a subskill isolated on its own. However, performances on them should not be confused with the performance on how they are directed.

Table 6 – LoA Modeling Task Model.

Target KSA	**Tasks** What should be asked from the student?	**Elicited observables and explicit inferences** What should be observed? And what does that mean?	**Possible noise** What else could explain it?
Simplify removing unnecessary details	Provide a complex system or concept with existing constructs and a problem. Ask the learner to: **Identify Important Features**: Identify and extract important features from existing constructs. **Create New Construct**: Use the extracted features to devise a new construct that is still able to solve the problem. **Justify Design Choices**: Justify the selection of features and decisions on what to include or omit in the new construct based on their relevance to the context.	**Feature Identification**: When important features are being identified, observe the assessment of the relevance of details. See Calibration observables and inferences, row one. **Construct Creation**: When a new construct is being devised, observe the features being used. Constructs with features different than those extracted shall indicate inability to create new constructs from a subset of features from more complex ones. **Simplicity Comprehension**: When the design choices are being justified, observe the sensibility to the problem. While reasons that take the problem into account shall indicate awareness of the purposes of simplification, reasons that do not shall indicate superficial notions of simplicity.	**Pre-established Simplifications**: If the construct represents something with well-known classes or within popular hierarchies, the learner might perform well by association rather than ability to simplify.
Generalize recognizing patterns	Provide a variety of concrete constructs with shared features. Ask the learner to: **Recognize Patterns**: Identify common features among multiple concrete constructs. **Define Constants and Variables**: Determine which features should be constant across contexts and which can vary. **Propose Generic Construct**: Propose a new, more abstract construct that maintains the identified essential features and can be instantiated in different contexts represented by the concrete constructs.	**Patterns**: When common features are being identified, observe the strategies. While selecting a feature and checking if each construct share it shall indicate pattern matching capabilities, confronting multiple constructs at the same time to spot commonalities shall indicate pattern recognition capabilities. **Constants and Variables**: When features are being individually assessed, observe which are defined as constants and which are variables. Important common features as constants shall indicate ability to employ simplification. While specific features as variables shall indicate ability to predict possible instances and variations of the generic construct. **Generic Construct**: When the generic construct is presented, observe its features. Novel, abstract constructs that encapsulate essential features while allowing for instantiation in diverse contexts shall indicate the application of abstraction to model generalizations.	**Pre-established Generalizations**: If the constructs represent instances of well-known classes or within popular hierarchies, the learner might perform well by association rather than ability to generalize.
Refine adding necessary details	Provide a conceptual framework or system with existing constructs. Ask the learner to: **Identify Relevant Information**: Identify areas where additional relevant information can enhance the existing constructs. **Detailing Process**: Introduce new details and complexities to iteratively refine and elaborate the constructs. **Demonstrate Application**: Apply the detailed constructs to solve specific problems or address practical scenarios.	**Information Gap Identification**: When gaps are being identified, observe the assessment of the relevance of details. See Calibration observables and inferences, row one. **Construct Creation**: When new construct is being devised, observe the features being used. Constructs without new features shall indicate inability to create new constructs adding features to simpler ones. Constructs with conflicting features shall indicate inability to connect representations. **Refinement Comprehension**: When the additional details are being applied, observe the suitability to the problem. While applications that require the use of the new details shall indicate awareness of the purposes of refinement, applications that do not require (simply may use them) shall indicate superficial notions of refinement.	**Pre-established Refinements**: If the constructs represent something with well-known instances or more detailed descriptions, the learner might perform well by association rather than ability to refine.
Decompose dealing with complexity	Provide a complex concept. Ask the learner to: **Complexity Assessment**: Identify if the concept is too complex for a given task. Deciding if it should be simplified going up in abstraction or decomposed going down. **Decompose Construct**: Break down the construct into smaller, more manageable parts. **Evaluate Benefits**: Evaluate the benefits of decomposition in terms of reusability, ease of treatment, and potential efficiency gains.	**Complexity Recognition**: When the complexity of concepts is being assessed, observe the decisions and their reasons. Opting for decomposition because of the need for a more concrete solution, ease of individual parts' treatment or due to reusability of parts shall indicate awareness of the benefits of decomposition to treat complexity. **Division**: When the construct is being broken down, observe the parts. Smaller, simpler, more manageable parts that sum up the original construct shall indicate application of abstraction to model decomposition.	**Pre-established Decompositions**: If the constructs represent something well-known for being composite or a group of inner parts, the learner might perform well by association rather than ability to decompose.

LoA Recognition Tasks. The first set of tasks (Table 3), targeting LoA Recognition, revolve around introducing different representations from different levels of detail. Learners will be asked to distinguish them using abstraction as a criterion, connect corresponding representations, organize all that in layers and think about the implications of such organization. As discussed, the KSA itself is unobservable, we cannot look into the learner's mind to verify if they are able to visualize LoA. So, the KSA of LoA visualization is approached through the task of organizing a system in LoA. This assumes their organization is the expression of their visualization, which may not be entirely true, but must provide a good enough estimation.

Dependency. We will briefly get back to the discussion of the dependency of KSA on each other here because the Task Model reveals those relations more concretely. For instance, LoA visualization involves other KSAs (distinguishing and connecting). It depends on the product of these other KSAs, but those could be given, isolating the task. If the learner received the system with the representations already connected and distinguished by abstraction, there would still be work to do, they must consistently distribute and preserve the relations of the items across the layers. On the other hand, items can be distributed and have their relations preserved even when ill-distinguished and connected. Thus, the visualization KSA does not depend on distinguishing and connecting, but bad performances on them could cause confusion and make it harder to perform on the visualization.

Consistency between layers. That happens because the KSAs of distinction between abstract and concrete, and the connection of multiple representations both have local scopes, while the visualization requires fitting them into a layered system. Consider A, B and C items representing one concept, while b and c represent another. In terms of abstraction A>B=b>C=c, but they are put in the layers as follows: 1 A, b; 2 B, c; and 3 C. In this case the distinction and connection may be individually correct, but not collectively so. The items were not correctly distributed into the layers within the system.

LoA Calibration Tasks. The second set of tasks (Table 4), targeting LoA Calibration, revolve around positioning oneself in relation to the level of abstraction. Learners will be asked to decide which details to keep and which to hide, navigate through the LoA, evaluating, comparing and selecting them. It is not about taking whatever actions are needed to change LoA views in a given environment, it is about knowing how to analyze and decide in which level of abstraction one should work in order to solve a given problem.

Make explicit the implicit. The KSA of LoA Calibration is the one we expect to have the most transferability as a general-purpose skill. Almost any problem in any field could benefit from a good calibration of abstraction, but this KSA is also expected to happen implicitly most of the times. So, the challenge of designing tasks for it, is

to make the learner explicitly show this calibration process. In this regard, most tasks here require justifications and open-ended answers that are more subjective.

LoA Interaction Tasks. The third set of tasks (Table 5), targeting LoA Interaction, revolve around analyzing and managing the structure of the LoA and its interconnections. Learners will be asked to consider more than a single LoA simultaneously, understand how they connect to each other, the consequences of one onto the other(s) and even replacing LoA. The tasks often require systems thinking, to always keep track of the layered structure and the connections between representations at different levels of detail.

LoA Modeling Tasks. The fourth set of tasks (Table 6), targeting LoA Modeling, revolve around actively contributing to the system, devising new constructs through abstraction. Learners will be asked to simplify unnecessary complexity, generalize groups of specific constructs once patterns are recognized, refine insufficient plainness and decompose impracticalities. The tasks always demand an initial phase of identifying the opportunity to employ one of the abstraction modeling skills. That is, to evaluate if it is too simple/specific/limited or too complex/vague/intractable, and whether to go up or down abstraction to treat it.

LoA counter-intuition. Although intuitive, it is not true that we must always go up in abstraction to deal with complexity, nor always go down to enhance feasibility, bringing it closer to the real world. When you face an overcomplicated problem, you may want to employ decomposition bringing the abstraction down, or simplification, bringing it up. When you face a very limited problem, you may want to employ generalization bringing the abstraction up, or refinement, bringing it down. The ability to identify when to use one or another should be stimulated by the tasks.

Table 7 – Task Model via Graming.

Target KSA	Grame	Tasks
Distinguish between abstract and concrete	**Land of Abstraction – Presentation**: The learner is introduced to a grame with 9 characters (high LoA), which are either superheroes, princesses or pets (mid LoA): the green, red and blue superheroes; the princesses of the swamp, snow and sea; and a dog, a fish and a lizard (low LoA). **More/Less Abstract:** One of those representations randomly appears for the learner. **Most/Least Abstract:** One representation of each LoA is randomly put together and presented to the learner.	**Follow** the presentation of the characters in each LoA. **Compare** and classify the current representation as more or less abstract than the previous. **Identify** the most and least abstract representation.
Connect multiple representations	**Land of Abstraction – Match Pairs**: Two columns are presented to the learner: one with the representations of superheroes, princesses and pets (mid LoA); and the other with the green, red and blue superheroes, the princesses of the swamp, snow and sea, a dog, a fish and a lizard (low LoA).	**Link**/draw a line from the concrete representations (low LoA) to their respective abstract representations (mid LoA).
Visualize in layers of abstraction	**Land of Abstraction – Scramble Lvl 1**: The learner must choose one of the nine concrete characters (low LoA). Then the LoA is raised (to mid) and they are slowly shuffled. **Land of Abstraction – Scramble Lvl 2**: Same, but the LoA is raised to the highest.	**Choose** the abstract (mid LoA) character corresponding to the concrete one chosen at the beginning. **Choose** the abstract (high LoA) character.
Acknowledge information hiding	**Land of Abstraction – Pet Feeding**: The learner is presented three identical actions (rules) on mid LoA: a pet eats a food. However, at the lower LoA each kind of pet eats a certain kind of food, which will cause failure if there is a mismatch of pets/food. The source of error is invisible at the mid LoA.	**Explain** why the actions (rule applications) could be failing on mid LoA. **Feed** all three pets taking the right actions on low LoA.
Assess relevance of details	**Virtual Pet – Presentation**: The learner is presented to a simple game where they have to take care of a virtual pet: feed it when it is hungry; clean it when it is dirty; and play with it when it is sad.	**Satisfy** all pet's needs. **Reason** about how much is necessary to know about feeding, cleaning and playing with pets to play the game.
Navigate layers of abstraction	**Land of Abstraction – Dancing Pairs**: All characters receive an invitation letter giving each specific character the right to bring a companion of a certain role: superhero, princess or pet. If a character invites another, the latter does not need to use their own invitation. One character will be left alone, he/she will be the one singing.	**Switch** between LoA to check character roles and specific characters. **Match** four pairs according to the invitation letters.
Evaluate layers of abstraction	**Land of Abstraction – Queen Arrival**: The queen is arriving to make an important announcement, but she demands organization. The learner is told to prepare the characters for it. They are free to choose in which LoA they will work. **Land of Abstraction – Dress Code**: The queen announces a party at the castle, but she demands a dress code. The learner is told to prepare the characters for buying clothes. They are free to choose in which LoA they will work.	**Sort by role** all characters, that is, make three groups: superheroes, princesses and pets. **Justify** the LoA. **Sort by color** all characters, that is, make three groups: red, green and blue. **Justify** LoA chosen for that.
Select the best layer of abstraction	**Land of Abstraction – Pet Return**: After the party, the learner is asked to return all three pets to home through a tiled path repeatedly using actions of their choice: a generic one that moves any pet a tile; one that moves the dog a tile; one that moves the fish a tile; and one that moves the lizard a tile.	**Choose** the actions to take (any combination). **Arrive** at the top tiles with each pet. **Justify** the chosen actions and the LoA.
Cross layers of abstraction	**Land of Abstraction – Invitation Letters**: All characters receive an invitation letter giving each specific character the right to bring a companion of a certain role: superhero, princess or pet. If a character invites another, the latter does not need to use their own invitation. One character will be left alone, he/she will be the one singing.	**Explain** the meaning of each invitation. **Acknowledge** representations from different LoA are together in each of them.
Identify how they are connected	**Virtual Pet – Alternative Play**: In a modeling game, processes abstracted as simple actions in the virtual pet game are described as 5-step processes. After modeling a 5-step process to describe how to play with the dog, an alternative 5-step must be given.	**Model** 5 steps to play with the pet, different from another. **Explain** what makes both 5 steps able to be represented by the simple action.
Analyze the impact	**Virtual Pet – 5-step Management**: In a modeling game, processes abstracted as simple actions in the virtual pet game are described as 5-step processes. During the modeling of the steps, they can be deleted, created or replaced.	**Explain** how the changes in the steps affect the game or other processes previously described.
Replace layers of abstraction	**Virtual Pet – A new Pet**: Considering 5-step processes refining all simple actions in the virtual pet game, the pet now should be replaced for another. It still needs to be fed when hungry, cleaned when dirty and played with when sad. However, how you feed, clean or play may drastically change, which would impact the 5-step processes.	**Propose** another pet to take its place. **Spot** what stays the same and what must be changed. **Identify** how the replaced content relates to the unchanged.

Target KSA	Grame	Tasks
Simplify removing unnecessary details	**Virtual Pet – Endless Instructions**: After the virtual pet is taken care of, how you actually feed a dog is thoroughly explained as a gigantic series of several detailed steps.	**Model** a basic 5-step process to explain it, summarizing the endless series.
Generalize recognizing patterns	**Virtual Pet – Generic Instructions**: After another pet replaces the dog and other 5-step are necessary to describe the actions specifically for that pet, the challenge is to merge them into a single one again, suitable for both pets.	**Identify** patterns linking the existing processes. **Model** generic 5-step processes to replace the both specific ones previously designed.
Refine adding necessary details	**Virtual Pet - Everything in 5 Steps**: The simple actions from the virtual pet game (feed, clean and play) need to be explained to a friend who wants to take care of a real pet.	**Model** 5-step processes to detail the actions of cleaning and playing.
Decompose dealing with complexity	**Virtual Pet – Long Steps**: When the actions from the virtual pet game (feed, clean and play) are explained in 5 steps, a fairly long and composite step might be intuitively modeled.	**Identify** a step is much bigger than others. **Model** smaller steps by breaking down the big one.

Validation Method

Proof of Concept. In order to validate the Task Model we conducted a small experiment applying the specific tasks instantiating it via the graming approach (Table 7). The experiment consisted of a cognitive interview along all tasks as an effort to identify the reasoning behind the learners' actions during the tasks. As described in depth in the Presentation Model (Section 5.5), the tasks are approached through an educational game running on GrameStation. We interviewed four children of ages 8, 10, 11 and 12, being three girls and a boy, from different cities in the Rio Grande do Sul state of Brazil: Pelotas, Dom Pedrito and Itaqui. The sessions were after-school, in-person, individual and lasting between 30 and 40 minutes each. The activity and the use of data for research were also explained in-person to the legal guardians of the participants, who signed the IC form (Annex A, in Portuguese) and the term of rights regarding image and voice (Annex B, In Portuguese). This was necessary because we recorded the interviews to transcribe and analyze them in depth.

Think Aloud with extra support. We employed the Think Aloud protocol (ERICSSON; SIMON, 1998) since we wanted to hear about the reasoning of their learners during the activity. However, as the subjects were young and often were silent, we had to interfere, inciting them to express their reasoning more explicitly. Sometimes the students weren't confident enough to finish their sentences. After giving them the opportunity to conclude by themselves for a while, we offered scaffolds to their answers, boosting their confidence. Beyond that, the interviewer stated and explained all tasks to the students as they progressed into the activities. We also harnessed the opportunity to throw some questions regarding possible noise, in an attempt to verify if unwanted reasons could have influenced the students' performances.

Post-processing. The interviews were initially transcribed using a private speech-to-text AI tool, then thoroughly undergoing three rounds of revision: AI correction, listening to the recordings and verifying if the AI-generated translation was correct; anonymity protection, the whole text was screened and all personal references were omitted; translation, after an automated translation from Portuguese to English, the text was reviewed again to fix eventual translation issues. The resulting transcriptions are attached as Appendix B–E. Consult them (from where the following quotes were extracted) and Table 7 (describing the games and tasks) if you need more information while reading the discussions below.

First Contact. The activity begins with a small explanation of the concept of abstraction, its LoA and how they are managed in GrameStation: the abstractometer. All students demonstrated a lack of previous knowledge of what is abstraction. However, they have also shown curiosity towards the word. There was engagement as soon as the name of the game was spoken: "What is abstraction?", a learner asked interrupting the introduction. Similar reactions were observed in all students. This is expected,

since “abstraction” is not a usual word, particularly for children. That is an important discussion, since abstraction is a quintessential concept of CT and yet, children often have never heard of it.

Explaining abstraction. The explanation given for all students were variations of drawing a cat, like the following:

> “Abstraction is basically when we think about things without worrying too much about the details. We only focus on the things that matter. Like, I don’t know, when you’re going to draw a cat or a dog, you only draw the things that matter about that dog, draw its ears, its snout, its eyes... You don’t draw every single detail. Like, you’re not going to draw every single hair on that dog in the drawing. So, when you’re drawing, you’re abstracting. The more you abstract, the more you stay in the world of ideas, things that require no existence. You’re only focusing on the things that matter. The less you abstract, the closer you get to the real world, to the concrete things that exist here, with more details”

Feedback on distinguish tasks. The first tasks, to classify random representations as more or less abstract, as well as to pick the most and the least, seemed very basic and intuitive for the students. They were able to distinguish the level of abstraction of each representation with ease. However, from the start to their final remarks they have eventually mixed it up, inverting what is more abstract with what is less abstract and vice-versa. For instance, when the interviewer scaffolded “So, when we are in a high, very abstract layer, we...”, the student answered “There is a lot of detail”. It seemed to be a consequence of being unfamiliar with the word, rather than lack of the skill to distinguish, as can be seen in the concluding dialog of a student: “I learned that more detailed is not having... What is the name?” – “Abstraction.” – “Abstraction! And with fewer details more abstract”.

Feedback on connect tasks. The learners have also excelled in connecting multiple representations during the match pairs game. This part was generally quick and straightforward. But there was an unforeseen noise, the youngest student gave too much importance to the looks of the representation of princesses (mid LoA): “Princess? That’s a bride!”, and that made the student reluctant to connect the specific princesses (low LoA) to the class of princesses (mid LoA), since they were not brides.

Feedback on visualize tasks. The scramble game was the first task that felt like a game and that engaged the students. The idea behind the game is to make students keep track of the characters and their respective more abstract counterparts, acknowledging it’s the same system under different views. However, some noise haunted this activity and was even spotted by the students themselves: “I’m not very good at memory, but...”. Memory and spatial skills might have interfered with the performances of the students. But the underlying logic of visualizing the system as a connected, layered

structure seemed to be understood through the coherence of the position of representations from different LoA. As could be interpreted when the interviewer asked “Now I want to ask you how did you know this was the [superhero in the red uniform]?” and had the answer “Ah... It was in the same place”, or the comment of another student “Because since you started shuffling, I’ve been following you”.

Feedback on evaluate tasks. After the scramble, the activity progressed to the queen arrival and dress code minigames, where the student should choose a LoA to sort characters by role and color, respectively. It was expected that the mid LoA were chosen to sort by role, however, three students chose the lowest and weren’t able to justify the choice. When asked to justify, they answered “First I was just guessing”, plain “I don’t know”s or brought up the interpretation of the representation again “I think this one, because in this one those guys look more like a bride than a princess.”. Understanding the solvability of the problem in a higher LoA came from asking if it was possible to solve using the mid LoA, and then asking the same about the next minigame (which was not possible to solve there). The one student that chose the mid LoA for the first game justified “(...) it seems like it was easier to differentiate (with the middle layer) than with it (the lowest one)”. All of them got that the second sort couldn’t be done in the mid LoA, with reasons like “There is not much detail”. Thus, overall they seemed to struggle at first, and then evaluated LoA through some intuition, but they didn’t seem very aware of what could make a LoA suitable to solve a problem or not.

Feedback on cross tasks. Activity progressed into the invitation letters minigame, where characters received invitations allowing them to go to a party with a companion of a certain role. The invitations pairing a specific representation (low LoA) with a generic one (mid LoA) wasn’t a trouble for the students. However, there was a major misunderstanding that happened with all of them: students thought they should match both invitations of the pairs at the same time. This would be a cool logic puzzle, but the minigame wasn’t designed after that. It was much simpler, you just had to use 4 invitations, obeying their rules, the second halves of the pairs didn’t have to use their own invitations, as they were already in the party as companions of others. Another thing that bothered the students was the fact that one of the characters was left alone, since the game have an odd number of characters. Therefore, the feedback here is to correct those design flaws, make it an even number of characters and match all invitations into a logic puzzle that perfectly fits in the end.

Feedback on navigate tasks. After receiving and understanding the invitation letters, the dancing pairs minigame was to pair up the characters for the party according to the invitations. Although students were free to switch LoA during this minigame, this resource was underused. It was expected that they would keep switching to check the roles of the companions, but at this point of the game, students had already memorized the roles, in addition to them being fairly obvious. Thus, this task failed to make ground

for observables and respective estimations on LoA navigation. From this experience, the suggestion for future task designs is to present new systems, or more complex ones, where constantly navigating through the LoA is actually necessary.

Feedback on select LoA tasks. After the party, everyone goes home and the students are asked to bring the pets to their homes. They can do it using specific rules to move each pet (low LoA), a generic one to move a pet (mid LoA) or any combination of them. What is expected from this minigame is that the students realize there is no advantage in moving the pets with specific rules, and there is the cost of switching between them. While the single generic rule could move them all. So far, students seemed very confused about this task and chose to apply specific rules using the logic that they needed to move each of the pets individually. This can be seen in the following dialog, starting with the student answering how he will solve it: “I’ll go one by one, like this” – “It could be one for one, but why would you choose one for one? And what is the advantage of one for one?” – “Isn’t it so that everyone can get their food? This one, this one and this one?”. That is true, they must move individually, but not specifically, the generic rule can move each individually, without having to switch rules. One of the students chose the generic one, but couldn’t elaborate on that choice: “Because I don’t know. To move any animal”. Therefore, it is difficult to know if the task successfully suggests that students struggle with the skills necessary for good LoA selection, or if the task is flawed and doesn’t offer them the proper space to employ those.

Feedback on information hiding tasks. Once the pets arrived home, they would find food and should be fed. However, there was three rules, indentical on mid LoA, but defining a different food for each specific pet on low LoA. It was expected that they would try to feed the pets on mid LoA and fail, then they should explain the possible reasons for such failure (that there is relevant information hidden in the low LoA). There wasn’t much elaboration on this one, the interviewer’s questions and explanations were just nodded most of the times. One of the students failed on mid LoA “(starts using the rules) Oh no, wait, this one, look...” – “That’s right, but you can’t do that by looking at it at this level of abstraction, because you don’t know who’s who.” then silently changed the LoA and finished the task. Information on this task was very limited, the interviewer tried to engage answers without success and ended up explaining everything alone.

Feedback on detail relevance tasks. This was the point where we went to the next grame: Virtual Pet. After a brief explanation about the simple mechanics of attending the pet’s needs with respective rules, it was asked for students to wonder how much they knew about the actions of the game (feed, clean and play). More importantly, how much it was necessary for them to know in order to play the game. This questions though, were overshadowed by the fast pace of the game. Comments like “Oh my God, he’s hungry, he’s sleepy, he’s dirty!” and “This game is impossible!” followed a rush to complete the game, and the questions were ignored. The idea here was for

the game itself to be irrelevant, just for engagement, the real tasks were the questions. This failed. Thus, reaction time-based game should be used with caution if the reaction is not what is being measured.

Feedback on simplification tasks. After winning the game, the simple actions begin to be explained. Purposely annoying, “feed” is explained through a giant series of over-detailed steps. The point of this minigame was to trigger the realization that there is a cost associated with the density of details, which is not always worth paying. There were no verbalizations on that, but students often expressed explicit annoyance with their faces. However, when they were asked to simplify it as a 5-step process, they had to do it from memory, which caused them to summarize the first steps from the long series, but completely make up the rest. Therefore, the target KSA subskill, simplification, wasn’t well worked in this minigame. This could have been the case if the students were given the full series of over-detailed steps to consult, rather than just listened to it once.

Feedback on refinement tasks. Once “feed” was explained as 5 steps, so were “clean” and “play”, but those had the abstract, simple action as a starting point. As we have just discussed, “feed” ended up being another refinement rather than a simplification. These 5-step modeling minigames presented different challenges from student to student, but they all seemed excited about writing their own steps. Opposed to the rest of the activities where they felt like they should be looking for the “right answer”, here they felt free to express themselves as they wished, as long as it resulted in 5 steps. The answers revealed different takes on each action depending on their personal experiences with their real-life pets. For instance, while most students explained “cleaning” as they bathing the pet themselves, a student described the steps of taking the pet to the petshop and waiting for the bath. This link to concrete references from their daily lives seemed to influence a lot the steps designed. For instance, some fairly specific steps were proposed, such as “Lock him up, otherwise he’ll bite my hand” and “Call someone to open the house for me, because I don’t have a key”.

Feedback on decomposition tasks. Yet on the 5-step modeling minigame, they didn’t seem to have a plan from the start and eventually had to simplify merging two steps into one or decompose one into two so they could reach the goal of 5. That is, they didn’t calculate how abstract the steps should be in order to be a total of 5 from the start. They modeled unmeasured steps on the fly, and once they finished describing the whole process they would go back, count and adjust. This might indicate a lower calibration skill, but an iterative strategy shouldn’t be seen as an underperformance for modeling skills. This, however, brings some implications to this task. As long as this is not an explicit task, but rather a possible event within other tasks, this might fail to assess decomposition because the opportunity might never exist in some cases. Thus, a task on its own should be designed to approach decomposition.

Feedback on impact analysis tasks. Everything discussed on the previous paragraph also applies here. Although students were directly asked how the creation, deletion and modification of steps impacted the game and the other 5-step processes designed, it didn't feel like a task on its own. All the 5-step minigames felt too isolated/independent from each other and it was pretty straight-forward that there was no impact on the rest of the system. Therefore, this task didn't give us the opportunity to assess the impact analysis subskill, because there wasn't any to be analyzed in the first place. This task needs to be redesigned using a more interconnected system where an impact analysis could actually reveal something.

Feedback on LoA connection identification tasks. Once all three actions (feed, clean and play) were described as 5 steps, it was asked if another 5 steps could describe "play" in a different way and what took the steps to be able to describe a "play". The answers were all "yes", but the "why" had to be scaffolded. The interviewer asked how the dog had to be before using "play" in the virtual pet game (sad), and how it stayed after using it ("not sad", thus happy). Therefore, they were led to the realization that any 5 steps that start with a sad pet and end up with a happy one could represent "play" in a lower LoA. Once again, the students were shy on verbalizations and the interviewer had to compensate with scaffolds, thus, this is more of an assumption than a conclusion out of observation. Remarkably, one of the students included the requirements for the abstract actions as the first steps for their concrete processes, such as "Having a sad little animal" for "play". This student, however, did it from the start, for all actions. Although it wasn't a consequence of this specific minigame/task, nor captured by it, we can consider it was indeed an indicative of the subskill of identifying how LoA are connected.

Feedback on LoA replacement tasks. Finally, the last minigame was to imagine we didn't have a dog as our virtual pet, but something else, like a fish. The goal of this minigame was to trigger the realization that the high LoA should be kept the same, any pet would have needs and they have to be attended. However, the 5-step detailing how you attend the pet's needs would change. That is, we would be replacing the low LoA without changing the high. Students were reluctant to accept the high LoA would stay the same, they put that "Fish don't get dirty", you can't clean a fish "Because there's no way we can catch the fish, get a sponge to wash it!" or play with it "I don't know, how do you play with a fish?". A student said playing with a fish could kill it "But that gives fish a heart attack! (...) My mother said that they could die of fear". Thus, this task may have revealed the impact analysis and replaceability assessment of the students, but it didn't turn out the way it was expected. Maybe using another pet to replace the dog could make the students accept the high LoA staying the same.

Feedback on LoA generalization tasks. Given the distance from dog to fish and the reluctance from the students, this minigame became impracticable. They proposed

new 5-step to feed, clean and play with a fish, barely accepting one could actually do those. Since they didn't agree with those actions for a fish, there was a lot of scaffolding by the interviewer to reach the answers. We judged that this significantly biased the answers and was saying more about the interviewer than about the students. Noteworthy, that was also the end of the activity, i.e. 30–40 minutes in, the students seemed to be getting tired at this point. Thus, the minigame to generalize the pairs of 5-step (those about the dog, and those about the fish) was never reached.

Implications for the model. The experiment didn't raise concerns about the generic task model (Tables 3–6), but did raise some about our instantiation (Table 7). The following minigames had design flaws revealed: Invitation Letters; Dancing Pairs; Returning Pets Home; Virtual Pet; Endless Instructions; 5-step Management; A New Pet; Long Steps and Generic Instructions. Those flaws hindered the assessment of their target KSA subskills. Being 9 out of 16 (56.25%) minigames flawed, a new iteration of the instantiated task model is necessary. Noteworthy, Invitation Letters issues caused confusion, but didn't compromise the assessment of its target KSA, contrary to Dancing Pairs. Therefore, we can say the tasks were successful for evaluating LoA Recognition, then LoA Navigation, LoA Modeling and LoA Interaction, from most to least. The Virtual Pet game gave great insights on refinement and impact analysis, but not necessarily in the minigames/tasks designed after those.

5.4 Evidence Model

How to measure. The evidence model provides detailed instructions on how we should update our information about the proficiency model variables (the unobservable proficiency θ we mentioned there) given a performance in the form of students' work products (responses to a task from a given task model). It is composed of: tasks evidence rules, how observable variables summarize performance in a particular task, guiding a response scoring model; and measurement model, which connects the score from the observable variables to the unobservable KSA from the proficiency model, guiding a summary scoring model (MISLEVY; ALMOND; LUKAS, 2003).

Multifaceted assessment. Since our take on CT and abstraction is that they are general-purpose problem-solving skills and that we should be looking at the process, the path, not the final products, a straightforward objective assessment from final answers didn't seem appropriate. In this regard, we decided to make a composite assessment with three complementary systems: GrameScore, the traditional purely quantitative one; CompaCT Logs, a hybrid; and Design Diary, a purely qualitative one.

GrameScore. Our first assessment model is the traditional response scoring model, defining scores for each correct answer and penalties for wrong ones. Penalizing wrong answers follows the logic that mistakes reveal more about performances

than successes and it has shown to be worth it even considering it discriminates risk averse students (ESPINOSA; GARDEAZABAL, 2010). It is a common strategy to deal with guessing and can be a simple constant as we use in this work or a more complex penalty function taking into account other factors, such as the ratio of wrong answers over the total (SCHAGAEV et al., 2012). Each minigame have a total of 10 points, but given the penalties, the final score can be negative. The Table 8 lists all response scoring rules, identifying by Grame and Minigame, how to gain points (Scoring Bonus), how to lose (Scoring Penalty) and the maximum score for each minigame.

Table 8 – GrameScore for Land of Abstraction and Virtual Pet.

Grame	Minigame	Scoring Bonus	Scoring Penalty	Max.
Land of Abstraction	More/Less Abstract	+1 for each correct comparison	-0.5 for each wrong one	5
	Most/Least Abstract	+0.5 for each correct classification	-0.25 for each wrong one	5
	Match Pairs	+1 for each correct pair	-0.5 for each wrong	12
	Scramble Lvl 1	+1 for picking the right one	-0.5 for picking one of another role	1
	Scramble Lvl 2	+1 for picking the right one	-0.5 for picking one of another role	1
	Queen Arrival and Dress Code	+1 for having 3 groups +1 for each pair correctly grouped	-0.25 for each pair wrongly grouped	10
	Invitation Letters	+1 for each correctly read	-0.5 for each misunderstood	9
	Dancing Pairs	+1 for each correct pair	-0.5 for each incorrect pair	4
	Return Pets Home	+1 for each generic move	-3 for each rule switch	12
	Pet Feeding	+1 for each pet fed	-1 for each rule application fail	3
Virtual Pet	Presentation	+10 for satisfying all needs	-1 for each wrong action taken	10
	Endless Instructions Everything 5 Steps Alternative Play A New Pet	+2 for each step	-2 for each additional one (>5) -2 for each missing one (<5)	10
	Steps Management	+1 for each impact captured	-1 for each impact missed	–
	Long Steps	+1 for each long step broken down	-1 for each long step kept	–
	Generic Instructions	+2 for each step out of 2 specifics	-1 for each that does not generalize	10

Scoreboard. The scoreboard for our experiment guided by GrameScore and considering the subjects F8, F11, M12 and F10 (named after their gender and age) is depicted in Table 9. Each row represents a minigame and each column a scoring bonus (+), scoring penalty (-), total score (=), or a normalized score (%) that divides the total score by the maximum score. Each of those for each of the subjects. Overall, all students had similar scores, with totals of 64, 76, 75 and 69.7 (max 132); and normalized scores of 11.3, 12.9, 13.3 and 11.6 (max 17). Noteworthy, a single minigame (A New Pet) is responsible for 108 out of the 148.2 total penalties (72.9%). If we consider this minigame never existed, the total scores drastically improve to 82, 106, 87, and 95.7 (+30.2%) and normalized ones to 11.9, 13.9, 14 and 12.5 (+6.5%).

Feedback and scaffolding. It is important to highlight that the whole activity was monitored and interacted with the interviewer, who provided instant feedback and scaffolds whenever the students struggled. This isn't captured by the GrameScore system, but it does have a huge impact. For instance, F8's first answer inverted the logic, considering that things with more details were more abstract and those with fewer details, less abstract. This confusion was untangled by the interviewer right after the answer. It is very likely that if this instant feedback hadn't happened, the confusion would be kept for at least the whole minigame, propagating the error and drastically changing F8's

score. No answer was entirely given by the interviewer, but several times the students couldn't express themselves and got a scaffold from the interviewer beginning the answer. The scaffolds impacted much more the subjective reasonings and justifications than the objective goals that are captured by GrameScore, yet, it is worth noting that there might be some influence.

Table 9 – Scoreboard for Land of Abstraction and Virtual Pet.

Minigame	**F8**				**F11**				**M12**				**F10**			
	(+)	**(-)**	**(=)**	**(%)**	**(+)**	**(-)**	**(=)**	**(%)**	**(+)**	**(-)**	**(=)**	**(%)**	**(+)**	**(-)**	**(=)**	**(%)**
More/Less Abstract	4	.5	3.5	.7	5	0	5	1	5	0	5	1	5	0	5	1
Most/Least Abstract	5	0	5	1	5	0	5	1	5	0	5	1	5	0	5	1
Match Pairs	9	1.5	7.5	.62	12	0	12	1	12	0	12	1	11	.5	10.5	.87
Scramble Lvl 1	1	0	1	1	0	0	0	0	1	0	1	1	1	0	1	1
Scramble Lvl 2	0	0	0	0	1	0	1	1	1	0	1	1	1	0	1	1
Queen Arrival	10	0	10	1	10	0	10	1	10	0	10	1	8	.75	7.25	.72
Dress Code	10	0	10	1	10	0	10	1	10	0	10	1	10	0	10	1
Invitation Letters	7	0	7	.78	5	0	5	.56	9	0	9	1	6	0	6	.67
Dancing Pairs	4	0	4	1	4	0	4	1	4	0	4	1	4	0	4	1
Return Pets Home	4	0	4	.33	12	0	10	1	5	9	-4	-.3	11	3	8	.67
Pet Feeding	3	0	3	1	3	1	2	.67	3	0	3	1	3	4	-1	-.3
Virtual Pet	10	15	-5	-.5	10	3	7	.7	10	0	10	1	10	1	9	.9
Endless Instructions	10	0	10	1	10	0	10	1	10	0	10	1	10	0	10	1
Everything 5 Steps	10	0	10	1	10	0	10	1	10	0	10	1	10	0	10	1
Alternative Play	10	0	10	1	10	0	10	1	10	0	10	1	10	0	10	1
A New Pet	6	24	-18	-.6	0	30	-30	-1	4	26	-22	-.7	2	28	-26	-.9
Steps Management	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Long Steps	2	0	2	1	3	0	3	1	2	1	1	.33	0	0	0	0
Generic Instructions	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Total	105	41	64	11.3	110	34	76	12.9	111	36	75	13.3	107	37.2	69.7	11.6

Unfinished tasks. Some of the minigames were not completed by the students due to time pressure or fatigue. For instance, F8 and M12 struggled with the Return Pets Home minigame. Because of the lack of progress, even after scaffolds were given, the interviewer completed the minigame for them in order to ensure that the whole activity would be finished within the time constraints. The only points considered for the scoreboard were those obtained by the students while they were trying to solve the minigame. None of the actions taken by the interviewer were scored. Steps Management minigame had no impacts to be captured, as discussed on the previous section. A New Pet minigame was very briefly approached at the end of the activity. Instead of modeling 5 steps, students barely talked about possible processes, which were considered as the modeling of single steps. We consider this minigame wasn't properly approached and could be excluded from this assessment. Just like Generic Instructions, that should follow it, but was barely mentioned.

Bugs. The Pet Feeding minigame faced a technical issue with the platform, where the mapping required to apply the rule was being incorrectly classified as an error. Therefore, the points that made into the scoreboard were manually given by observation of the interviewer, rather than the data from GrameStation. Some random "X"s signalizing mistakes also happened to pop up on the screen during various minigames, especially the first ones, that were completed by dragging elements everywhere. The game engine tried to recognize serial selections as mapping attempts, while the stu-

dents were just dragging elements around. None of those “errors” scored neither positively or negatively for any minigame.

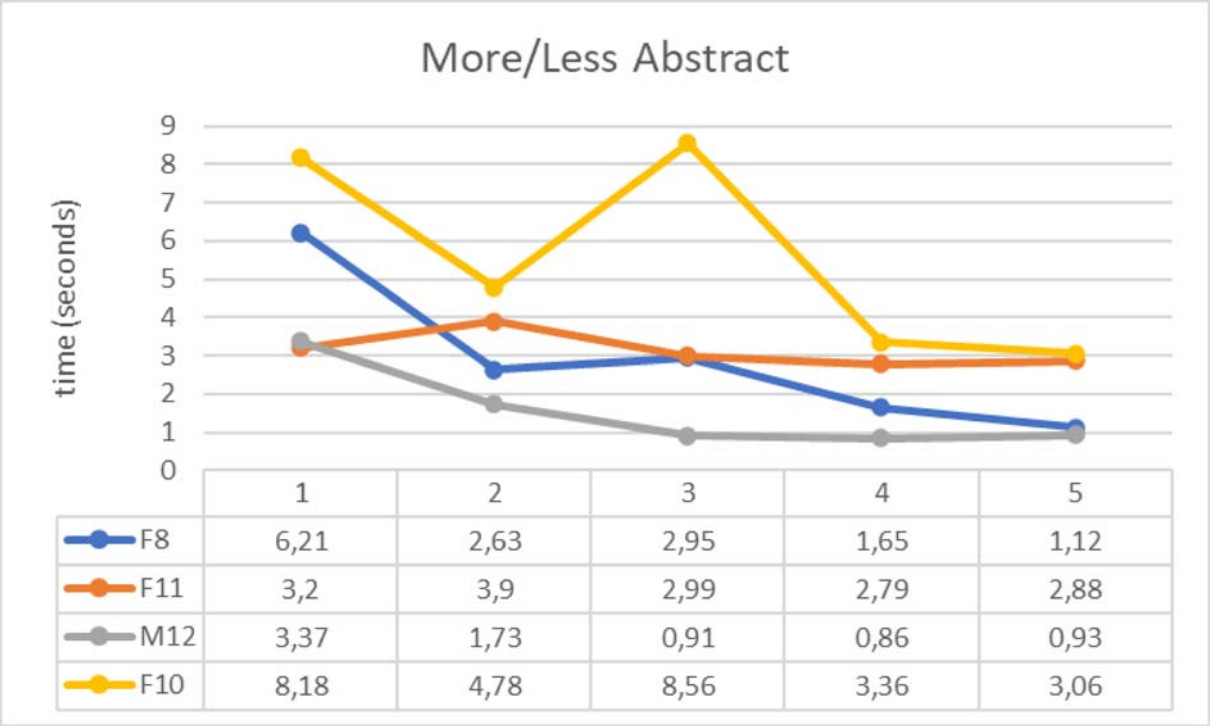
Table 10 – 5-step processes modeled by the students.

Task	F8	F11	M12	F10
Feed 1	go to the pot	Get the food from the cupboard	Have a pet	Find the food
Feed 2	lift the lid of the pot	Shake it to make noise	Buy food	get the dispenser
Feed 3	get the food	Call the pet	Wait for it to get hungry	put the food in the dispenser
Feed 4	put the food in the pot	Open it and put half in the bowl	take the food	put it from the dispenser into the bowl
Feed 5	put the pot on the floor	Wait for the pet to eat	put it in the jar	call the dog to eat
Clean 1	Get the pet and bath supplies	Find the pet	Have a dirty pet	Catch the dog
Clean 2	Turn on the tap, watering the pet.	Put the pet on a leash	Buy soap	Get the shampoo
Clean 3	Soap the pet	Go to the pet shop	Taking the pet to the bathroom	Turn on the tap or shower
Clean 4	Rinse the pet	Wait until the bath is over	Bathe the pet	Soap the dog
Clean 5	Dry the pet	Go back with the pet	Dry the pet	Rinse
Play 1	Get the toy	Find the pet	Have a sad little animal	Pick a ball
Play 2	Take the pet outside	Put the pet on a leash	Have a little ball	Call the dog
Play 3	Give one of the toys to the pet	Take it for a walk	It liked the ball	Throw the ball
Play 4	Run after it like crazy	Get back home with it	Throw it for the pet to fetch	The dog catches it
Play 5	Get the toys and the pet and go back home	Put away the leash	The pet fetches it and brings it back	call the dog with the ball to play again
Play^2 1	Get a little stick	Find the pet	Have a sad dog	Call the dog
Play^2 2	Get the pet’s attention showing it	Take it to the sofa	Take it outside	put the leash on it
Play^2 3	Throw the stick	Cuddle it	Take a ball with you	open the gate
Play^2 4	Wait for the pet to bring it	Let it turn his belly	Take two slippers and put them on the floor separately	Go for a walk with him
Play^2 5	Pick up the stick and put it away	Pet its belly	Kick the ball, the pet will be the goalkeeper	close the gate

Five steps. All minigames from Land of Abstraction have their answers easily verified by comparing them with a predefined right answer. This automation of quantitative assessment is challenging for open-ended tasks like all modeling ones from the Virtual Pet grame. What effectively scored in the GrameScore system was the number of steps. Since the whole activity was conducted by the interviewer, we knew the 5 steps were valid for the task, but that could simply not be the case. The steps should be assessed to verify if they describe the actions they should. A way for doing that is to use LLMs to analyze the steps. A lot more work is required to verify if it is safe to use them to make such assessments in general, but in our little experiment ChatGPT 3.5 was able to identify that the steps were modeling what they should. The final answers for all modeling games are reported in Table 10.

CompaCT logs. The hybrid system gathers objective information, summarize it and show them to the people in charge of the assessment. Despite being quantitative data, this information often may lead to different conclusions based on interpretation, hence the qualitative part. GrameStation collects all kinds of relevant information in its logs, such as interface requests (clicks, buttons pressed) and GG events (rule applications, modifications of GG components). Which rules were applied, in which order and

which mapping errors the user committed are usually very helpful data for GG activities that would compose CompaCT summaries. However, these minigames didn't have their focus on GG. Thus, the relevant data the logs revealed was restricted to the time taken for each meaningful action in the platform. This time was calculated using the difference between the timestamps registered on the selection of relevant elements. All timesheets are presented in Figures 33–43. Initially, another data we thought that could be helpful is the abstractometer data: how, when and to which LoA the user got into. But for most of the activity, it was a single decision at the beginning of a minigame, as can be seen in the interviews (Appendix B–E). Thus, we deemed it irrelevant and excluded this data from the log summary.



	1	2	3	4	5
F8	6,21	2,63	2,95	1,65	1,12
F11	3,2	3,9	2,99	2,79	2,88
M12	3,37	1,73	0,91	0,86	0,93
F10	8,18	4,78	8,56	3,36	3,06

Figure 33 – More/Less Abstract minigame timesheet.
Source: Elaborated by the author.

Outliers. There are some wild outliers across the timesheets, such as the third More/Less Abstraction classification by F10 (Figure 33); the seventh pair matched by F10 (Figure 35); or the third step of Endless Instructions by F11 (Figure 40). This happened because there were no explicit time constraints presented to the students, and the time captured by the logs do not have the sensibility to analyze if the student was discussing, arguing or questioning. The outliers are all moments where the student was dialoguing with the interviewer. Most of the dialogs happened in between minigames, and therefore didn't influence much the assessment, but in the rare cases where it happened in the middle of a minigame, it had great impacts.

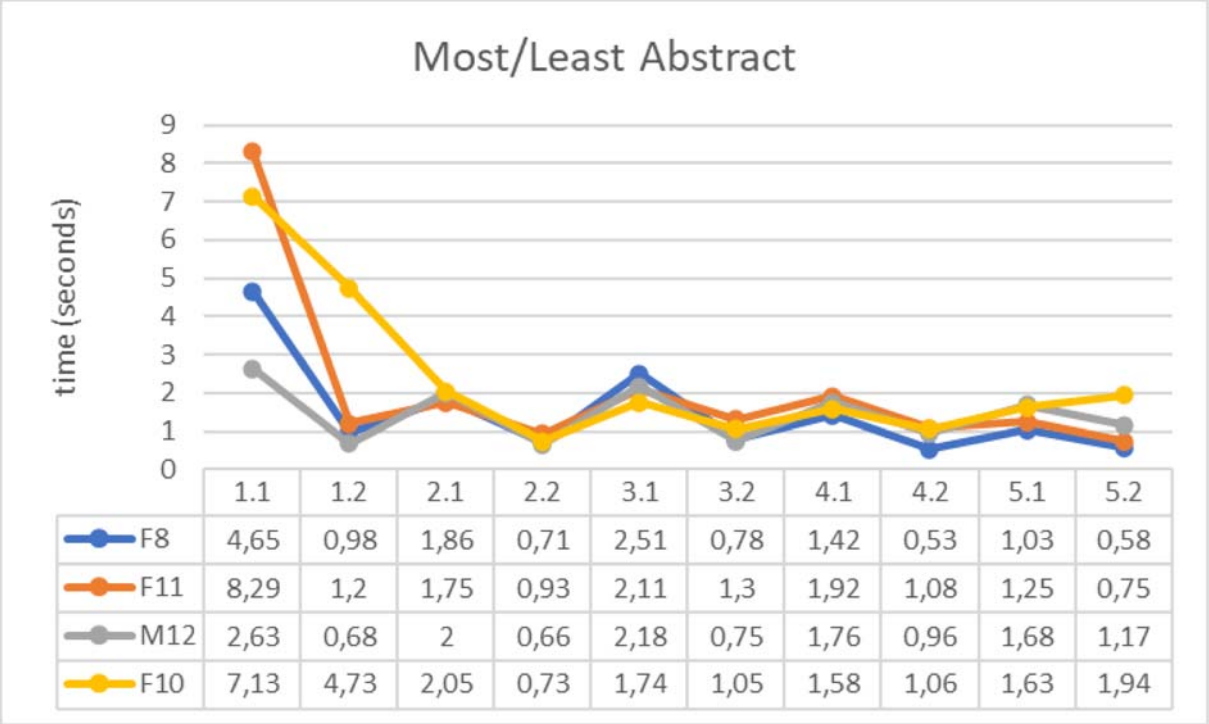
Slow start. Some of the minigames clearly showed a learning curve, with the students getting faster the more they played, such as in Match Pairs (Figure 35), or

at least a steep start, with students taking their time to comprehend the task, position and prepare themselves to complete it, such as in Most/Least Abstract (Figure 34) and Everything in 5 Steps (Figure 41). Noteworthy, the explanation of the minigame always happened before it was made available to play. That means this initial time taken by the students is on top of the presentation of the minigame.

Student profiles. Students had different profiles and that impacted the time they spent solving the minigames: F8 was talkative and liked to comment about each little thing, which slowed her; F11 was very timid, silent, and not very confident, but her objectivity counterweighted those; M12 had more of a gaming profile, he tried to be ready and complete the tasks as soon as they were given, regardless of the lack of explicit time constraints; F10 also struggled with confidence, she always looked at the interviewer seeking confirmation while completing the tasks, dragging her actions a little longer than she could. This reflected on the total time captured by the logs as effective time playing the minigames (Figure 43): 10:23 minutes for F8; 9:25 for F11; 6:51 for M12; and 9:46 for F10. In contrast with the total time of the activity (respectively 37:50, 35:00, 30:38, 36:55), we can say that only about a fourth (respectively 27.4%, 26.9%, 22.4% and 26.5%) of the time was spent actually solving tasks, the rest was spent in-between minigames, with explanations and disccusions.

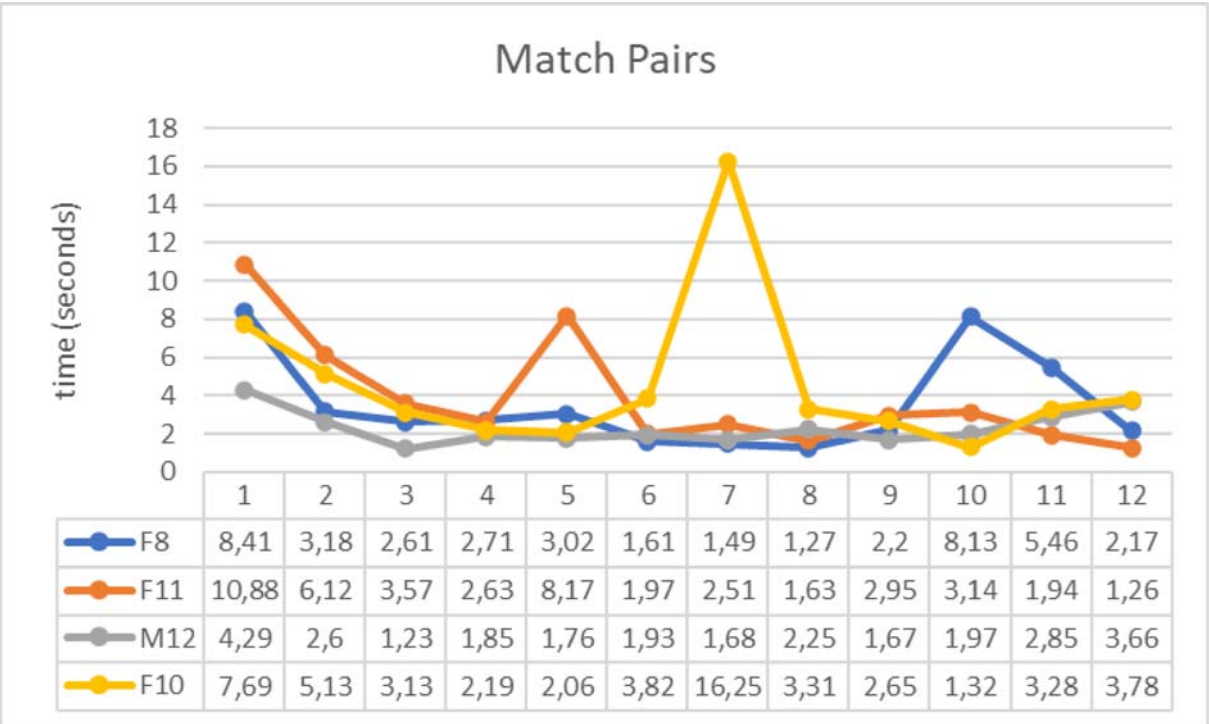
Cumulative impact. The Virtual Pet minigame (Figure 39) had an automated creation of needs for the pet every few seconds. When F8 started playing it, it was already running for a while, creating needs, which made her enter in a rushed loop: while she was up to attend to a need, another popped up. All students had to outrun the timers of needs creation, that was the game after all, but given F8's late start, by the time she completed the first series of needs, there was already a second and almost a third. Additionally, the minigames of Invitation Letters and Dancing Pairs (Figure 38) took students a long time because it was the first they had to use rules (to consult the invitations), so they needed some time to understand the interface. Then, especially F8, wasted some time due to the confusion with the minigame's mechanics, thinking it was a logic puzzle where all invitations should match.

Insights from the summary. All discussed in the previous paragraphs are examples of insights the data from the summary can give someone assessing the intervention. The data per se is risky to be part of a quantitative evaluation without proper interpretation, we designed CompaCT Logs to be a support tool for informed assessment, not the assessment itself. Due to time restraints, this process wasn't fully automated in GrameStation yet, so the summaries for this work were manually extracted from the logs. Alternatively, this interpretation of the data could be automated using AI too, but this requires further study.



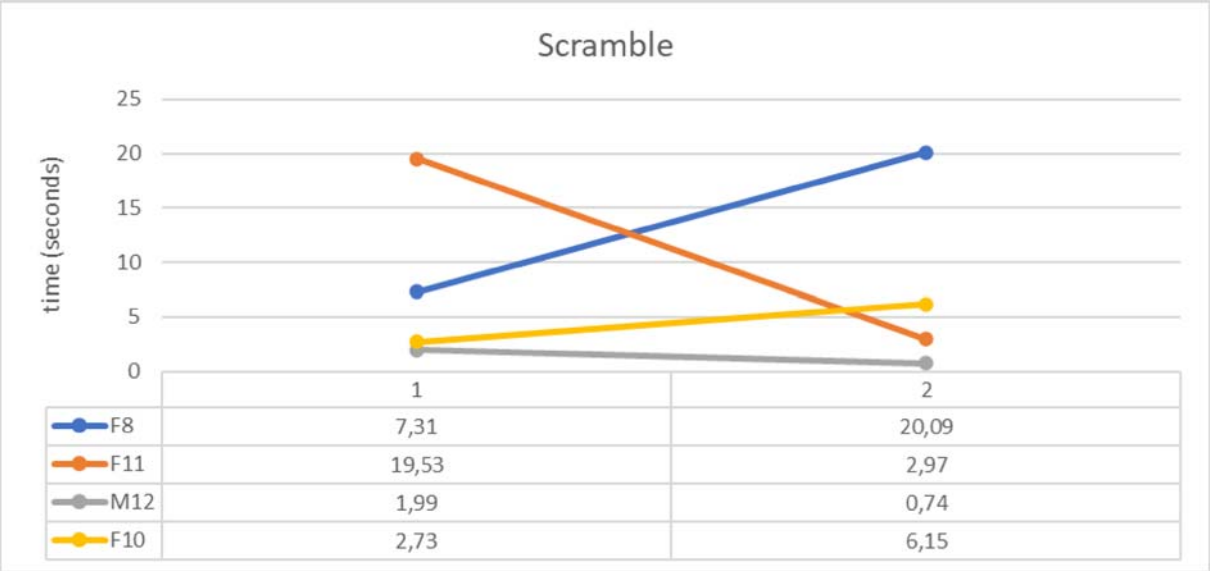
	1.1	1.2	2.1	2.2	3.1	3.2	4.1	4.2	5.1	5.2
F8	4,65	0,98	1,86	0,71	2,51	0,78	1,42	0,53	1,03	0,58
F11	8,29	1,2	1,75	0,93	2,11	1,3	1,92	1,08	1,25	0,75
M12	2,63	0,68	2	0,66	2,18	0,75	1,76	0,96	1,68	1,17
F10	7,13	4,73	2,05	0,73	1,74	1,05	1,58	1,06	1,63	1,94

Figure 34 – Most/Least Abstract minigame timesheet.
Source: Elaborated by the author.



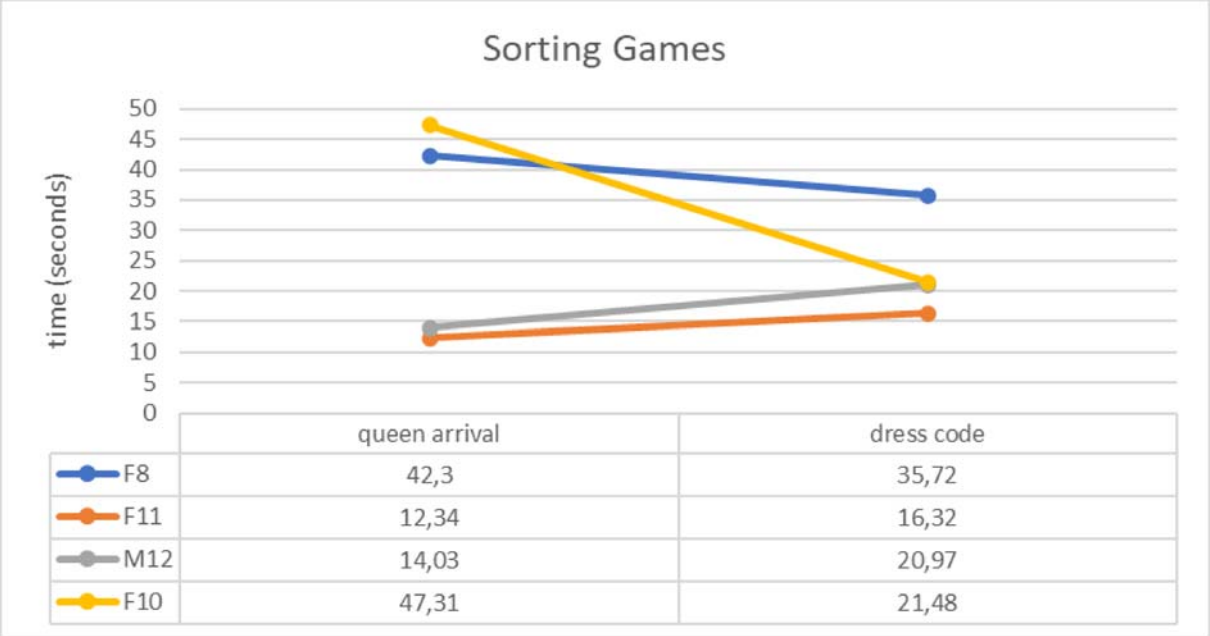
	1	2	3	4	5	6	7	8	9	10	11	12
F8	8,41	3,18	2,61	2,71	3,02	1,61	1,49	1,27	2,2	8,13	5,46	2,17
F11	10,88	6,12	3,57	2,63	8,17	1,97	2,51	1,63	2,95	3,14	1,94	1,26
M12	4,29	2,6	1,23	1,85	1,76	1,93	1,68	2,25	1,67	1,97	2,85	3,66
F10	7,69	5,13	3,13	2,19	2,06	3,82	16,25	3,31	2,65	1,32	3,28	3,78

Figure 35 – Match Pairs minigame timesheet.
Source: Elaborated by the author.



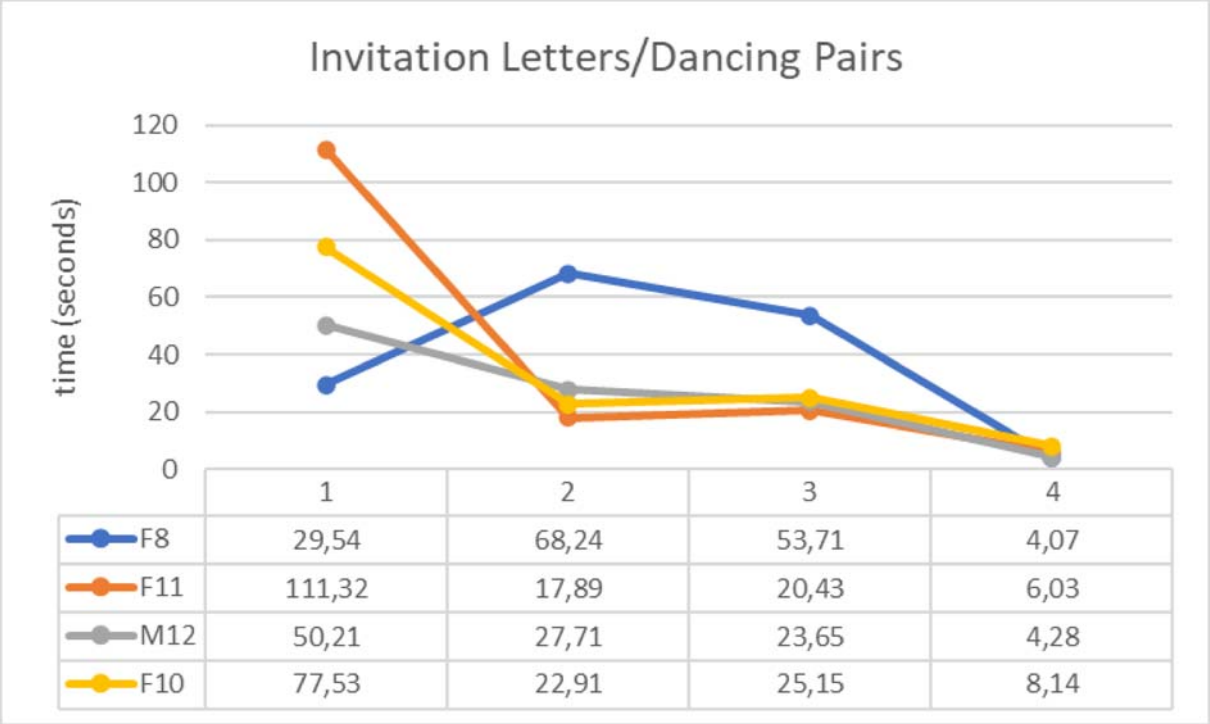
	1	2
F8	7,31	20,09
F11	19,53	2,97
M12	1,99	0,74
F10	2,73	6,15

Figure 36 – Scramble minigame timesheet.
Source: Elaborated by the author.



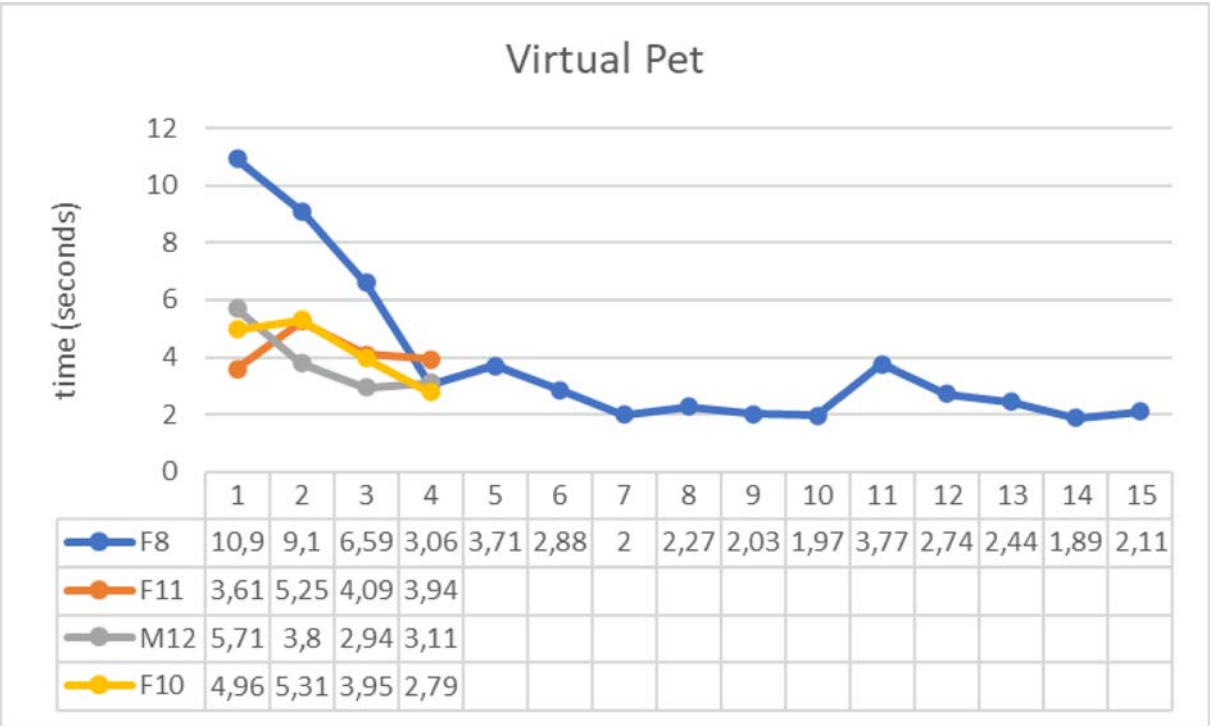
	queen arrival	dress code
F8	42,3	35,72
F11	12,34	16,32
M12	14,03	20,97
F10	47,31	21,48

Figure 37 – Queen Arrival and Dress Code minigames timesheet.
Source: Elaborated by the author.



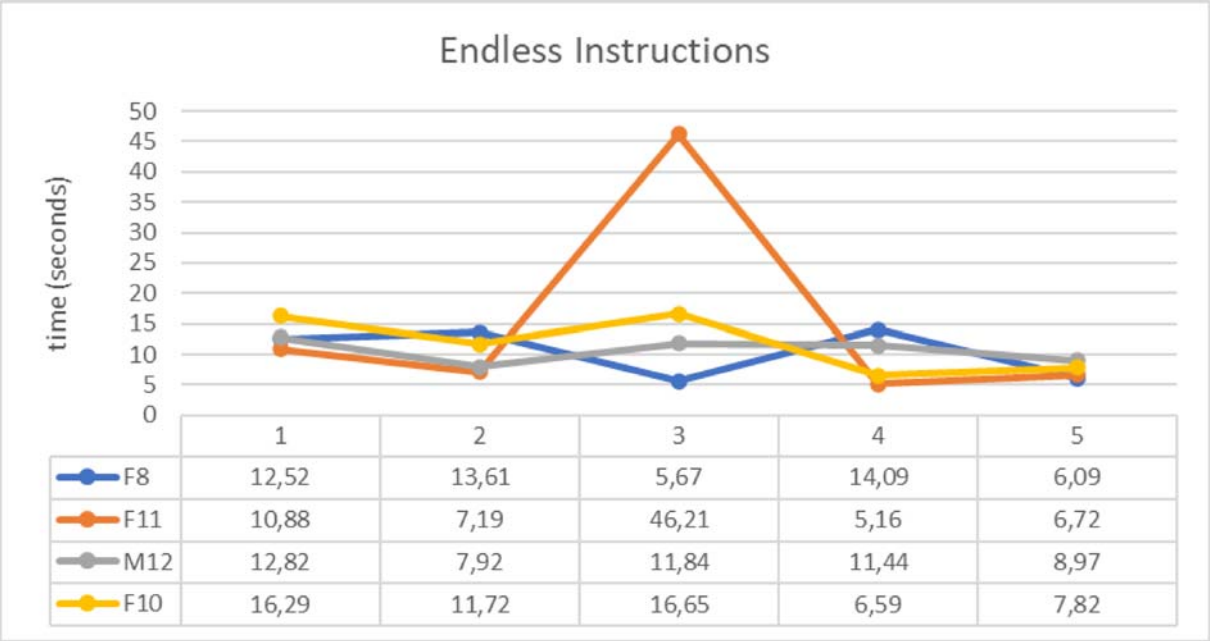
	1	2	3	4
F8	29,54	68,24	53,71	4,07
F11	111,32	17,89	20,43	6,03
M12	50,21	27,71	23,65	4,28
F10	77,53	22,91	25,15	8,14

Figure 38 – Invitation Letters and Dancing Pairs minigames timesheet.
Source: Elaborated by the author.



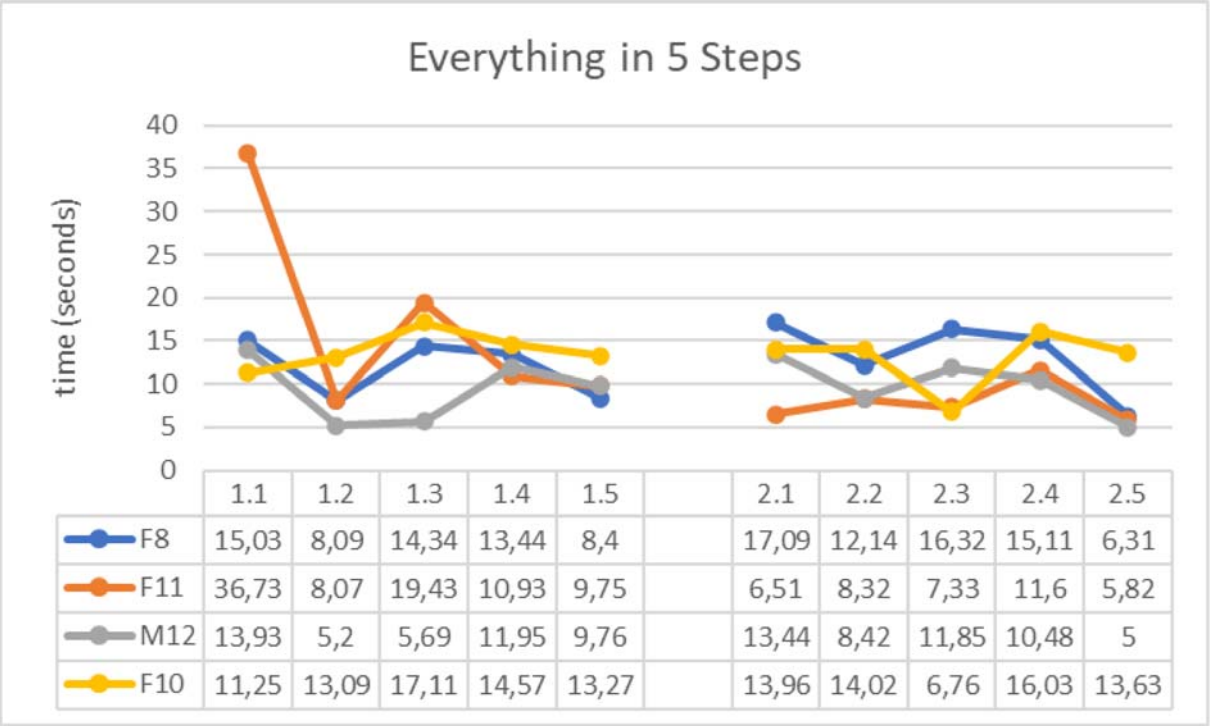
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
F8	10,9	9,1	6,59	3,06	3,71	2,88	2	2,27	2,03	1,97	3,77	2,74	2,44	1,89	2,11
F11	3,61	5,25	4,09	3,94											
M12	5,71	3,8	2,94	3,11											
F10	4,96	5,31	3,95	2,79											

Figure 39 – Virtual Pet minigame timesheet.
Source: Elaborated by the author.



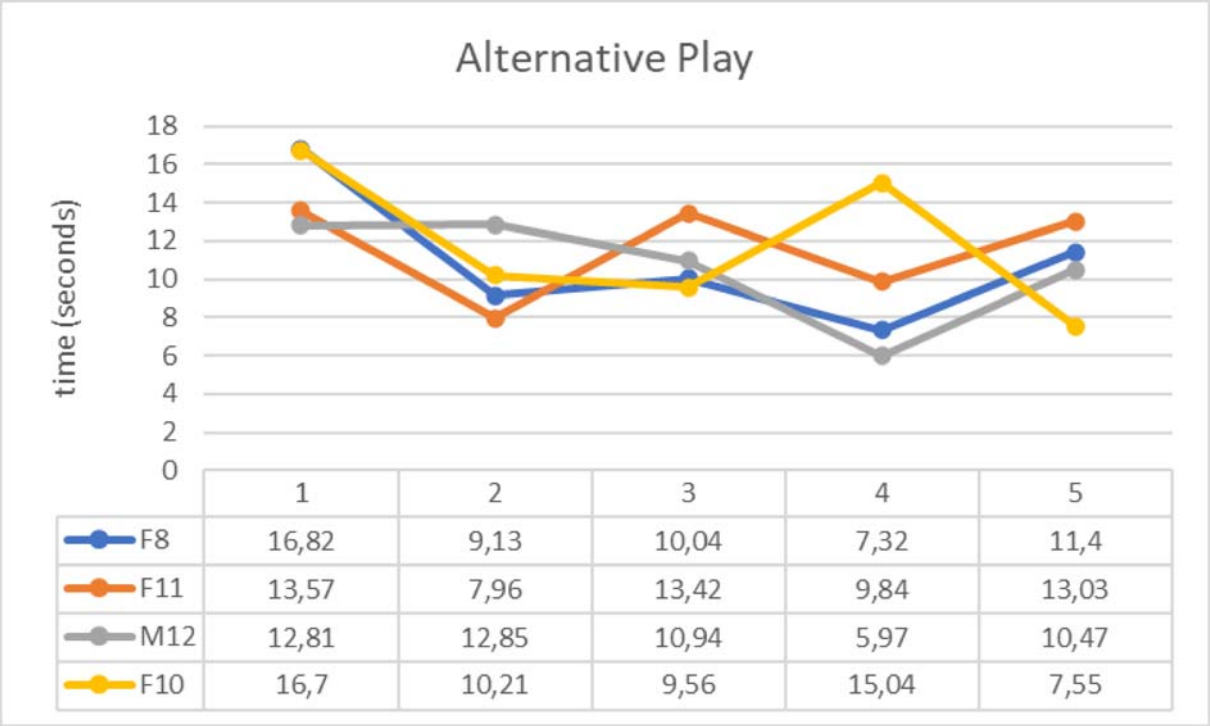
	1	2	3	4	5
F8	12,52	13,61	5,67	14,09	6,09
F11	10,88	7,19	46,21	5,16	6,72
M12	12,82	7,92	11,84	11,44	8,97
F10	16,29	11,72	16,65	6,59	7,82

Figure 40 – Endless Instructions minigame timesheet.
Source: Elaborated by the author.



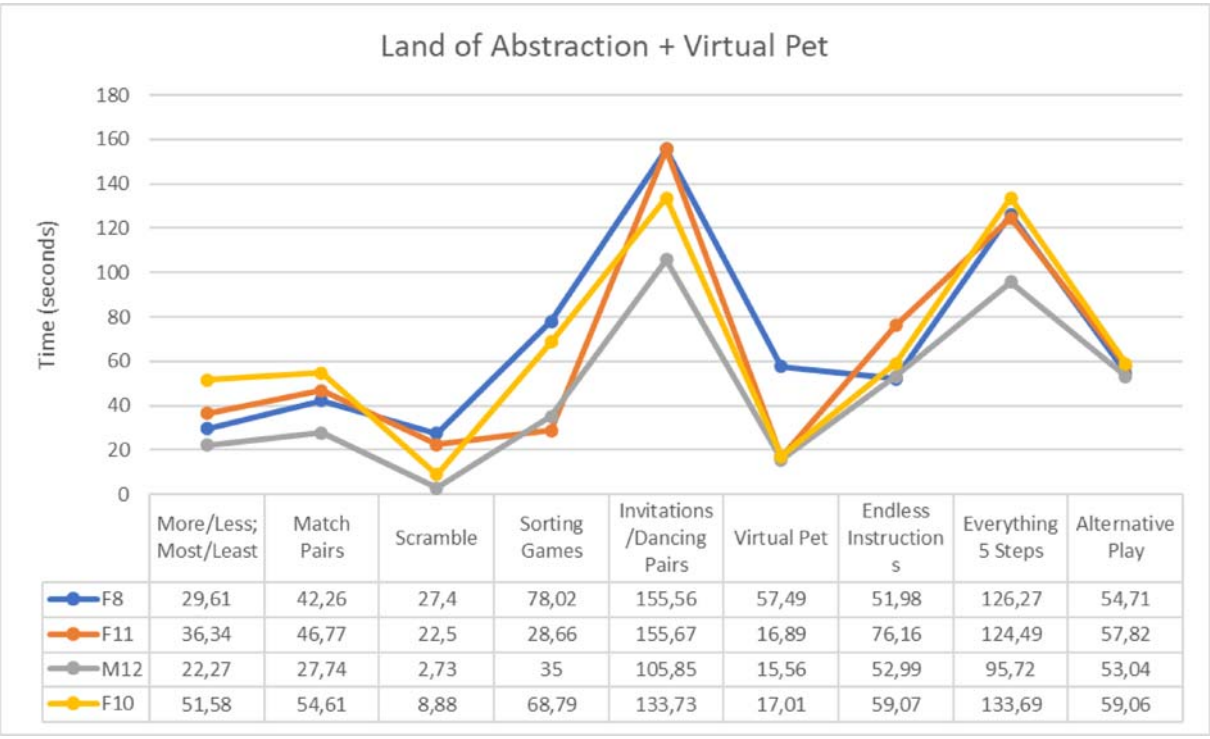
	1.1	1.2	1.3	1.4	1.5		2.1	2.2	2.3	2.4	2.5
F8	15,03	8,09	14,34	13,44	8,4		17,09	12,14	16,32	15,11	6,31
F11	36,73	8,07	19,43	10,93	9,75		6,51	8,32	7,33	11,6	5,82
M12	13,93	5,2	5,69	11,95	9,76		13,44	8,42	11,85	10,48	5
F10	11,25	13,09	17,11	14,57	13,27		13,96	14,02	6,76	16,03	13,63

Figure 41 – Everything in 5-step minigame timesheet.
Source: Elaborated by the author.



	1	2	3	4	5
F8	16,82	9,13	10,04	7,32	11,4
F11	13,57	7,96	13,42	9,84	13,03
M12	12,81	12,85	10,94	5,97	10,47
F10	16,7	10,21	9,56	15,04	7,55

Figure 42 – Alternative Play minigame timesheet.
Source: Elaborated by the author.



	More/Less; Most/Least	Match Pairs	Scramble	Sorting Games	Invitations /Dancing Pairs	Virtual Pet	Endless Instructions	Everything 5 Steps	Alternative Play
F8	29,61	42,26	27,4	78,02	155,56	57,49	51,98	126,27	54,71
F11	36,34	46,77	22,5	28,66	155,67	16,89	76,16	124,49	57,82
M12	22,27	27,74	2,73	35	105,85	15,56	52,99	95,72	53,04
F10	51,58	54,61	8,88	68,79	133,73	17,01	59,07	133,69	59,06

Figure 43 – Land of Abstraction and Virtual Pet timesheet.
Source: Elaborated by the author.

Design Diary. The cognitive interview support system was conceived to be implemented using Speech-To-Text technology intermediated by a pedagogical agent inside GrameStation. This would be the automation of the Think Aloud Protocol. We would harness the logs information to track a detailed trajectory of the activity along with the respective student's thoughts narrated by themselves. We expect that with the use of LLMs the explanations shouldn't be a problem for the pedagogical agent to replace the interviewer, which could feel even more natural using Text-To-Speech technology too. This automation, however, is for the future, in this work we tested a "manual", relying on humans, recordings and transcriptions. But this is the first part, the data gathering. We managed to use ChatGPT 4-o for the qualitative assessment that uses this data as input. More work is necessary to properly test the capabilities of LLMs in that matter, but we took a peek at its potential.

Prompt engineering. Without major concerns about the development of an optimal prompt, we used the following to get feedback from the LLM concerning the interview transcriptions:

> I want a qualitative assessment of the attached transcribed interview concerning the students' competencies related to Layers of Abstraction (LoA). Give me an overview and more objective insights targeting each of the following 4 competencies, each with 4 subskills: (1) LoA Recognition (1.1 - Distinguish between abstract and concrete, 1.2 - Connect multiple representations, 1.3 - Visualize in layers of abstraction, and 1.4 - Acknowledge information hiding); (2) LoA Calibration (2.1 - Assess relevance of details, 2.2 - Navigate layers of abstraction, 2.3 - Evaluate layers of abstraction, and 2.4 - Select the best layer of abstraction); (3) LoA Interaction (3.1 - Cross layers of abstraction, 3.2 - Identify how they are connected, 3.3 - Analyze the impact on others, 3.4 - Replace layers of abstraction); and (4) LoA modeling (4.1 - Simplify removing unnecessary details, 4.2 - Generalize recognizing patterns, 4.3 - Refine adding necessary details, and 4.4 - Decompose dealing with complexity). Highlight the subskills that became more evident during the interview and those that didn't show up much. Focus on the students' answers and mention their specifics when elaborating on the assessment.

Prompt impact. The LLM assessment upon the interviews can be read in the Annex C. The first big thing to note is that given the prompt used, the LLM had no information about the structure of the activity and how it was separated in minigames, each targeting a specific subskill. Therefore, the LLM analyzed the interview as a whole, picking examples of manifestations of subskills all over the activities (from minigames that weren't supposed to target them). The information given to the LLM about the competencies was also a very superficial version of our Proficiency model, basically

listing all KSAs and subskills. Without their detailed descriptions, they were open to different interpretations of what their “names” meant. A more complete prompt would likely result in a better analysis, but this also requires further research to be tested.

Correct and mistaken insights. The LLM could identify well some of the underexplored subskills, such as the impact analysis from LoA Interaction, decomposition and simplification from LoA Modeling. On the other hand, it identified generalization during the sorting minigame. Whether the concept of generalization is related to the process of sorting or not is debatable, but generalization as a modeling subskill as conceived in our Proficiency model does not seem to fit into such minigame. The LLM also seemed to interpret the cross LoA subskill in a way distant from our definition and it was not able to recognize the LoA replacement that occurred when the pet was replaced at the end of the activity. Overall, it did sound commentaries and could contribute to assessing, especially when we think about escalating this to several students. But before it is safe to use that tool, it is necessary to provide a richer, well-engineered prompt and a proper analysis of the risks, mistakes and errors.

5.5 Presentation Model

How does it look? The presentation model defines the means by which the assessment instrument will be delivered. It is the face of the instrument, describing “how the tasks appear in various settings, providing a style sheet for organizing the material to be presented and captured” (MISLEVY; ALMOND; LUKAS, 2003). In our case, the instrument is delivered as a grame, an educational game based on GG made in GrameStation. Actually, it was two games[2], each including several minigames reporting to each target KSA, as seen in Table 7 and 8. Here we will present the underlying GG for them and show its gameplay interfaces in GrameStation. All images used in the grames were taken from GrameStation basic pack, or generated by private software of generative AI, namely Firefly from Adobe[3].

Land of Abstraction types. The first grame made use of wrappers to create a hierarchy of representation along three LoA. As seen in its type graph (Figure 44), we have a generic character type wrapping up princesses, superheroes and pets, each of which respectively wraps up the specific types: princess of the swamp, princess of the sea, princess of the snow; superheroes with green, red and blue uniforms; lizard, dog and fish. There is a castle, to where characters go when correctly using their invitations. A generic food that wraps up specific food for dogs, fish and lizards. And an orange tile, with a loop representing ways between its instances, that was used for the Returning Pets Home minigame.

[2]Both games can be downloadded and played at:
`https://wp.ufpel.edu.br/pensamentocomputacional/gramestation-pt/jogos/`

[3]`https://firefly.adobe.com/`

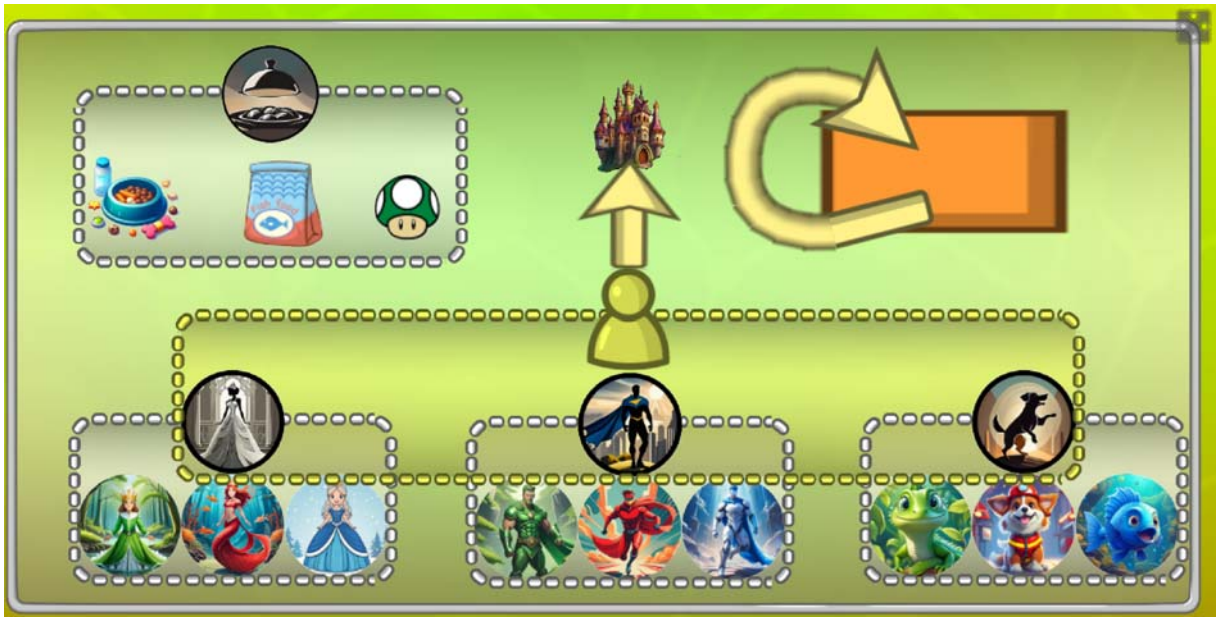


Figure 44 – The type graph of the Land of Abstraction grame.
Source: Elaborated by the author.

Land of Abstraction start. The initial graph of Land of Abstraction (Figure 45) contains a castle and one of each specific character, all nested and wrapped up in their respective more abstract representations. The grame had a simple, default pilot graph (Figure 46) describing a single player with access to all rules of the game from the start, initializing the game in phase with ID:4 (our initial graph, Figure 45) and revealing the existence of another phase with ID:15 (Return Pets Home minigame, Figure 53). During the start, students were presented to the abstractometer on the right corner of the GrameStation interface and the different LoA views for the game (Figure 47).

Figure 45 – The initial graph of the Land of Abstraction grame.
Source: Elaborated by the author.

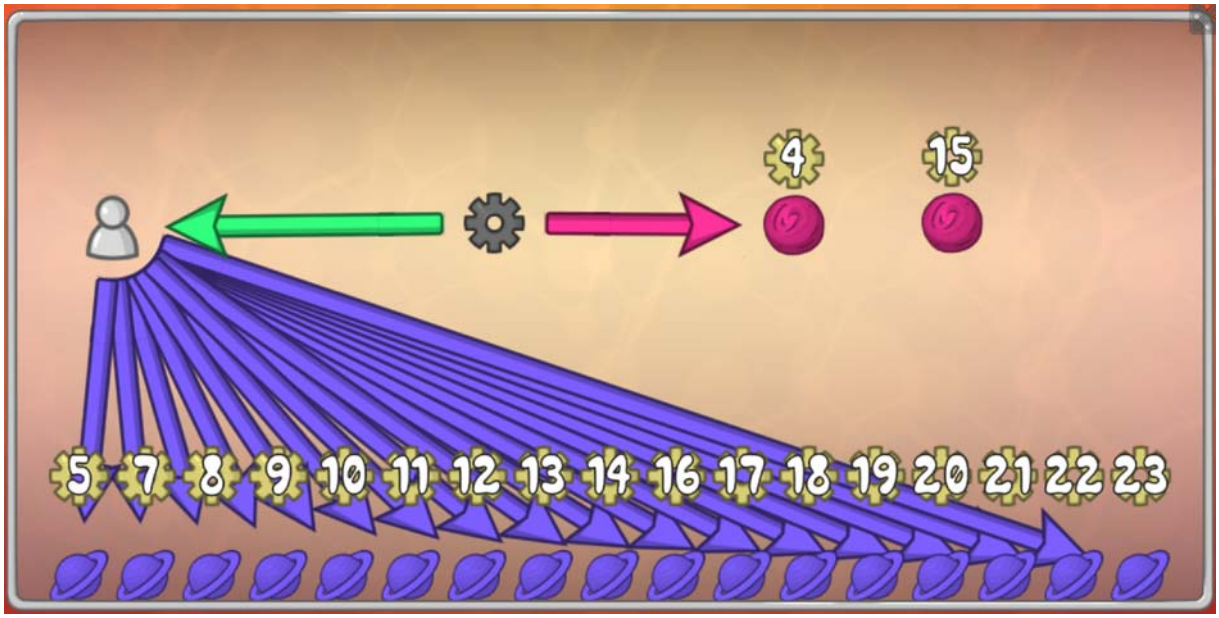


Figure 46 – The pilot graph of the Land of Abstraction grame.
Source: Elaborated by the author.

Figure 47 – The Land of Abstraction LoA views in GrameStation, from top to bottom.
Source: Elaborated by the author.

Land of Abstraction first minigames. All the introductory minigames weren't modeled by the GG, relying on simple reorganization of the elements on the screen. GrameStation allows the users to drag and drop elements at their will, we made use of that feature to complete the first minigames: More/Less Abstract; Most/Least Abstract; Match Pairs; Scramble (Figure 48); Queen Arrival (Figure 49, left); and Dress Code (Figure 49, right).

Figure 48 – The Scramble minigame in Land of Abstraction through all LoA views, from highest to lowest (from right to left).
Source: Elaborated by the author.

Figure 49 – The sorting minigames of Land of Abstraction: Queen Arrival (left), sort by role; and Dress Code (right), sort by color.
Source: Elaborated by the author.

Land of Abstraction invitation minigames. The first set of rules of the GG that describes the game are for the Invitation Letters and Dancing Pairs minigames. At first, they were designed to link the guests to the castle, but for that, the rules would have to include wrappers all the way up to the generic character vertex, since it is the one that may have a relation with the castle vertex. For simplicity's sake, we just showed the two vertices (a specific character and a role). We still made a different rule for each invitation letter, so the user could select them to consult at any time (Figure 50). It was expected that the abstractometer would be used to switch LoA in order to check roles for pairing (Figure 51, left and right).

Figure 50 – The rule selection interface for the Invitation Letters minigame.
Source: Elaborated by the author.

Figure 51 – The Invitation Letters and Dancing Pairs minigames of Land of Abstraction. Mid LoA view (left); and Low LoA view (right).
Source: Elaborated by the author.

Land of Abstraction pets minigames. After the Dancing Pairs, we call a rule to load a second phase (Figure 53). During this phase, we designed four rules (Figure 54) for the minigame Return Pets Home: generic pet move (top left); dog move (top right); fish move (bottom left); and lizard move (bottom right). They all move a pet from one tile to another. For the next minigame, Pet Feeding, we designed three more rules (Figure 52) for once they reached the top tiles, with the food: feed dog (left); feed fish (center); and feed lizard (right). They all feed each specific pet with a specific food, deleting it. They all look the same rule when seen on mid LoA (Figure 55, left), but reveal to be different when seen on low LoA (Figure 55, right).

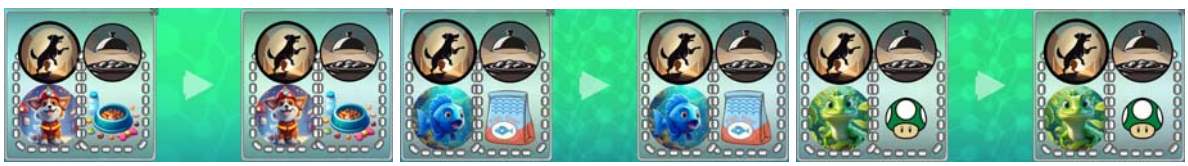


Figure 52 – Pet Feeding minigame rules: feed dog (left); fish (center); and lizard (right).
Source: Elaborated by the author.

Figure 53 – The second phase graph of the Land of Abstraction grame.
Source: Elaborated by the author.

Figure 54 – Return Pets Home minigame rules: generic pet move (top left); dog move (top right); fish move (bottom left); and lizard move (bottom right).
Source: Elaborated by the author.

Figure 55 – Pet Feeding minigame under different views: mid LoA (left); and low LoA (right).
Source: Elaborated by the author.

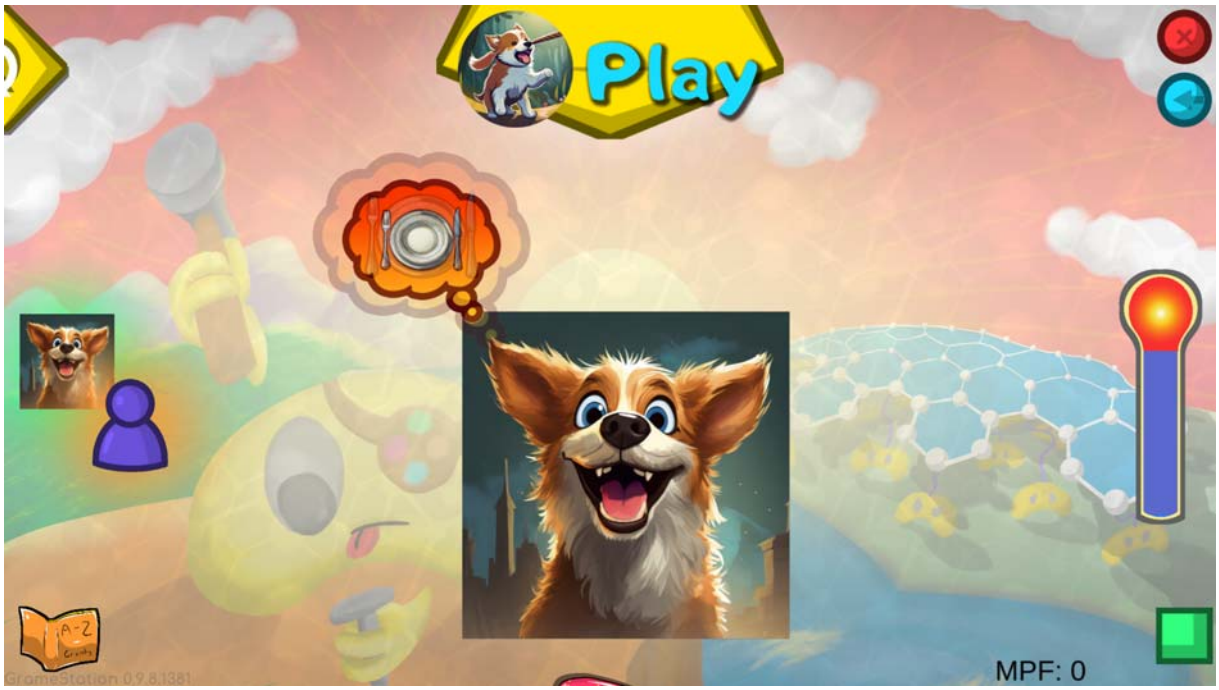


Figure 56 – Virtual Pet grame in GrameStation.
Source: Elaborated by the author.

Virtual Pet. The Pet Feeding minigame concludes Land of Abstraction. Then we go to the second grame, Virtual Pet. It is a quite simple game (Figure 56) where the player is notified of the pet's needs through balloons and must choose the right action (Figure 57) to attend to them. It has a very simple GG (Figure 58) with the following types (top): dog, hunger, dirt, and sadness, represented respectively by the red, green and blue balloons. A dog in its initial graph (mid) and 6 rules (bottom), being 3 pairs to create and delete each need: get hungry (top left); feed (top right); get dirty (mid left); clean (mid right); get sad (bottom left); and play (bottom right).

Figure 57 – Virtual Pet actions/rule choices.
Source: Elaborated by the author.

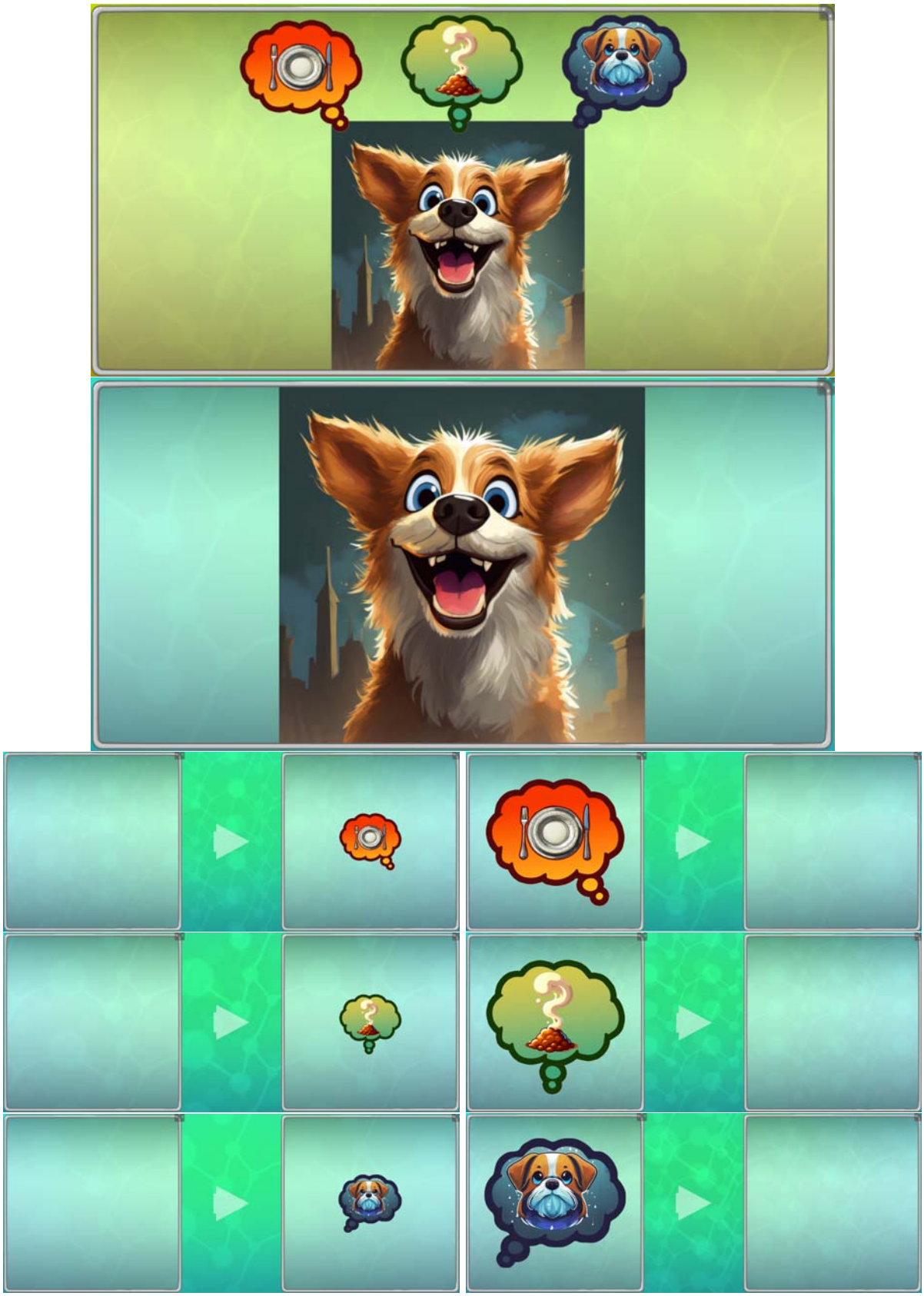


Figure 58 – Virtual Pet GG: type graph (top); initial graph (mid); and rule set (bottom).
Source: Elaborated by the author.

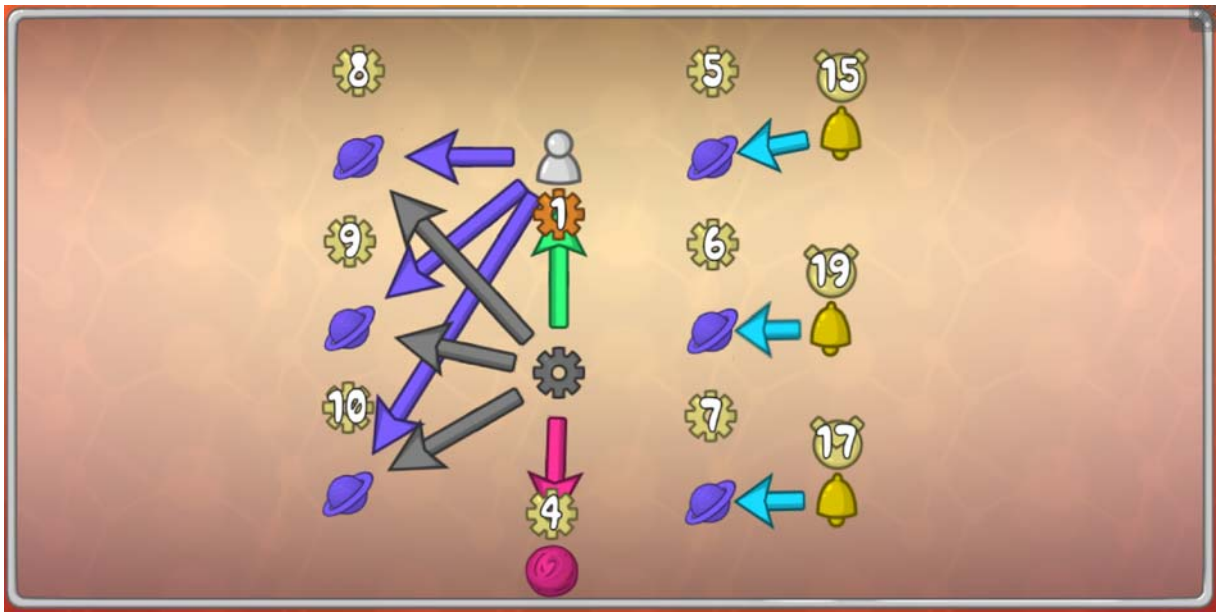


Figure 59 – Virtual Pet pilot graph.
Source: Elaborated by the author.

Automated grame. Contrary to Land of Abstraction, the Virtual Pet grame had automatic rules being applied by the engine. This is controlled by gears, as seen in the pilot graph (Figure):

- the cog in the center represents the engine itself;
- the white vertex represents the player, and its ID 1 defines the character this player roleplays in this game (shown at the left corner in Figure 56);
- the green edge represents the engine is currently allowing this player to play;
- the pink vertex represents a phase, our initial graph (Figure 58, top);
- the pink edge represents that the engine is loading this phase (that is what makes this phase the initial graph);
- the purple planets represent rules with their IDs (5, 6 and 7 are get hungry, dirty and sad, while 8, 9 and 10 are feed, clean and play);
- the yellow bells represent watchers, agents that can apply rules on their own;
- their clocks are timers setting the period they will try to apply the rule they are connected to by the blue edges;
- purple edges represent access from the player to certain rules; and
- gray edges represent rules where the interface will be hidden (the player only needed to select the balloon, without seeing the rule, in order to apply it, as seen in Figure 56, where the rule Play is selected, but its LHS and RHS are hidden)

PS3: ECD^{GS}: Layers of Abstraction, an evidence-centered design psychometric model for LoA.
PS4: Abstraction Land^{GS}, an educational game made in GrameStation to introduce concepts of LoA to children.

Recap. This chapter presented how we reached the conclusion that the proposed solutions above would be a reasonable approach aiming our goals. In this regard, we discussed the development of assessment instruments and the importance of evidentiary reasoning; we presented a KSA graph of LoA as a Proficiency model featuring LoA recognition, calibration, interaction and modeling; a generic Task model after all 16 subskills of the KSA graph, along with its instantiation under the graming approach; a composite assessment system merging the quantitative analysis of GrameScore, the hybrid one of CompaCT logs and the qualitative analysis of the Design Diary into an Evidence model; and the whole GG that specifies the Land of Abstraction and Virtual pet grames as Presentation model.

6 CONCLUSION

The problem. From two fields experiencing a really disruptive period comes CSEd with a series of urgencies as fast-paced as technology. Computing starts entering basic, compulsory education under the arguments of the general purpose of CT, but without a proper way to assess its variable components. A never-ending, ever-growing wave of new technologies invades the world of education, demanding teachers to learn more and more, to teach more and more. The oversaturation of teacher training suggests a shift in our solutions from providing more training to requiring less training. The practice reveals CT to be more about coding with VPL than about its theoretical conceptualizations. Being loyal to the original concept imposes the challenge of teaching abstract problem-solving skills, especially in k-12. The introduction of novel psychometric constructs with CT and computing as a whole raises concerns about the validity and reliability of the activities and tools fostering and assessing them.

The solution. In the pursuit of an alternative for introducing computing centered on the problem-solving process prior to coding, we proposed testing if specification could handle it. For teacher training, we sought non-demanding ways to find, create and use educational assets, testing if specification tools could be friendly to use and made easy to find and share. In the end, we elected a GG gaming approach to be the most aligned with education trends we could in our alternative to introduce computing. For CT, we focused on abstraction and its iconic manifestation in CS with LoA. For working with LoA in k-12, we added to an educational GG based game engine a feature able to bring hierarchical organization to graphs. With that new tool, we approached KSA related to LoA in a friendly way, using games, an “abstractometer” and a PA to guide the users. For assessing such KSA we designed a game that goes along with a cognitive interview, based on a psychometric framework.

The methods. We reviewed what has been done with CT in the last years through an overview of CT SLR. The analysis revealed educational trends, such as PjBL and Maker Culture, as well as theories, such as the theory of Flow in GBL. In an effort to harness the best of these, we proposed GBL with GG. This proposal was made possible thanks to the iterative development of an educational game engine based on GG

in a process analogous to the grounded theory. That is, experiments generating data were run along the development and the analysis of the results dictated the directions that would be taken from there on. We used PA to engage users and present functionalities in a humanized and friendly way. We applied UX and Usability tests such as SUS and AttrackDiff to assess how far from being friendly and self-contained the engine was (so not much training would be necessary). We used HGG to extend the theoretical foundation of the platform, bringing a way to model and interact with LoA. We delved into the ECD framework and evidentiary reasoning to build a solid assessment instrument for LoA. We reviewed the CT literature on abstraction to create a KSA graph for LoA via domain analysis, analyzed by an advisory panel. We used a small-scale cognitive interview experiment to check the LoA Task models we proposed. We designed a response scoring scheme to reach a quantitative assessment, summary and presentation of log analysis to reach a hybrid assessment, and TA protocol to reach a qualitative assessment. We developed two grames in GrameStation to implement the task model. And we took advantage of the recent boom of LLMs to take a peek at how they could enhance and facilitate such assessments.

The results. The key results of this work are long-term oriented and yet to be fully validated. The graming approach is extremely versatile and open to be explored in various different ways. This work laid the foundations and tools, so future research can progress investigating strengths and weaknesses. Results of most of the experiments conducted during this work served the purpose of guiding further development for the platform. On the other hand, the products are numerous and purposeful. The GG game engine, GrameStation, allows non-specialists to manage GG in a ludic environment, a direction with concerns and goals that are different from all other GG tools. The wrappers feature allows the design and execution of activities of varied natures to approach LoA in GG. The ECD^{GS}:LoA contributes to the operationalization of CT as a problem-solving process, psychometrically modeling abstraction under the perspective of LoA. The LoA assessment games, Abstraction Land and Virtual Pet, carefully introduce key concepts of LoA, while allowing to assess the students capabilities.

The contributions. The main contributions to theory is: to explore the characteristics of GG specification as an alternative approach to introduce CT; to show smarter, self-contained tools, concerned with UX and usability, as an alternative to avoid teacher training. The main contributions to practice is: to offer a complete ecosystem of ready-to-use tools for approaching computing with GG specification; to materialize a reliable assessment instrument for LoA as part of CT. But the most important contribution of this work is being a solid first step into a new take on CT using GG and how educational tools can become a powerhouse for automated and/or supported assessment, opening up various fronts of future work.

Text-to-Grame. One of our projects targets the integration of natural language

processing (NLP) to develop a Text-To-Grame system, enabling the automatic generation of functional games from verbal descriptions. This system would allow users to input natural language narratives, which would then be parsed and interpreted to create structured game elements, such as types, phases and rules, represented within the GG-based framework of grames. Leveraging NLP techniques, the system would bridge the gap between natural language and GG modeling, automating the design process and opening up game creation to a broader audience, including those without technical expertise. Beyond allowing the rapid prototyping of games based on high-level ideas or storytelling, the goal of this project is to hide the GG automatically created and use it as support for a PA to guide the user while he/she is modeling his/her game.

Talk to me, Gramer. Another front looks forward to integrating LLMs, PAs, text-to-speech, and speech-to-text technologies to enable direct and natural communication with users of the grame engine. By leveraging these tools, users will be able to interact with the engine through spoken language, making the game design process more intuitive and accessible. Gramers will guide users, offering real-time feedback and suggestions while ensuring a smooth dialogue. Additionally, internal systems will be used to automatically analyze the grame on the fly as it's being specified, identifying opportunities for applying modeling techniques such as generalization, refinement, and modularization. This would help users enhance their problem-solving and design skills, promoting the development of advanced modeling concepts in an unobtrusive, ubiquitous manner. The system aims to not only simplify game creation but also foster deeper understanding of complex modeling practices through natural, interactive learning.

Stories, Built-in Craftable Tutorials. This path was considered since the first iterations of the development of the engine, because we needed tutorials for new users. However, new features appeared all the time, and tutorials for specific activities independent of new features of the engine also started to be necessary. Because of this, we thought of a system where users could create their own tutorials to share, instead of the developers, strengthening the sense of community. The system of "stories" was partially implemented in GrameStation. The stories are made of scenes and casts; the actors, dialogues and scenarios of the casts can be positioned in scenes that have configurable conditions to be called. Scenes overlay GrameStation basic UI, being suitable even for stories about the UI. Scenes may have all kinds of conditions not just to appear, but to disappear too, enabling the creation of those tutorial scenes that will ask you to click a button and won't go off until you do so. The whole system is highly configurable, so it can be used for creating "cinematics" and "quests" in addition to the tutorials. However, it is a system more complex than the usual grame specification, so it is expected that it will be restricted to advanced users.

Custom assessment. A different direction focuses on developing systems that

empower users to define custom assessment rules for providing feedback to games based on collected data. These systems will allow users to create personalized frameworks for how points are gained or lost in the GrameScore system, offering flexible, data-driven evaluation of game performance. Additionally, users will be able to specify which log data should be highlighted in CompaCT summaries, ensuring that the most relevant information is featured for analysis. The project will also introduce tools for defining rubrics to guide the automated qualitative assessment of the Design Diary, enabling more structured and meaningful feedback on the design process. The custom assessment components would be guided by the same ECD principles employed in this work, allowing users to define KSA graphs and Task Models. This approach promotes a highly customizable and adaptive evaluation system, enhancing the educational value of the grame engine by aligning assessments with user-defined goals and metrics.

REFERENCES

ABANAZIR, C. Institutionalisation in e-sports. **Sport, Ethics and Philosophy**, [S.l.], v.13, n.2, p.117–131, 2019.

AHMAD, I.; POTHUGANTI, K. Design & implementation of real time autonomous car by using image processing & IoT. In: THIRD INTERNATIONAL CONFERENCE ON SMART SYSTEMS AND INVENTIVE TECHNOLOGY (ICSSIT), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.107–113.

AHMED, I. **Modeling and Supporting Middle School Mathematics Collaboration using Student Motivation across Different Digital Learning Platforms**. 2023. Tese (Doutorado em Ciência da Computação) — University of Pittsburgh.

AKÇAYIR, G.; AKÇAYIR, M. The flipped classroom: A review of its advantages and challenges. **Computers & Education**, [S.l.], v.126, p.334–345, 2018.

ALLEN, D. E.; DONHAM, R. S.; BERNHARDT, S. A. Problem-based learning. **New directions for teaching and learning**, [S.l.], v.2011, n.128, p.21–29, 2011.

ANGELI, C. et al. A K-6 computational thinking curriculum framework: Implications for teacher knowledge. **Journal of Educational Technology & Society**, [S.l.], v.19, n.3, 2016.

ARANDA, G.; FERGUSON, J. P. Unplugged Programming: The future of teaching computational thinking? **Pedagogika**, [S.l.], v.68, n.3, 2018.

ARAUJO, A. L. S. O.; ANDRADE, W. L.; GUERRERO, D. D. S.; MELO, M. R. A. How many abilities can we measure in computational thinking? A study on Bebras challenge. In: ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 50., 2019. **Anais...** [S.l.: s.n.], 2019. p.545–551.

ARIELI-ATTALI, M.; CAYTON-HODGES, G. Expanding the CBAL™ mathematics assessments to elementary grades: the development of a competency model and a rational number learning progression. **ETS Research Report Series**, [S.l.], v.2014, n.1, p.1–41, 2014.

ARIELI-ATTALI, M.; WARD, S.; THOMAS, J.; DEONOVIC, B.; VON DAVIER, A. A. The expanded evidence-centered design (e-ECD) for learning and assessment systems: A framework for incorporating learning goals and processes within assessment design. **Frontiers in psychology**, [S.l.], v.10, p.853, 2019.

ARMONI, M. On teaching abstraction in CS to novices. **Journal of Computers in Mathematics and Science Teaching**, [S.l.], v.32, n.3, p.265–284, 2013.

ARMONI, M. Computer science, computational thinking, programming, coding: the anomalies of transitivity in K-12 computer science education. **ACM Inroads**, [S.l.], v.7, n.4, p.24–27, 2016.

ARSLANYILMAZ, A.; SHARIF, B. Eye Tracking in Assessing Computational Thinking. In: SOCIETY FOR INFORMATION TECHNOLOGY & TEACHER EDUCATION INTERNATIONAL CONFERENCE, 2018. **Anais...** [S.l.: s.n.], 2018. p.9–14.

ATMATZIDOU, S.; DEMETRIADIS, S. Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. **Robotics and Autonomous Systems**, [S.l.], v.75, p.661–670, 2016.

AZNAR, A.; SOWDEN, P.; BAYLESS, S.; ROSS, K.; WARHURST, A.; PACHI, D. Home-schooling during COVID-19 lockdown: Effects of coping style, home space, and everyday creativity on stress and home-schooling outcomes. **Couple and Family Psychology: Research and Practice**, [S.l.], 2021.

AZZI, G. G. et al. The verigraph system for graph transformation. In: **Graph Transformation, Specifications, and Nets**. [S.l.]: Springer, Cham, 2018. p.160–178.

BAECKER, A. N.; GEISKKOVITCH, D. Y.; GONZÁLEZ, A. L.; YOUNG, J. E. Emotional support domestic robots for healthy older adults: Conversational prototypes to help with loneliness. In: COMPANION OF THE 2020 ACM/IEEE INTERNATIONAL CONFERENCE ON HUMAN-ROBOT INTERACTION, 2020. **Anais...** [S.l.: s.n.], 2020. p.122–124.

BAR-EL, D.; E. RINGLAND, K. Crafting game-based learning: An analysis of lessons for minecraft education edition. In: INTERNATIONAL CONFERENCE ON THE FOUNDATIONS OF DIGITAL GAMES, 15., 2020. **Proceedings...** [S.l.: s.n.], 2020. p.1–4.

BARANA, A.; MARCHISIO, M.; RABELLINO, S. Automated assessment in mathematics. In: IEEE 39TH ANNUAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, 2015., 2015. **Anais...** [S.l.: s.n.], 2015. v.3, p.670–671.

BATTISTELLA, P.; WANGENHEIM, C. G. von. Games for teaching computing in higher education–a systematic review. **IEEE Technology and Engineering Education Journal**, Piscataway, NJ, EUA, v.9, n.1, p.8–30, 2016.

BECKER, K.; GOTTSCHLICH, J. AI Programmer: autonomously creating software programs using genetic algorithms. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE COMPANION, 2021. **Proceedings...** [S.l.: s.n.], 2021. p.1513–1521.

BEGHETTO, R. A. Creative learning: A fresh look. **Journal of Cognitive Education and Psychology**, [S.l.], v.15, n.1, p.6–23, 2016.

BEGHETTO, R. A. Creative learning in education. In: **The Palgrave handbook of positive education**. [S.l.]: Palgrave Macmillan, Cham, 2021. p.473–491.

BELL, T.; ALEXANDER, J.; FREEMAN, I.; GRIMLEY, M. Computer science unplugged: School students doing real computing without computers. **The New Zealand Journal of Applied Computing and Information Technology**, [S.l.], v.13, n.1, p.20–29, 2009.

BENNEDSEN, J.; CASPERSEN, M. Model-driven programming. In: **Reflections on the Teaching of Programming**. [S.l.]: Springer, 2008. p.116–129.

BERGMANN, J.; SAMS, A. **Flip your classroom**: Reach every student in every class every day. [S.l.]: International society for technology in education, 2012.

BERMINGHAM, S.; CHARLIER, N.; DAGNINO, F.; DUGGAN, J.; EARP, J.; KIILI, K.; LUTS, E.; STOCK, L. van der; WHITTON, N. Approaches to collaborative game-making for fostering 21st century skills. In: EUROPEAN CONFERENCE ON GAMES BASED LEARNING, 2013. **Anais...** [S.l.: s.n.], 2013. p.45.

BESSEN, J. Toil and technology: Innovative technology is displacing workers to new jobs rather than replacing them entirely. **Finance & Development**, [S.l.], v.52, n.001, 2015.

BOCCONI, S.; CHIOCCARIELLO, A.; DETTORI, G.; FERRARI, A.; ENGELHARDT, K.; KAMPYLIS, P.; PUNIE, Y. Developing computational thinking in compulsory education. **European Commission, JRC Science for Policy Report**, [S.l.], 2016.

BONTCHEV, B.; ANTONOVA, A.; DANKOV, Y. Educational video game design using personalized learning scenarios. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE AND ITS APPLICATIONS, 2020. **Anais...** [S.l.: s.n.], 2020. p.829–845.

BOWEN, J. P.; HINCHEY, M. G. Seven more myths of formal methods. **IEEE software**, [S.l.], v.12, n.4, p.34–41, 1995.

BOWEN, J.; STAVRIDOU, V. Safety-critical systems, formal methods and standards. **Software Engineering Journal**, [S.l.], v.8, n.4, p.189–209, 1993.

BRAYNE, S. The criminal law and law enforcement implications of big data. **Annual Review of Law and Social Science**, [S.l.], v.14, p.293–308, 2018.

BRENNAN, K.; RESNICK, M. New frameworks for studying and assessing the development of computational thinking. In: AMERICAN EDUCATIONAL RESEARCH ASSOCIATION, VANCOUVER, CANADA, 2012., 2012. **Proceedings...** [S.l.: s.n.], 2012. v.1, p.25.

BRIGHOUSE, H. **On education**. [S.l.]: Routledge, 2006.

BRILLIANT, S. S.; WISEMAN, T. R. The first programming paradigm and language dilemma. In: SIGCSE TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 1996. **Proceedings...** [S.l.: s.n.], 1996. p.338–342.

BUNKER, A.; GOPALAKRISHNAN, G.; MCKEE, S. A. Formal hardware specification languages for protocol compliance verification. **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, [S.l.], v.9, n.1, p.1–32, 2004.

BUSATTO, G.; KREOWSKI, H.-J.; KUSKE, S. Abstract hierarchical graph transformation. **Mathematical Structures in Computer Science**, [S.l.], v.15, n.4, p.773–819, 2005.

CAVALHEIRO, S. A. d. C.; FOSS, L.; RIBEIRO, L. Theorem proving graph grammars with attributes and negative application conditions. **Theoretical Computer Science**, Amsterdan, Netherlands, v.686, p.25–77, 2017.

CETINIC, E.; SHE, J. Understanding and creating art with AI: Review and outlook. **ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)**, [S.l.], v.18, n.2, p.1–22, 2022.

CHEN, P.; YANG, D.; METWALLY, A. H. S.; LAVONEN, J.; WANG, X. Fostering computational thinking through unplugged activities: A systematic literature review and meta-analysis. **International Journal of STEM Education**, [S.l.], v.10, n.1, p.47, 2023.

CHOMSKY, N. **Topics in the theory of generative grammar**. [S.l.]: Walter de Gruyter, 2013. v.56.

CORRADINI, A.; MONTANARI, U.; ROSSI, F.; EHRIG, H.; HECKEL, R.; LÖWE, M. Algebraic approaches to graph transformation–part i: Basic concepts and double pushout approach. In: **Handbook Of Graph Grammars And Computing By Graph Transformation**: Volume 1: Foundations. Singapore: World Scientific, 1997. p.163–245.

CRUZ-BENITO, J.; VISHWAKARMA, S.; MARTIN-FERNANDEZ, F.; FARO, I. Automated source code generation and auto-completion using deep learning: Comparing and discussing current language model-related approaches. **AI**, [S.l.], v.2, n.1, p.1–16, 2021.

CSIKSZENTMIHALYI, M. Flow: The psychology of optimal experience (Vol. 41). **HarperPerennial**, New York, 1991.

CSIZMADIA, A.; CURZON, P.; DORLING, M.; HUMPHREYS, S.; NG, T.; SELBY, C.; WOOLLARD, J. Computational thinking–A guide for teachers. **Computing At School**, [S.l.], 2015.

DAGIENE, V.; FUTSCHEK, G. Bebras international contest on informatics and computer literacy: Criteria for good tasks. In: INTERNATIONAL CONFERENCE ON INFORMATICS IN SECONDARY SCHOOLS-EVOLUTION AND PERSPECTIVES, 2008. **Proceedings...** [S.l.: s.n.], 2008. p.19–30.

DANIELSSON, O.; HOLM, M.; SYBERFELDT, A. Augmented reality smart glasses in industrial assembly: Current status and future challenges. **Journal of Industrial Information Integration**, [S.l.], v.20, p.100175, 2020.

DEEVA, G.; BOGDANOVA, D.; SERRAL, E.; SNOECK, M.; DE WEERDT, J. A review of automated feedback systems for learners: Classification framework, challenges and opportunities. **Computers & Education**, [S.l.], v.162, p.104094, 2021.

DEMETRIADIS, S.; BARBAS, A.; MOLOHIDES, A.; PALAIGEORGIOU, G.; PSILLOS, D.; VLAHAVAS, I.; TSOUKALAS, I.; POMBORTSIS, A. “Cultures in negotiation”: teachers’ acceptance/resistance attitudes considering the infusion of technology into schools. **Computers & Education**, [S.l.], v.41, n.1, p.19–37, 2003.

DEMPERE, J.; MODUGU, K. P.; HESHAM, A.; RAMASAMY, L. The impact of ChatGPT on higher education. **Dempere J, Modugu K, Hesham A and Ramasamy LK (2023) The impact of ChatGPT on higher education. Front. Educ**, [S.l.], v.8, p.1206936, 2023.

DENNING, P. J.; TEDRE, M. **Computational Thinking**. [S.l.]: Mit Press, 2019.

DENNING, P. J.; TEDRE, M. Computational thinking: A disciplinary perspective. **Informatics in Education**, [S.l.], v.20, n.3, p.361–390, 2021.

DORODCHI, M.; DEHBOZORGI, N.; FALLAHIAN, M.; POURIYEH, S. Teaching Software Engineering using Abstraction through Modeling. **Informatics in Education**, [S.l.], v.20, n.4, 2021.

ROZENBERG, G. (Ed.). **Handbook of graph grammars and computing by graph transformation**: volume I. Foundations. Singapore: World Scientific Publishing, 1997.

ERICSSON, K. A.; SIMON, H. A. How to study thinking in everyday life: Contrasting think-aloud protocols with descriptions and explanations of thinking. **Mind, Culture, and Activity**, [S.l.], v.5, n.3, p.178–186, 1998.

ESPINOSA, M. P.; GARDEAZABAL, J. Optimal correction for guessing in multiple-choice tests. **Journal of Mathematical psychology**, [S.l.], v.54, n.5, p.415–425, 2010.

EZEAMUZIE, N. O.; LEUNG, J. S.; TING, F. S. Unleashing the potential of abstraction from cloud of computational thinking: A systematic review of literature. **Journal of Educational Computing Research**, [S.l.], v.60, n.4, p.877–905, 2022.

FAGERLUND, J. et al. Computational thinking in programming with scratch in primary schools: A systematic review. **Computer Applications in Engineering Education**, [S.l.], v.29, n.1, p.12–28, 2021.

FERREIRA, A. P. L. **Object-oriented graph grammars**. 2005. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul.

FERREIRA, A. P. L.; FOSS, L.; RIBEIRO, L. Formal verification of object-oriented graph grammars specifications. **Electronic Notes in Theoretical Computer Science**, [S.l.], v.175, n.4, p.101–114, 2007.

Finney, K. Mathematical notation in formal specification: too difficult for the masses? **IEEE Transactions on Software Engineering**, [S.l.], v.22, n.2, p.158–159, Feb 1996.

FIRAT, M. How chat GPT can transform autodidactic experiences and open education. **Department of Distance Education, Open Education Faculty, Anadolu Unive**, [S.l.], 2023.

FREY, C. B.; OSBORNE, M. A. The future of employment: How susceptible are jobs to computerisation? **Technological forecasting and social change**, [S.l.], v.114, p.254–280, 2017.

FU, J. Complexity of ICT in education: A critical literature review and its implications. **International Journal of education and Development using ICT**, [S.l.], v.9, n.1, p.112–125, 2013.

GAUTAM, A.; BORTZ, W.; TATAR, D. Abstraction through multiple representations in an integrated computational thinking environment. In: ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 51., 2020. **Proceedings...** [S.l.: s.n.], 2020. p.393–399.

GIANNAKOS, M. N.; PAPPAS, I. O.; JACCHERI, L.; SAMPSON, D. G. Understanding student retention in computer science education: The role of environment, gains, barriers and usefulness. **Education and Information Technologies**, [S.l.], v.22, n.5, p.2365–2382, 2017.

GONZÁLEZ-PÉREZ, L. I.; RAMÍREZ-MONTOYA, M. S. Components of Education 4.0 in 21st century skills frameworks: systematic review. **Sustainability**, [S.l.], v.14, n.3, p.1493, 2022.

GRIER, R. A.; BANGOR, A.; KORTUM, P.; PERES, S. C. The system usability scale: Beyond standard usability testing. In: HUMAN FACTORS AND ERGONOMICS SOCIETY ANNUAL MEETING, 2013. **Proceedings...** [S.l.: s.n.], 2013. v.57, n.1, p.187–191.

GROVER, S.; BASU, S.; BIENKOWSKI, M.; EAGLE, M.; DIANA, N.; STAMPER, J. A framework for using hypothesis-driven approaches to support data-driven learning analytics in measuring computational thinking in block-based programming environments. **ACM Transactions on Computing Education (TOCE)**, [S.l.], v.17, n.3, p.1–25, 2017.

HALL, A. Seven myths of formal methods. **IEEE software**, [S.l.], v.7, n.5, p.11–19, 1990.

HAN, J.; LIU, G.; GAO, Y. Learners in the Metaverse: A systematic review on the use of roblox in learning. **Education Sciences**, [S.l.], v.13, n.3, p.296, 2023.

HASSENZAHL, M.; BURMESTER, M.; KOLLER, F. AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität. In: **Mensch & computer 2003**. [S.l.]: Springer, 2003. p.187–196.

HINCHEY, M.; BOWEN, J. P.; ROUFF, C. A. Introduction to formal methods. In: **Agent Technology from a Formal Perspective**. [S.l.]: Springer, 2006. p.25–64.

HOPPE, H. U.; WERNEBURG, S. Computational thinking—More than a variant of scientific inquiry. **Computational thinking education**, [S.l.], p.13–30, 2019.

UNESCO (Ed.). **A Guideline Framework for Hybrid Education, Learning, and Assessment (HELA)**. [S.l.]: Beijing: National Engineering Research Center for Cyber-learning and Intelligent Technology (CIT), 2022.

HUANG, W.; LOOI, C.-K. A critical review of literature on “unplugged” pedagogies in K-12 computer science and computational thinking education. **Computer Science Education**, [S.l.], v.31, n.1, p.83–111, 2021.

HUCK, C.; ZHANG, J. Effects of the COVID-19 Pandemic on K-12 Education: A Systematic Literature Review. **New Waves-Educational Research and Development Journal**, [S.l.], v.24, n.1, p.53–84, 2021.

HUSSEIN, M.; TRAUTMAN, L. J.; HOLLOWAY, R. Technology Employment, Information and Communication in the Digital Age. **Information and Communication in the Digital Age (January 8, 2021)**, [S.l.], 2021.

HUSSIN, A. A. Education 4.0 made simple: Ideas for teaching. **International Journal of Education and Literacy Studies**, [S.l.], v.6, n.3, p.92–98, 2018.

IOANNOU, A.; MAKRIDOU, E. Exploring the potentials of educational robotics in the development of computational thinking: A summary of current research and practical proposal for future work. **Education and Information Technologies**, [S.l.], v.23, n.6, p.2531–2544, 2018.

IQBAL, M. Z.; MANGINA, E.; CAMPBELL, A. G. Current challenges and future research directions in augmented reality for education. **Multimodal Technologies and Interaction**, [S.l.], v.6, n.9, p.75, 2022.

ISTE; CSTA. **Computational Thinking leadership toolkit**. 1.ed. [S.l.: s.n.], 2011.

JANTAKUN, T.; JANTAKOON, T. Digital Educational Computer Games Environments Supporting Education (DECGE-SE). **Higher Education Studies**, [S.l.], v.11, n.2, p.91–98, 2021.

KAKAVAS, P.; UGOLINI, F. C. Computational thinking in primary education: A systematic literature review. **Research on Education and Media**, [S.l.], v.11, n.2, p.64–94, 2019.

KALELIOGLU, F.; GULBAHAR, Y.; KUKUL, V. A framework for computational thinking based on a systematic research review. **Baltic J. Modern Computing**, [S.l.], 2016.

KE, F.; LIU, R.; SOKOLIKJ, Z.; DAHLSTROM-HAKKI, I.; ISRAEL, M. Using eye tracking for research on learning and computational thinking. In: INTERNATIONAL CONFERENCE ON HUMAN-COMPUTER INTERACTION, 2021. **Anais...** [S.l.: s.n.], 2021. p.216–228.

KESER, H.; SEMERCI, A. Technology trends, Education 4.0 and beyond. **Contemporary Educational Researches Journal**, [S.l.], v.9, n.3, p.39–49, 2019.

KIM, H.; CHOI, H.; KANG, H.; AN, J.; YEOM, S.; HONG, T. A systematic review of the smart energy conservation system: From smart homes to sustainable smart cities. **Renewable and Sustainable Energy Reviews**, [S.l.], v.140, p.110755, 2021.

KLIZIENE, I.; TAUJANSKIENE, G.; AUGUSTINIENE, A.; SIMONAITIENE, B.; CIBULSKAS, G. The impact of the virtual learning platform EDUKA on the academic performance of primary school children. **Sustainability**, [S.l.], v.13, n.4, p.2268, 2021.

KOKOTSAKI, D.; MENZIES, V.; WIGGINS, A. Project-based learning: A review of the literature. **Improving schools**, [S.l.], v.19, n.3, p.267–277, 2016.

KOLEK, L.; MOCHOCKI, M.; GEMROT, J. Review of Educational Benefits of Game Jams: Participant and Industry Perspective. **Homo Ludens**, [S.l.], n.1 (15), p.115–140, 2022.

KONOPKA, C. L.; ADAIME, M. B.; MOSELE, P. H. et al. Active teaching and learning methodologies: some considerations. **Creative Education**, [S.l.], v.6, n.14, p.1536, 2015.

KOSSAK, F.; MASHKOOR, A.; GEIST, V.; ILLIBAUER, C. Improving the understandability of formal specifications: An experience report. In: INTERNATIONAL WORKING CONFERENCE ON REQUIREMENTS ENGINEERING: FOUNDATION FOR SOFTWARE QUALITY, 2014. **Proceedings...** [S.l.: s.n.], 2014. p.184–199.

KRAMER, J. Abstraction in computer science & software engineering: A pedagogical perspective. **System Design Frontier Journal**, [S.l.], v.3, n.12, p.1–9, 2006.

KRISHNAMURTHI, S.; FISLER, K. 13 Programming Paradigms and Beyond. **The Cambridge handbook of computing education research**, [S.l.], p.377, 2019.

LAI, X.; WONG, G. K.-w. Collaborative versus individual problem solving in computational thinking through programming: A meta-analysis. **British Journal of Educational Technology**, [S.l.], v.53, n.1, p.150–170, 2022.

LIU, R.; LUO, F.; ISRAEL, M. What Do We Know about Assessing Computational Thinking? A New Methodological Perspective from the Literature. In: ACM CONFERENCE ON INNOVATION AND TECHNOLOGY IN COMPUTER SCIENCE EDUCATION V. 1, 26., 2021. **Proceedings...** [S.l.: s.n.], 2021. p.269–275.

LIU, Y.; YAO, Y.; TON, J.-F.; ZHANG, X.; CHENG, R. G. H.; KLOCHKOV, Y.; TAUFIQ, M. F.; LI, H. Trustworthy LLMs: a Survey and Guideline for Evaluating Large Language Models' Alignment. **arXiv preprint arXiv:2308.05374**, [S.l.], 2023.

LÓPEZ-BELMONTE, J.; SEGURA-ROBLES, A.; MORENO-GUERRERO, A.-J.; PARRA-GONZÁLEZ, M.-E. Robotics in education: a scientific mapping of the literature in Web of Science. **Electronics**, [S.l.], v.10, n.3, p.291, 2021.

LUO, F.; ISRAEL, M.; LIU, R.; YAN, W.; GANE, B.; HAMPTON, J. Understanding students' computational thinking through cognitive interviews: A learning trajectory-based analysis. In: ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 51., 2020. **Proceedings...** [S.l.: s.n.], 2020. p.919–925.

MANNILA, L.; DAGIENE, V.; DEMO, B.; GRGURINA, N.; MIROLO, C.; ROLANDSSON, L.; SETTLE, A. Computational thinking in K-9 education. In: OF THE 2014 INNOVATION & TECHNOLOGY IN COMPUTER SCIENCE EDUCATION CONFERENCE, 2014. **Proceedings...** ACM, 2014. p.1–29.

MCARTHUR, V.; TEATHER, R. J. Serious mods: A case for modding in serious games pedagogy. In: IEEE GAMES ENTERTAINMENT MEDIA CONFERENCE (GEM), 2015., 2015. **Anais...** [S.l.: s.n.], 2015. p.1–4.

MEDEIROS, R. P.; RAMALHO, G. L.; FALCÃO, T. P. A systematic literature review on teaching and learning introductory programming in higher education. **IEEE Transactions on Education**, [S.l.], v.62, n.2, p.77–90, 2018.

MERKX, C.; NAWIJN, J. Virtual reality tourism experiences: Addiction and isolation. **Tourism Management**, [S.l.], v.87, p.104394, 2021.

MERSAND, S. The state of makerspace research: A review of the literature. **TechTrends**, [S.l.], v.65, n.2, p.174–186, 2021.

MIRAZ, M. H.; ALI, M.; EXCELL, P. S. Adaptive user interfaces and universal usability through plasticity of user interface design. **Computer Science Review**, [S.l.], v.40, p.100363, 2021.

MIROLO, C.; IZU, C.; LONATI, V.; SCAPIN, E. Abstraction in Computer Science Education: An Overview. **Informatics in Education**, [S.l.], v.20, n.4, p.615–639, 2022.

MISLEVY, R. J.; ALMOND, R. G.; LUKAS, J. F. A brief introduction to evidence-centered design. **ETS Research Report Series**, [S.l.], v.2003, n.1, p.i–29, 2003.

MOLIN, G. The role of the teacher in game-based learning: A review and outlook. **Serious games and edutainment applications**, [S.l.], p.649–674, 2017.

MOYA, E. C. Using Active Methodologies: The StudentśView. **Procedia-Social and Behavioral Sciences**, [S.l.], v.237, p.672–677, 2017.

NASCIMENTO, S.; PÓLVORA, A. Maker cultures and the prospects for technological action. **Science and engineering ethics**, [S.l.], v.24, n.3, p.927–946, 2018.

NAVARRETE, C. C.; MINNIGERODE, L. Exploring 21 st Century Learning: Game design and creation, the students' experience. In: WORLD CONFERENCE ON EDUCATIONAL MULTIMEDIA, HYPERMEDIA AND TELECOMMUNICATIONS, 2013. **Proceedings...** [S.l.: s.n.], 2013. p.282–293.

O'MALLEY, C.; FRASER, D. S. **Literature review in learning with tangible technologies**. [S.l.]: Citeseer, 2004.

OPENAI. **GPT-4 Technical Report**. [S.l.]: OpenAI, 2023.

PAN, Z.; CUI, Y.; LEIGHTON, J. P.; CUTUMISU, M. Insights into computational thinking from think-aloud interviews: A systematic review. **Applied Cognitive Psychology**, [S.l.], v.37, n.1, p.71–95, 2023.

PAPADAKIS, S.; KALOGIANNAKIS, M. Evaluating a course for teaching introductory programming with Scratch to pre-service kindergarten teachers. **International Journal of Technology Enhanced Learning**, [S.l.], v.11, n.3, p.231–246, 2019.

PAPAVLASOPOULOU, S.; SHARMA, K.; GIANNAKOS, M.; JACCHERI, L. Using eye-tracking to unveil differences between kids and teens in coding activities. In: PROCEEDINGS OF THE 2017 CONFERENCE ON INTERACTION DESIGN AND CHILDREN, 2017. **Anais...** [S.l.: s.n.], 2017. p.171–181.

PERTTULA, A.; KIILI, K.; LINDSTEDT, A.; TUOMI, P. Flow experience in game based learning–a systematic literature review. **International Journal of Serious Games**, [S.l.], n.1 (4), p.57–72, 2017.

PETRI, C. A.; REISIG, W. Petri net. **Scholarpedia**, [S.l.], v.3, n.4, p.6477, 2008.

PETRI, G.; WANGENHEIM, C. G. von. How games for computing education are evaluated? A systematic literature review. **Computers & education**, [S.l.], v.107, p.68–90, 2017.

PETRI, G.; WANGENHEIM, C. G. von; BORGATTO, A. F. MEEGA+: an evolution of a model for the evaluation of educational games. **INCoD/GQS**, [S.l.], v.3, 2016.

PHELPS, G.; GITOMER, D. H.; IACONANGELO, C. J.; ETKINA, E.; SEELEY, L.; VOKOS, S. Developing assessments of content knowledge for teaching using evidence-centered design. **Educational Assessment**, [S.l.], v.25, n.2, p.91–111, 2020.

POMBO, L.; MARQUES, M. M. Guidelines for Teacher Training in Mobile Augmented Reality Games: Hearing the Teachers' Voices. **Education Sciences**, [S.l.], v.11, n.10, p.597, 2021.

POZO-RICO, T.; GILAR-CORBÍ, R.; IZQUIERDO, A.; CASTEJÓN, J.-L. Teacher training can make a difference: tools to overcome the impact of COVID-19 on primary schools. An experimental study. **International Journal of Environmental Research and Public Health**, [S.l.], v.17, n.22, p.8633, 2020.

PREGOWSKA, A.; MASZTALERZ, K.; GARLIŃSKA, M.; OSIAL, M. A worldwide journey through distance education—from the post office to virtual, augmented and mixed realities, and education during the COVID-19 pandemic. **Education Sciences**, [S.l.], v.11, n.3, p.118, 2021.

QIAN, M.; CLARK, K. R. Game-based Learning and 21st century skills: A review of recent research. **Computers in human behavior**, [S.l.], v.63, p.50–58, 2016.

QIAN, Y.; CHOI, I. Tracing the essence: ways to develop abstraction in computational thinking. **Educational technology research and development**, [S.l.], v.71, n.3, p.1055–1078, 2023.

QIU, K.; HAGHIASHTIANI, G.; MCALPINE, M. C. 3D printed organ models for surgical applications. **Annual review of analytical chemistry (Palo Alto, Calif.)**, [S.l.], v.11, n.1, p.287, 2018.

RAABE, A. L. A.; ZORZO, A. F.; FRANGO, I.; RIBEIRO, L.; GRANVILLE, L.; SALGADO, L.; CRUZ, M.; BIGOLIN, N.; CAVALHEIRO, S.; FORTES, S. Referenciais de formação em computação: Educação básica. **Sociedade Brasileira de Computação**, [S.l.], 2017.

RAHMAN, M. M.; WATANOBE, Y. ChatGPT for education and research: Opportunities, threats, and strategies. **Applied Sciences**, [S.l.], v.13, n.9, p.5783, 2023.

RAMESH, A.; PAVLOV, M.; GOH, G.; GRAY, S.; VOSS, C.; RADFORD, A.; CHEN, M.; SUTSKEVER, I. Zero-shot text-to-image generation. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 2021. **Anais...** [S.l.: s.n.], 2021. p.8821–8831.

RENSINK, A. The GROOVE simulator: A tool for state space generation. In: INTERNATIONAL WORKSHOP ON APPLICATIONS OF GRAPH TRANSFORMATIONS WITH INDUSTRIAL RELEVANCE, 2003, New York, NY, USA. **Proceedings...** [S.l.: s.n.], 2003. p.479–485.

RESNICK, M. et al. Scratch: Programming for all. **Communications of the ACM**, [S.l.], v.52, n.11, p.60–67, 2009.

RIBEIRO, L. Métodos formais de especificação: gramáticas de grafos. **VIII Escola de Informática da SBC-Sul**, Porto Alegre, RS, BR, p.1–33, 2000.

ROBINS, A. V. 12 Novice Programmers and Introductory Programming. **The Cambridge handbook of computing education research**, [S.l.], p.327, 2019.

ROMÁN-GONZÁLEZ, M. Computational Thinking Test: design guidelines and content validation. In: ANNUAL INTERNATIONAL CONFERENCE ON EDUCATION AND NEW LEARNING TECHNOLOGIES (EDULEARN 2015), 7., 2015. **Proceedings...** [S.l.: s.n.], 2015. p.2436–2444.

ROMÁN-GONZÁLEZ, M.; MORENO-LEÓN, J.; ROBLES, G. Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions. In: **Computational Thinking Education**. [S.l.]: Springer, 2019. p.79–98.

ROMÁN-GONZÁLEZ, M.; PÉREZ-GONZÁLEZ, J.-C.; JIMÉNEZ-FERNÁNDEZ, C. Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. **Computers in human behavior**, [S.l.], v.72, p.678–691, 2017.

ROSE, K.; ELDRIDGE, S.; CHAPIN, L. The internet of things: An overview. **The internet society (ISOC)**, [S.l.], v.80, p.1–50, 2015.

ROSSI, S.; SANTINI, S. J.; DI GENOVA, D.; MAGGI, G.; VERROTTI, A.; FARELLO, G.; ROMUALDI, R.; ALISI, A.; TOZZI, A. E.; BALSANO, C. et al. Using the Social Robot NAO for Emotional Support to Children at a Pediatric Emergency Department: Randomized Clinical Trial. **Journal of Medical Internet Research**, [S.l.], v.24, n.1, p.e29656, 2022.

SAMUEL, M. S. An insight into programming paradigms and their programming languages. **Journal of Applied Technology and Innovation**, [S.l.], v.1, n.1, p.37–57, 2017.

SANTOS, P. S.; ARAUJO, L. G. J.; BITTENCOURT, R. A. A Mapping Study of Computational Thinking and Programming in Brazilian K-12 Education. In: IEEE FRONTIERS IN EDUCATION CONFERENCE (FIE), 2018., 2018. **Proceedings...** IEEE, 2018. p.1–8.

SARI, T.; NAYIR, F. Challenges in distance education during the (Covid-19) pandemic period. **Qualitative Research in Education**, [S.l.], v.9, n.3, p.328–360, 2020.

SCHAGAEV, I.; FOLIC, N.; IOANNIDES, N.; BACON, E. Multiple choice answers approach: Assessment with penalty function for computer science and similar disciplines. **International Journal of Engineering Education**, [S.l.], v.28, n.6, p.1294, 2012.

SCHINA, D.; ESTEVE-GONZÁLEZ, V.; USART, M. An overview of teacher training programs in educational robotics: characteristics, best practices and recommendations. **Education and Information Technologies**, [S.l.], v.26, n.3, p.2831–2852, 2021.

SCHKOLNE, S.; ISHII, H.; SCHRODER, P. Immersive design of DNA molecules with a tangible interface. In: IEEE VISUALIZATION 2004, 2004. **Anais...** [S.l.: s.n.], 2004. p.227–234.

SCHNEIDER, W. J.; MCGREW, K. S. The Cattell–Horn–Carroll Theory of Cognitive Abilities. In: **Contemporary Intellectual Assessment**: Theories, Tests, and Issues. [S.l.]: Guilford Publications, 2022.

SELBY, C.; WOOLLARD, J. Refining an understanding of computational thinking. **University of Southampton (E-prints)**, [S.l.], 2014.

SENGUPTA, P.; KINNEBREW, J. S.; BASU, S.; BISWAS, G.; CLARK, D. Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. **Education and Information Technologies**, [S.l.], v.18, n.2, p.351–380, 2013.

SENTANCE, S.; BARENDSEN, E.; HOWARD, N.; SCHULTE, C. **Computer Science Education**: Perspectives on Teaching and Learning in School. [S.l.]: Bloomsbury Academic, 2023.

SHAW, M. J. **E-commerce and the Digital Economy**. [S.l.]: Routledge, 2015.

SHUTE, V. J.; SUN, C.; ASBELL-CLARKE, J. Demystifying computational thinking. **Educational Research Review**, [S.l.], v.22, p.142–158, 2017.

SHUTE, V. J.; TORRES, R. Where streams converge: Using evidence-centered design to assess Quest to Learn. **Technology-based assessments for 21st century skills: Theoretical and practical implications from modern research**, [S.l.], v.91124, 2012.

SILVA, J. V. da; SILVA JUNIOR, B. A. da; FOSS, L.; COSTA CAVALHEIRO, S. A. da. A User Experience and Usability Test on Playing Games Specified as Graph Grammars in GrameStation. In: XXX WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO, 2022. **Anais...** [S.l.: s.n.], 2022. p.298–309.

SILVA JUNIOR, B. A.; CAVALHEIRO, S. A. C.; FOSS, L. Revisitando um Jogo Educacional para desenvolver o Pensamento Computacional com Gramática de Grafos. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO-SBIE, 2019. **Anais...** [S.l.: s.n.], 2019. v.30, n.1, p.863–872.

SILVA JUNIOR, B. A.; CAVALHEIRO, S. A. d. C.; FOSS, L. A Última Árvore: exercitando o Pensamento Computacional por meio de um jogo educacional baseado em Gramática de Grafos. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO-SBIE, 2017. **Anais...** Porto Alegre: SBC, 2017. v.28, n.1, p.735–744.

SILVA JUNIOR, B. A. d. **GGasCT**: bringing formal methods to the computational thinking. 2020. Thesis (Master's) – Master of Computer Science, Center for Technological Development, Federal University of Pelotas, Rio Grande do Sul, 2020.

SILVA JUNIOR, B. A. da; SILVA, J. V. da; COSTA CAVALHEIRO, S. A. da; FOSS, L. Featuring Layers of Abstraction as a Modeling Resource in an Educational Game-Based Learning Platform. In: XXXIV SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 2023. **Anais...** [S.l.: s.n.], 2023. p.1874–1885.

SIMONET, C.; NOYCE, A. J. Domotics, Smart Homes, and Parkinson's Disease. **Journal of Parkinson's Disease**, [S.l.], v.11, n.s1, p.S55–S63, 2021.

SORMUNEN, K.; JUUTI, K.; LAVONEN, J. Maker-centered project-based learning in inclusive classes: supporting students' active participation with teacher-directed reflective discussions. **International Journal of Science and Mathematics Education**, [S.l.], v.18, n.4, p.691–712, 2020.

STAHL, B. C.; WRIGHT, D. Ethics and privacy in AI and big data: Implementing responsible research and innovation. **IEEE Security & Privacy**, [S.l.], v.16, n.3, p.26–33, 2018.

STRIUK, A. M.; SEMERIKOV, S. O. The dawn of software engineering education. In: COMPUTER SCIENCE & SOFTWARE ENGINEERING. PROCEEDINGS OF THE 2ND STUDENT WORKSHOP (CS&SE@ SW 2019), KRYVYI RIH, UKRAINE, NOVEMBER 29, 2019, 2019. **Anais...** [S.l.: s.n.], 2019. n.2546, p.35–57.

SUN, L.; GUO, Z.; HU, L. Educational games promote the development of students' computational thinking: a meta-analytic review. **Interactive Learning Environments**, [S.l.], v.31, n.6, p.3476–3490, 2023.

SVYATKOVSKIY, A.; ZHAO, Y.; FU, S.; SUNDARESAN, N. Pythia: Ai-assisted code completion system. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY & DATA MINING, 25., 2019. **Proceedings...** [S.l.: s.n.], 2019. p.2727–2735.

TAENTZER, G. AGG: A graph transformation environment for modeling and validation of software. In: INTERNATIONAL WORKSHOP ON APPLICATIONS OF GRAPH TRANSFORMATIONS WITH INDUSTRIAL RELEVANCE, 2003, New York, NY, USA. **Proceedings...** [S.l.: s.n.], 2003. p.446–453.

TANG, X.; YIN, Y.; LIN, Q.; HADAD, R.; ZHAI, X. Assessing computational thinking: A systematic review of empirical studies. **Computers & Education**, [S.l.], v.148, p.103798, 2020.

TECHNOLOGIES, U. **Unity Real-Time Development Platform | 3D, 2D, VR and AR Engine**. Unity Technologies, 2020. Available at: `https://unity.com/`. Accessed: 2020-02-17.

TEKDAL, M. Trends and development in research on computational thinking. **Education and Information Technologies**, [S.l.], v.26, n.5, p.6499–6529, 2021.

TERZIMEHIĆ, N. Real-World Presence in the Era of Ubiquitous Computing. In: EXTENDED ABSTRACTS OF THE 2021 CHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2021. **Anais...** [S.l.: s.n.], 2021. p.1–4.

THORNHILL-MILLER, B.; CAMARDA, A.; MERCIER, M.; BURKHARDT, J.-M.; MORISSEAU, T.; BOURGEOIS-BOUGRINE, S.; VINCHON, F.; EL HAYEK, S.; AUGEREAU-LANDAIS, M.; MOUREY, F. et al. Creativity, critical thinking, communication, and collaboration: assessment, certification, and promotion of 21st century skills for the future of work and education. **Journal of Intelligence**, [S.l.], v.11, n.3, p.54, 2023.

TIKVA, C.; TAMBOURIS, E. Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. **Computers & Education**, [S.l.], v.162, p.104083, 2021.

United Nations. **Universal Declaration of Human Rights**. [S.l.: s.n.], 1948.

United Nations. **International Standard Classification of Education**. [S.l.: s.n.], 2011.

VAUDOUR, F.; HEINZE, A. Software as a service: Lessons from the video game industry. **Global Business and Organizational Excellence**, [S.l.], v.39, n.2, p.31–40, 2020.

VIBERG, O.; KHALIL, M.; BAARS, M. Self-regulated learning and learning analytics in online learning environments: A review of empirical research. In: OF THE TENTH INTERNATIONAL CONFERENCE ON LEARNING ANALYTICS & KNOWLEDGE, 2020. **Proceedings...** [S.l.: s.n.], 2020. p.524–533.

WEINTROP, D.; BEHESHTI, E.; HORN, M.; ORTON, K.; JONA, K.; TROUILLE, L.; WILENSKY, U. Defining computational thinking for mathematics and science classrooms. **Journal of Science Education and Technology**, [S.l.], v.25, n.1, p.127–147, 2016.

WING, J. M. A specifier's introduction to formal methods. **Computer**, [S.l.], v.23, n.9, p.8–22, 1990.

WING, J. M. Computational thinking. **Communications of the ACM**, [S.l.], v.49, n.3, p.33–35, 2006.

WING, J. M. Computational thinking and thinking about computing. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, [S.l.], v.366, n.1881, p.3717–3725, 2008.

WOO, B. **Getting a life**: The social worlds of Geek culture. [S.l.]: McGill-Queen's Press-MQUP, 2018.

XIONG, J.; HSIANG, E.-L.; HE, Z.; ZHAN, T.; WU, S.-T. Augmented reality and virtual reality displays: emerging technologies and future perspectives. **Light: Science & Applications**, [S.l.], v.10, n.1, p.1–30, 2021.

XU, D. Tangible user interface for children-an overview. In: UCLAN DEPARTMENT OF COMPUTING CONFERENCE, 2005. **Proceedings...** [S.l.: s.n.], 2005.

YADAV, A.; GRETTER, S.; GOOD, J.; MCLEAN, T. Computational thinking in teacher education. In: **Emerging research, practice, and policy on computational thinking**. [S.l.]: Springer, 2017. p.205–220.

YANG, K.; LIU, X.; CHEN, G. The influence of robots on students' computational thinking: A literature review. **International Journal of Information and Education Technology**, [S.l.], v.10, n.8, p.5, 2020.

YANG, Y.-T. C. Virtual CEOs: A blended approach to digital gaming for enhancing higher order thinking and academic achievement among vocational high school students. **Computers & Education**, [S.l.], v.81, p.281–295, 2015.

ZENG, J.; PARKS, S.; SHANG, J. To learn scientifically, effectively, and enjoyably: A review of educational games. **Human Behavior and Emerging Technologies**, [S.l.], v.2, n.2, p.186–195, 2020.

ZHU, C.; HUANG, S.; EVANS, R.; ZHANG, W. Cyberbullying among adolescents and children: a comprehensive review of the global situation, risk factors, and preventive measures. **Frontiers in public health**, [S.l.], v.9, p.634909, 2021.

ZINGALE, N. C. The phenomenology of sharing: Social media networking, asserting, and telling. **Journal of Public Affairs**, [S.l.], v.13, n.3, p.288–297, 2013.

Appendices

APPENDIX A – Online Form for the Evaluation of the KSA Graph on LoA

Knowledge, Skills and Abilities for Layers of Abstraction

LoA = Layers of Abstraction
KSA = Knowledge, Skills and Abilities

* Indica uma pergunta obrigatória

1. E-mail *

Free and Informed Consent (IC) Form

You are being invited to participate in a research project. After reading and clarifying the following information, if you agree to take part in the study, give your consent by checking the YES option in the following question. To continue enrolling in the activity, you must agree to the **Informed Consent (IC)**.

ExpPC Project - Exploring Computational Thinking for Elementary Education Qualification
Responsible researcher: Prof. Dr. Simone André da Costa Cavalheiro
Phone: (53) 3284 3860
Email: exppc@inf.ufpel.edu.br

This project aims to create an educational network to consolidate Computational Thinking in the educational context, which is being developed by teachers and students of Computing courses at the Federal University of Pelotas.

By participating in this project, you will attend a review activity that deals with Computational Thinking, including didactic-pedagogical activities that address Computing concepts, which may be filmed, recorded, observed and evaluated by researchers who make up the project team. In addition to the activities, you may also receive a socioeconomic and cultural questionnaire, as well as an evaluation form. You are free to refuse to participate and to withdraw from participation at any time without any prejudice. However, we request your collaboration so that we can obtain better results in the project. Whenever you want more information, you can contact the responsible researcher directly. Participation in this project does not bring any legal complications of any kind and the procedures used comply with the criteria of ethics in Research with Human Beings. The study presents minimal risks, as completing the socioeconomic questionnaire, when requested, may cause embarrassment to participants and may be interrupted at any time. By using virtual environments and online forms to collect personal data (name, email, place of study or work), filming and recordings, another risk that you may run when carrying out the research is to have the confidentiality of this content violated, even with all the confidentiality precautions that will be adopted. By participating in this research, you will have the opportunity to deepen your knowledge of the Computational Thinking methodology. You will not incur any type of expense for participating in this study, nor will you receive any type of payment for your participation.
By giving your consent to this term, it will be considered consent when responding to other course forms and questionnaires. If you wish to withdraw your consent to the use of your data for research, this can be done at any time and without prejudice, by sending a request addressed to the email address identified above.

After these clarifications, we request your free consent to participate in this research. To do so, answer the following question:

2. **FREE AND INFORMED CONSENT**: in view of the items presented above, I, in a free and informed manner, agree to participate in this project. *

Marcar apenas uma oval.

- ◯ Yes, I consent to the use of my data for research
- ◯ No, I do not consent to the use of your data for research

Personal Data

3. Name *

_____

4. Alternative e-mail

_____

5. Institution (place of study or work)

_____

6. Field of expertise

_____

7. Experience with computing education practice (teaching) *

Marcar apenas uma oval.

- ◯ No experience
- ◯ Less than 1 year
- ◯ Between 1 and 5 years
- ◯ Between 5 and 10 years
- ◯ More than 10 years

8. Experience with computing education theory (research, including, but not limited to empirical experiments) *

Marcar apenas uma oval.

- ◯ No experience
- ◯ Less than 1 year
- ◯ Between 1 and 5 years
- ◯ Between 5 and 10 years
- ◯ More than 10 years

9. In which of the **International Standard Classification of Education** (**ISCED**) levels of education do you have experience? *

Marque todas que se aplicam.

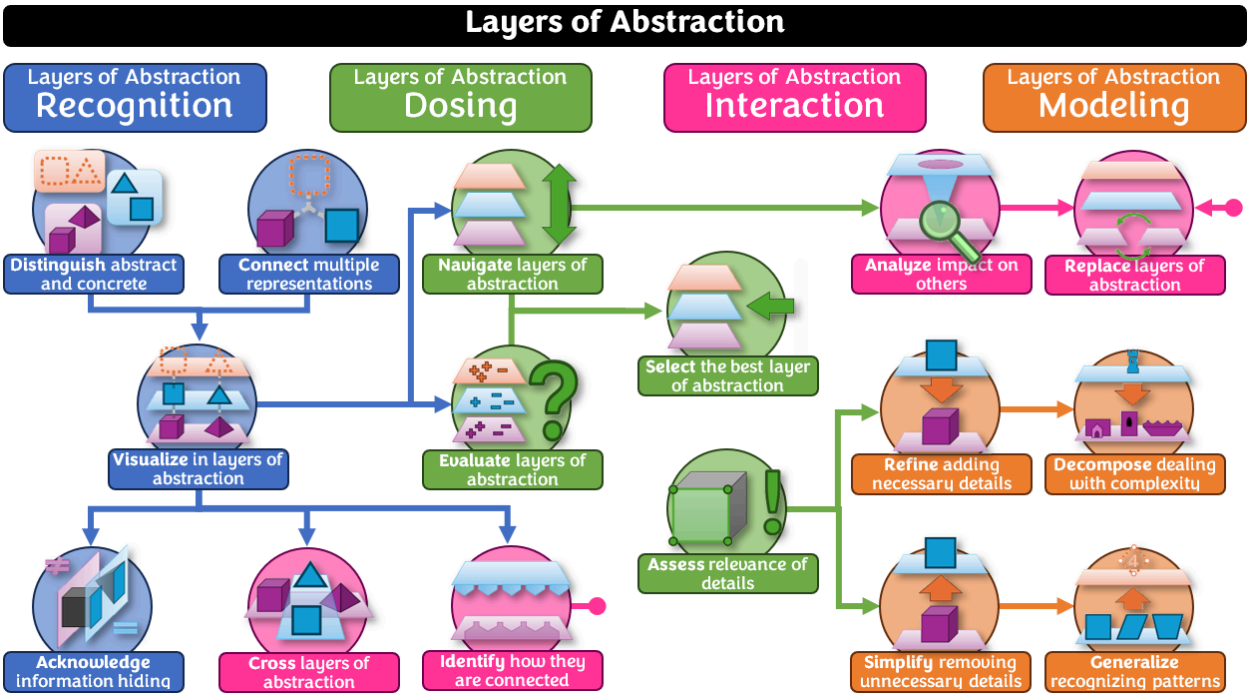
- ☐ ISCED Level 0 - Early Childhood Education (Pre-K) Includes creches and pre-schools [Approximate age: >5]
- ☐ ISCED Level 1 - Primary Education (K-5) Includes primary and elementary schools [Approximate age: 5-10]
- ☐ ISCED Level 2 - Lower Secondary Education (Grades 6-9) Includes middle schools [Approximate age: 11-14]
- ☐ ISCED Level 3 - Upper Secondary Education (Grades 10-12) Includes high schools [Approximate age: 15-17]
- ☐ ISCED Level 4 - Technical Education (Post-Secondary non-Tertiary) Includes vocational and trade schools [Approximate age: 18+]
- ☐ ISCED Level 5+ - Tertiary Education (Higher Education) Includes universities [Approximate age: 18+]

10. How do you want your contribution to be acknowledged in the thesis and any other publications involving this work? *

Marcar apenas uma oval.

- ◯ I want to stay anonymous
- ◯ I want to be credited by name
- ◯ I want to be credited by name and institution (when possible)
- ◯ I want to be credited by name, e-mail and institution (when possible)

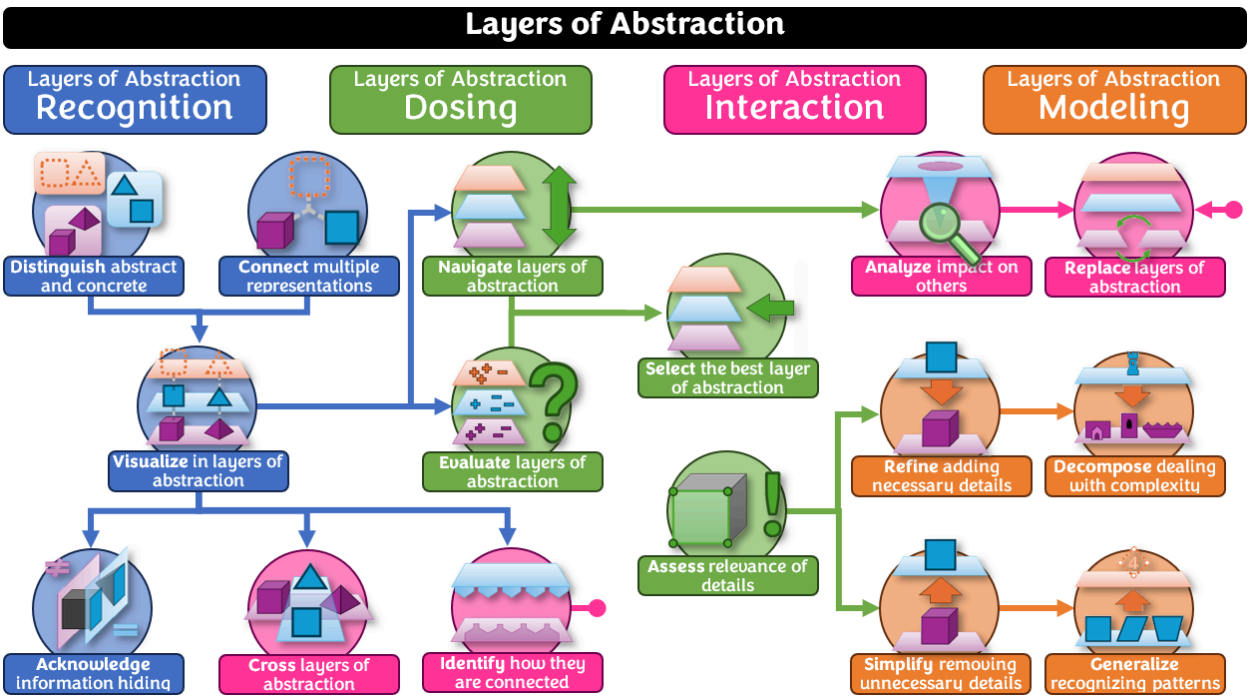
Seção sem título

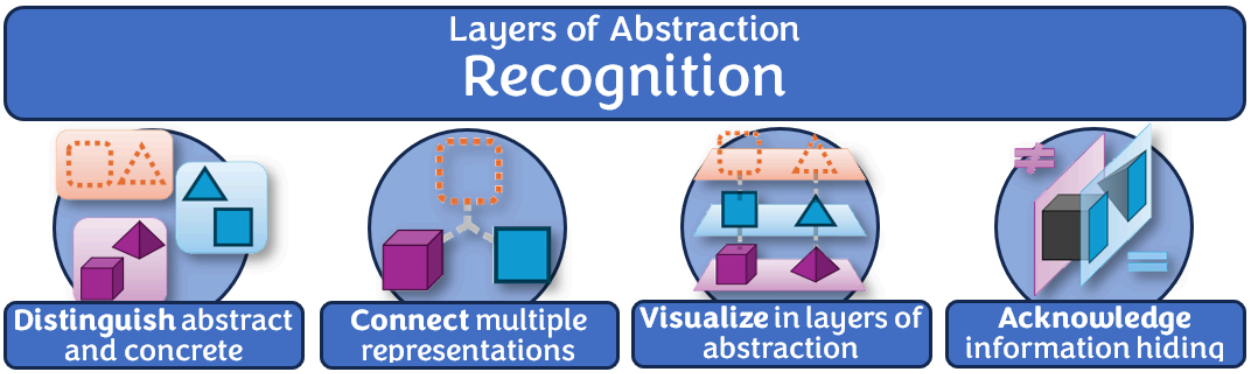

Layers of Abstraction
Layers of Abstraction Recognition
Layers of Abstraction Dosing
Layers of Abstraction Interaction
Layers of Abstraction Modeling
Distinguish abstract and concrete
Connect multiple representations
Navigate layers of abstraction
Analyze impact on others
Replace layers of abstraction
Visualize in layers of abstraction
Evaluate layers of abstraction
Select the best layer of abstraction
Refine adding necessary details
Decompose dealing with complexity
Assess relevance of details
Acknowledge information hiding
Cross layers of abstraction
Identify how they are connected
Simplify removing unnecessary details
Generalize recognizing patterns

This **Knowledge, Skills, and Abilities (KSA)** graph is under development rooted in **Layers of Abstraction (LoA)**, this shall serve as a guiding framework for designing learning objectives, educational activities and assessments. Consider the following definition for LoA: ``the hierarchical organization of complex systems or problems into multiple levels of representation, with each level hiding unnecessary details and exposing only the essential elements needed for a particular purpose" (Silva Junior, 2023). We are targeting LoA because we consider they an iconic manifestation of **Abstraction** in computing. In turn, Abstraction is considered a cornerstone of **Computational Thinking (CT).** Proposed as a mental process involving a set of problem-solving skills based on computing, or simply "thinking like a computer scientist" (Wing, 2006), CT quickly gained attention in the literature and became an important advocate for the introduction of computing in compulsory education (k-12).

Your insights and evaluation of this graph are crucial in refining its efficacy and applicability. Thank you for your contribution to advancing computing education. You are asked to assess its quality by leaving commentaries, suggestions and classifying it as: **Unacceptable (1)**, **Limited (2)**, **Adequate (3)**, or **Ideal (4)**, for each of the following criteria:

> Clarity and Structure: How clearly are the competencies defined? Is the structure of the KSA graph logical and intuitive? Are the relationships between different competencies evident?
> Coverage: Does the KSA graph cover a broad range of essential skills and abilities in computing related to abstraction? Are there any major gaps or redundancies in the coverage?
> Granularity: Are the competencies appropriately distributed in terms of Granularity? Should one be broken down into more, or various merged into one?
> Relevance and Alignment: How well do the competencies align with the learning objectives for students in the computing domain? To what extent does the graph reflect the KSA required in industry, education and research settings? Are there opportunities for students to apply these skills in real-world projects?
> interdisciplinarity: Does the KSA graph give space to incorporate relevant interdisciplinary connections with other fields? Are there opportunities for students to apply computing skills in diverse contexts?





The first KSA then refers to the general comprehension of what is a LoA and the acknowledgement of their existence (or absence) in a given system/problem.

> **[Distinguish abstract and concrete]** Realizing that things may have representations with different amount of details. Being able to tell if something is closer to an idea (more abstract) or the reality (more concrete). Grouping things with similar level of detail, or classifying which items belong to each group.
Examples: Distinguishing between software (abstract) and hardware (concrete); or a map (abstract) and the physical terrain it represents (concrete).

> **[Connect multiple representations]** Understanding that a single thing can be represented in different ways, that is, have multiple representations. Linking multiple representations to a single thing. Identifying all representations that can refer to the same thing.
Examples: Connecting a flowchart, a pseudo-code and a visual programming language to the same algorithm; or a real world phenomenon, an equation, and a computer simulation to the same chemical reaction.

> **[Visualize in LoA]** Stratify a system into different views upon the same elements, establishing an hierarchy over the abstraction of their various representations. The higher levels are those abstracting more, thus with less details. The lower we go, the more details there will be.
Examples: Visualizing living beings through kingdoms, classes, families, species and food-chain roles, with each individual consistently dispersed across the levels of organization.

> **[Acknowledge information hiding]** Demonstrating awareness about the fact that we hide/lose information when abstracting. Comprehending that things may be different at lower levels while being the same at higher levels.
Examples: Acknowledging that the summary of a book can tell me the story very quickly, but there will be lots of details I would not know just by reading it; or that, when seeing two games as just "racing games", they may still be different despite having the same general goals and mechanics.

11. Regarding **Clarity and Structure**, this is: *

Marcar apenas uma oval.

- (1) Unacceptable: Competencies lack clear definitions, making it challenging for stakeholders to understand their meaning. The overall structure lacks logic and intuition. For example, competency descriptions are vague and unclear, leaving the reader without a grasp of what it is about. Even experts can't tell what they mean.
- (2) Limited: Competencies are somewhat defined, but improvements are needed for clarity. The structure is somewhat logical, but some areas may cause confusion. For instance, some competency descriptions employ words with unusual meanings or lack specificity, causing ambiguity. Experts may understand, but it is unlikely others would.
- (3) Adequate: Competencies are clearly defined, and the overall structure is logical and intuitive. It doesn't seem to be missing anything important, but it could still benefit from additional information for further clarification. For example, competency descriptions are clear, but a few could be more explicit to enhance understanding. It is likely to be understood by non-specialists, as long as they have some education or computing background.
- (4) Ideal: Competencies are very well defined, and the structure is highly logical and intuitive. For instance, competency descriptions are precise and insightful, providing an advanced level of clarity. It is easy to understand it without any prior knowledge of the involved areas.

12. Unless you think it is already Ideal, please comment what you think that is missing, what you disagree or what you suggest to improve it. Feel free to leave notes or any other information you may see fit (even if you think it is ideal).

13. Regarding **Coverage**, this is: *

Marcar apenas uma oval.

- (1) Unacceptable: The KSA graph lacks essential skills and abilities in computing related to abstraction, leading to a fragmented understanding. Major gaps or redundancies hinder a comprehensive view. For example, key concepts in abstraction are missing, impacting the graph's completeness.
- (2) Limited: The coverage is somewhat limited, with notable gaps or redundancies. Some essential skills and abilities are missing or duplicated, affecting overall comprehensiveness. For instance, foundational concepts are present, but certain advanced topics are missing.
- (3) Adequate: The KSA graph covers a broad range of essential skills and abilities in computing related to abstraction. While there may be minor gaps or redundancies, they do not significantly impact overall coverage. For example, most critical concepts are covered, but a few specialized areas might be underrepresented.
- (4) Ideal: The coverage is comprehensive, encompassing a wide array of essential skills and abilities. The graph includes a thorough representation of both foundational and specialized concepts in abstraction. For instance, the reader can't think of an abstraction competency that isn't covered by the graph.

14. Unless you think it is already Ideal, please comment what you think that is missing, what you disagree or what you suggest to improve it. Feel free to leave notes or any other information you may see fit (even if you think it is ideal).

15. Regarding **Granularity**, this is: *

Marcar apenas uma oval.

- (1) Unacceptable: Competencies are inappropriately distributed in terms of granularity, causing confusion. Some are overly detailed, while others lack specificity. For example, certain competencies are excessively broad, making it hard to discern their practical application.
- (2) Limited: Most competencies exhibit consistent granularity, but there are outliers being overly detailed or too broad. For instance, some competencies should be merged or broken down to become coherent with the others.
- (3) Adequate: Competencies are appropriately distributed in terms of granularity, with reasonable coherence. Competencies strike a good balance between detail and breadth, but there might be minor adjustments needed. For example, some competencies might need to be slightly more or less restricted to be on aligned with the rest.
- (4) Ideal: Competencies are well-balanced in granularity. There is an advanced level of precision in how competencies were distributed. For instance, the granularity is not only balanced but also insightful, contributing to the creation of well-defined tasks and operationalization of the competencies.

16. Unless you think it is already Ideal, please comment what you think that is missing, what you disagree or what you suggest to improve it. Feel free to leave notes or any other information you may see fit (even if you think it is ideal).

17. Regarding **Relevance and Alignment**, this is: *

Marcar apenas uma oval.

- (1) Unacceptable: Competencies are poorly aligned with learning objectives, and there is a significant mismatch with industry, education, and research settings. Opportunities for real-world application are lacking. For example, the graph does not reflect current industry demands or relevant learning objectives, going on a different direction of the currently established guidelines.
- (2) Limited: Competencies are somewhat aligned with learning objectives, but important ones are missing. The graph does not fully reflect the KSA required in industry, education, and research settings. Limited opportunities for real-world application are present. For instance, a given competency is an explicit demand in oficial documents, but there is no trace of it in the proposal.
- (3) Adequate: Competencies align reasonably well with learning objectives, reflecting a connection to the KSA required in industry, education, and research settings. Reasonable opportunities for real-world application are present, but others could be incorporated. For example, the graph aligns with key learning objectives, but certain industry-specific skills could be more pronounced.
- (4) Ideal: Competencies align well with learning objectives, reflecting a strong connection to the KSA required in industry, education, and research settings. Opportunities for real-world application are well-integrated. For instance, the graph not only aligns with learning objectives but also showcases an advanced level of integration with industry demands and real-world scenarios.

18. Unless you think it is already Ideal, please comment what you think that is missing, what you disagree or what you suggest to improve it. Feel free to leave notes or any other information you may see fit (even if you think it is ideal).

19. Regarding **Interdisciplinarity**, this is: *

Marcar apenas uma oval.

- (1) Unacceptable: The KSA graph hinders interdisciplinary connections with other fields entirely, limiting its applicability. There are no opportunities for students to apply computing skills in diverse contexts. For example, the competencies are super-specialized and have their application bound to computing alone.
- (2) Limited: Interdisciplinary connections are conceivable, but impracticable. There are limited opportunities for students to apply computing skills in diverse contexts. For instance, there are theoretical possibilities to integrate interdisciplinary aspects, but they are challenging to approach or far-fetched.
- (3) Adequate: The KSA graph provides some space for incorporating interdisciplinary connections with other fields. There are modest opportunities for students to apply computing skills in diverse contexts. For example, activities fostering the graph competecies while approaching curricular content of other fields are feasible.
- (4) Ideal: Interdisciplinary connections can be easily integrated into the KSA graph. Opportunities for students to apply computing skills in diverse contexts are noticeable and naturally promoted. For instance, when a competency is read, example scenarios from other fields automatically comes to your mind.

20. Unless you think it is already Ideal, please comment what you think that is missing, what you disagree or what you suggest to improve it. Feel free to leave notes or any other information you may see fit (even if you think it is ideal).

APPENDIX B – Cognitive Interview Transcription - Subject 1

Interviewer: Come on! It's all going to have to be more or less, because you didn't let me finish!

Student: But also, you 're taking too long.

Interviewer: I'm taking my time.

Student: And I can't wait and there's no one to play with me.

Interviewer: Let me fix something here real quick.

Interviewer: If I had fixed it, it wouldn't be wrong.

Student: What game is this?

Interviewer: It's the land of abstraction. I'll explain it to you in a moment, just wait.

Student: What is abstraction?

Interviewer: Abstraction is thinking about things in a simpler way, without worrying about the details. Remember what I told you? It's like when we make a quick drawing, with only the essentials to convey the idea, you know?

Student: I don't do that! Just like here in [the character the student had drawn at an earlier point], I did every little detail.

Interviewer: Okay, but when you're going to draw [the student's dog], for example, you don't even draw the little flea that's hidden under its fur. You only draw the main details.

Student: Baby (calls her dog)! Baby will help me! Come on baby! Baby will help me play. Why are you recording?

Interviewer: When we worry more about the details, then we have something less abstract, which is more concrete, like the baby. Look at all the details she has! If you had to draw her with all the details, you would have to draw every little hair on her.

Student: Haha, it would take me a year! And why are you recording?

Interviewer: It would take a year! Because I want to see it later. I need to make you answer a few things. Well, anyway, the more you... at least you draw, of [the student's dog], the more concrete it becomes. Why? Because it becomes more similar to reality, you understand?

Student: Wow! Can you create an automatic petting machine for [student's dog]?

Interviewer: No. So let's go.

Student: It's just that she kills my hand asking for affection.

Interviewer: Okay, this is the land of abstraction, a place with a lot of characters.

Student: Is that a character? I thought it was a cardboard cutout!

Interviewer: They are characters. And here is a princess... Wait, let me enlarge it here...

Student: Princess? That's a bride!

Interviewer: It's a princess dressed as a bride. And there are little animals...

Student: So it's marriage?

Interviewer: No, right? She's just a princess. And she has little animals and superheroes.

Student: I think you should have put the picture of [the student's dog]. It's much prettier and cuter!

Interviewer: Hmm.

Student: On the pet screen, there is the super heroine [the student's dog] and the dog princess!

Interviewer: Leave her on the floor, otherwise she will be screaming.

Student: She doesn't scream, she growls!

Interviewer: Each of the characters is different.

Student: Uh-huh, and?

Interviewer: Can you tell which one of these is more abstract?

Student: What do you mean abs... more abstract?

Interviewer: What has less detail, and what has more detail?

Student: [mermaid princess]!

Interviewer: Hmm, why is she the one with the most details?

Student: She has the hair details, right?

Interviewer: Hmm, and why is this the one with the least detail?

Student: Because there isn't any.

Interviewer: Okay, so let's get to the game. In the land of abstraction we use this abstraction meter here. The higher it is, the less detail. The lower it is, the more detail.

Interviewer: Can you group these by abstraction?

Student: What do you mean by grouping by abstraction?

Interviewer: Those with more details and those with fewer details.

Student: I think so. Here, fewer details. This one and this one, do I have to click?

Interviewer: Uh-huh, drag it over there. Leave it together, uh-huh, anywhere.

Student: [mermaid princess] and the [fish], the puppy and the superhero.

Interviewer: That's right. And why can't it be like that?

Student: Ah... Because this one... Because this one doesn't have many details. Now the [fish]... It even has the details of the eye, the mouth, the scales, the little balls.

Interviewer: Uh-huh, it's much more detailed than this one. Okay, so that means these characters here aren't all the same?

Student: They are!

Interviewer: No, they are not. Why? Because when you add more details, you see that they are already different. When you see even more details, they are even more different. So, in fact, when you change the meter there, you are not showing new characters, you are showing the same ones, just from a different view.

Student: Uh-huh, that's what I was going to say!

Interviewer: Hmm, so look, the [swamp princess] here is the same [swamp princess] here, only she's just a princess. And here she's just a character, you know?

Student: But that's the last thing...

Interviewer: It's to see everything at the same time.

Student: Oh there!

Interviewer: So let's do the following. The first game will be the following. I'm going to choose a character. I'm going to choose [fish]. But then I'm going to change the level...

Student: And you chose the little animal.

Interviewer: And I'll shuffle it, and you won't know who's the [fish] anymore. [...] Who 's the [fish]?

Student: It's the little animals here.

Interviewer: But which one?

Student: And I think this one.

Interviewer: This one? Oh, you got it! So let's go again. I'll pick [the green-suited superhero]. The [green-suited superhero]... and now I'm going to shuffle...

Interviewer: Who is [green uniform superhero]?

Student: That!

Interviewer: Ahh! Okay, so the last one...

Student: I'm not very good at memory, but...

Interviewer: But you're doing great. The last one is this one, the [firefighter puppy]. Do you know who the [firefighter puppy] is?

Student: Yes.

Interviewer: Hmm, so let's shuffle.

Interviewer: Who is the [firefighter puppy]?

Student: I'm not sure if it's this one or this one? I know it's one of these two.

Interviewer: Okay, but... you have to just... just one.

Student: I know, but I'm thinking.

Interviewer: Hmm... three... two... one and...

Student: I think this is it.

Interviewer: This one? Aaai, I think that was the one you were unsure about. Okay , but what I want to know...

Student: But it's almost.

Interviewer: How do you know, when I go to this one, who's who?

Student: Like this?

Interviewer: How did you know this was the [firefighter puppy]? If you can't see him here?

Student: You know what it is? Come... go back there. It's just that when you go... go for another round, I memorize the animal's position. Like, the [firefighter dog] was there, so...

Interviewer: So when you go here and stay there, it's just in a different way, and then when you shuffle, you just keep following it.

Student: Yes! Yeah, but you just mixed them up together, so...

Interviewer: Now it got difficult. Oh, okay then! Let's move on to the next one. Oh, okay. Now we need to do the following. The Queen of Hearts is going to throw a party in the kingdom. The kingdom is here. But she will only throw this party if when she arrives everyone is well organized. And how do they have to be organized? The animals have to be on one side, the princesses on another and the superheroes on another.

Student: Yes, I was going to say that. Yes, that's what I was going to say!

Interviewer: How can you organize here?

Student: They already... Will I have to organize without seeing?

Interviewer: You can change the meter. Which meter do you want to use?

Student: I want to use this one in more detail.

Interviewer: This one or that one?,

Student: That!

Interviewer: Why this one and not this one?

Student: Because that one is easier to understand the characters, but this one is also pretty easy! I don't know which one I choose.

Interviewer: Just organize the princesses on one side...

Student: I know, but I think this one, because in this one those guys look more like a bride than a princess.

Interviewer: Oh, okay, now just organize it. Drag it to the side.

Student: Yes, but I'll leave it in good order.

Interviewer: It is fine.

Student: Hey! Let go! I didn't choose you, I chose the [snow princess].

Interviewer: Uh-huh.

Student: Let's pretend there are three rows of tables. So I'm organizing them here, look. One, two, three, one, two, three. And there's only one left here. One, two, three.

Interviewer: Very good! Now everyone is organized. Then the queen won't be able to complain.

Student: There is space here.

Interviewer: And is it organized like I asked? I asked to organize the dogs... the little animals one way, the princesses another and the...

Student: And what about them?

Interviewer: And the superheroes on the other side. Did you do it?

Student: Yes!

Interviewer: But wouldn't it have been easier if you had organized it here? Because here, look, you already know that they are little animals, superheroes and princesses.

Student: Yes, but she looks more like a bride than a princess.

Interviewer: Okay, but what if you didn't know? If you didn't know that [the superhero in the blue uniform] is a superhero , for example.

Student: But I would know because he's wearing a cape.

Interviewer: Hmm, but the [red-suited superhero] doesn't wear a cape. How do you know he's a superhero ?

Student: But he's jumping. And the [green-suited superhero] looks like a villain... but I know him.

Interviewer: Okay, but you did it. Now... now I...

Student: And I would think that too, if I didn't know they were superheroes. Here are the animals, which I know are animals. Here are the princesses, which I know are princesses. And here there can only be these left!

Interviewer: But what if there were only two superheroes and four princesses? There's no way of knowing.

Interviewer: It would have been easier to do this one, because it already says here that they are superheroes, princesses and stuff.

Student: BRIDES and business.

Interviewer: Okay, brides and...

Student: She looks more like a bride than a princess.

Interviewer: But okay, you've completed it. Now you have to organize them so they can go buy clothes and they have to buy clothes according to their main color. How are you going to organize it?

Student: The bride in white.

Interviewer: No, but each character has its own color.

Student: Okay? So put it there, right?

Interviewer: Oh, so it's not possible to organize this?

Student: No.

Interviewer: It has to be more detailed. So go ahead and organize it.

Student: [mermaid princess] is red. Everyone here is red. And [the superhero in the red uniform] is also red. [the snow princess] is blue. Me [the superhero in the blue

uniform] too, and [the fish] too. Oh, and [the firefighter puppy] is also red. And there are only these three left, so of course they are green. This phone has already rung.

Interviewer: Now yes!

Student: And it's not recording anymore.

Interviewer: Oh, now you've done it, all right. Now here's the thing, I didn't finish this, so we'll have to keep coming back here. The Queen gave each person an invitation , so the invitations are in pairs. The [firefighter puppy] got an invitation with the right to a companion. But the [firefighter puppy]'s companion can only be a little animal. The [green lizard] can only be accompanied by a superhero. And the [fish] can only be accompanied by a princess.

Student: And a princess's princess.

Interviewer: So, each one received an invitation from a king, so you need to group them together according to the invitations. First, choose an invitation.

Interviewer: Calm down, choose an invitation here, just for you to see. It should be here, but I'm not...

Student: But I remember. I remember only one.

Interviewer: What was it? It was the [mermaid princess] and what?

Student: And a little animal.

Interviewer: And a little animal, so get a little animal.

Student: Hmm, let me choose. I'll get the [fish], because it's like a princess, look from here.

Interviewer: So wait, so you got the [fish] with the princess and the [mermaid princess] with the animal?

Student: I've already done two.

Interviewer: Very well then. And that's it. Now you can choose another one.

Student: Okay.

Interviewer: Who are you going to do?

Student: I'll start with the princesses, and here in order is [swamp princess].

Interviewer: So [swamp princess] has to go with?

Student: The [green uniform superhero].

Interviewer: A superhero.

Student: But I already told her who she's going with. They're both green.

Interviewer: So they go together in green.

Student: And yes, they are both green.

Interviewer: But these ones here don't match as clothes, for example.

Student: Oh, but they got together, right? One with the other, the princess with an animal and the animal with the princess.

Interviewer: Yes, and they are both from the water.

Student: Yes, I click here?

Interviewer: Click here on the side. That's it. Now who else are you going to do?

Student: Oh, let me just see who the [superhero in the green uniform] was with. Oh, the [superhero in the green uniform] was with a princess!

Interviewer: Okay then! You've already made two again.

Student: And I hadn't even realized it.

Interviewer: Now you have to do one last thing.

Student: Okay, the one from Frozen.

Interviewer: Come back! Frozen is like a princess.

Student: With a princess. But there was no princess left for her!

Interviewer: So it can't be her, it has to be someone else.

Student: Like this?

Interviewer: You have to see who is left and see the invitation of who is still left.

Student: But no one was left of the princess.

Interviewer: No, then, but the princesses have already been invited, they've already been there in other ways, you understand? So now you go back there ... Here. See who was left out and see the invitations of these people.

Student: But she's not going? To the party?

Interviewer: Yes, everyone got an invitation. You have to see what the invitations are.

Student: Okay , so let me go back... Ah, it's here.

Student: But no one is inviting princess!

Interviewer: The [green uniform superhero] and the [fish] invite princesses.

Student: Yes, but [fish] has already invited [mermaid princess].

Interviewer: Then you'll have to change.

Student: And the [green uniform superhero] has already invited the [swamp princess]!

Interviewer: Then you'll have to change.

Student: But here, look! It's a perfect match!

Interviewer: But if you don't change, she'll stay... The [snow princess] will be left out. Go do the others then. Leave the [snow princess] aside for now.

Student: For now!

Interviewer: Here, let's do the [red uniform superhero] then.

Interviewer: The [red uniform superhero]... Is it with a little animal?

Student: Yes.

Interviewer: So he goes with the [green lizard]. Okay, what about the [firefighter puppy]?

Student: But the [superhero in the blue uniform]... he's like a superhero!

Interviewer: Then you can go with the [red uniform superhero]!

Student: But the [red uniform superhero] has a pet!

Interviewer: No, but look. The [superhero in the red uniform], he has his own invitation, which is with the little animal and he can invite someone if he uses the invitation. But if he is invited by the [superhero in the blue uniform], he doesn't need to use his invitation, understand?

Student: But then where will the little animal stay?

Interviewer: The pet has to be invited by someone else, use their invitation.

Interviewer: Look, if you do it like this, then you'll be missing these.

Student: Oh, let me see.

Interviewer: Who else? We're missing the [green lizard], the [firefighter puppy]...

Student: But then it would be like this and the [snow princess]?

Interviewer: The [Snow Princess] was left out.

Student: Then.

Interviewer: But there always has to be one left.

Interviewer: The [snow princess] will be the star, so she doesn't need anyone, she will be on stage, she will be the one who will be, who will sing, then all the others will dance in pairs and then she will stay to sing.

Student: Hmm! So let's go to the party.

Interviewer: And that.

Student: But you... oh yeah. So let's go.

Interviewer: No, no, no, no, leave it there.

Student: I'm just taking it.

Interviewer: But you already did it.

Student: Let me get it here. Here look, they're all heading to the party.

Interviewer: That's it, so you did it! The [snow princess] stayed to sing and... let's check it out. The [firefighter puppy] has an invitation with a little animal on it. Is this a little animal?

Student: Uh-huh!

Interviewer: It's a little animal, so it's okay. [Superhero in the blue uniform] goes with a superhero. [Superhero in the red uniform] is a superhero. So it's okay. And [Swamp Princess] goes with a superhero. [Superhero in the green uniform] is a superhero. So it's okay. And [Fish] goes as a princess. So it's okay. Okay, you've moved on to the next level!

Interviewer: Now. Oh, this one won't work...

Student: Why?

Interviewer: Why would I have to do something I couldn't do? But let's skip ahead.

Student: What would it be?

Interviewer: Let's skip to the second game, you got this one.

Interviewer: I completed the second game.

Student: Why did you jump?

Interviewer: Because I had to make a plan and there wasn't enough time. So let's go to the second game, which is the second game for you to take care of a pet.

Student: Woohoo!

Interviewer: Do you know how to take care of a pet?

Student: Uhum! I have a pet!

Interviewer: When he's hungry...

Student: You have to give him food. When he's thirsty, you have to give him water because it's dirty, you have to bathe him. If he's sleepy, you have to put him to sleep.

Interviewer: It appears that he is hungry. Then you have to choose what to do: feed him, bathe him or play with him.

Student: Is that the animal? I thought a... puppy would appear in the picture.

Interviewer: That's the beast.

Student: It doesn't look like a dog in the picture, no other animal appears...?

Interviewer: What do you mean in a picture?

Student: I'm talking off the page, like the [snail], which we made the other one.

Interviewer: Like this?

Student: Like, look, let me take [the student's dog] as an example.

Student: Let's take this as an example. Pretend that this is a computer screen. The little animal won't appear here in a figure, like this is a square figure. No! The little animal will appear alone...

Interviewer: OK I understand.

Student: Cut out of paper, talking, jumping, shaking, trembling.

Interviewer: Got it. No, only the image will appear.

Student: This guy doesn't even blink! What are you staring at?

Interviewer: Okay, now you have to take care of him.

Student: Okay, then give it to me.

Interviewer: You have to click here and choose between giving food... Then you have to click there... That's it!

Student: Oh, there's no way to even choose the food?

Interviewer: No, that's just for you to learn.

Student: And he keeps asking for food. And what's that?

Interviewer: That's because he's dirty. He pooped. You have to give him a bath.

Student: Here at Clean?

Interviewer: That's it! He's still dirty , give him another bath!

Student: Oh my God, he's hungry, he's sleepy, he's dirty!

Interviewer: It's because you 're taking too long and it's going to... Look... it's already dirty again!

Student: Hmmph!

Interviewer: There, it's clean! He's hungry. He's still hungry!

Interviewer: Oh! Now he's just sad, you have to play with him.

Student: I know. Look! It's already appeared again!

Interviewer: It will work.

Student: He didn't even leave the house!

Interviewer: Oh my god, it's so fast!

Student: And you only see this now?

Interviewer: There, finally. Only now he's joined the others.

Interviewer: It's back together, oh my god!

Student: This game is impossible!

Interviewer: Okay, I'll slow it down.

Student: Look. I'll get confused in a little while.

Interviewer: When you're taking a bath...

Student: Look, he came in again!

Interviewer: You can continue until it's clean. No, but then when it's something else, you have to change it. Now there's only one left. Go, quick!

Interviewer: No, you got it wrong.

Student: But I clicked.

Interviewer: You clicked on bathe, there it is for when he is sad. Here it is for when he is dirty.

Interviewer: Go, go. Until the sadness ends.

Student: Oh my God!

Interviewer: Come on, there's just one thing left! Take a bath now. Aaah, now that's all that's left.

Interviewer: No!

Student: Look! This game is impossible!

Interviewer: You did it! You won!

Student: He's already sad again.

Interviewer: No, but you won! It's just that I didn't finish it so when they're all done you win the game. Okay , but did you know that it's not that easy to play...

Student: For sure!

Interviewer: Yes. It's not that easy to take care of a pet.

Student: It's harder in the game than in real life!

Interviewer: No, actually, when you have to give food, giving food is not that easy, right? How do you give food?

Student: Just put a bowl of food and go.

Interviewer: But to put a bowl of food in, you have to raise your arm, put your hand on top of the bowl, close your hand, lift the bowl, bring it here...

Interviewer: See? Something so simple that you talk about how to give food? It can have so many details that you can't even stand to hear it because it's so boring.

Well, you don't need to explain it that much either! No one is that dumb... So how would you explain how to give food in five steps?

Student: I would say, go to the pot, lift the lid of the pot, get the food, put the food in the pot, and put the pot on the floor.

Interviewer: That's it. So in just one step, which was to give food, now you've explained it much better... and without having to explain a million things that no one wants to know. Well, how would you explain giving a bath?

Student: Take a bath.

Interviewer: In five steps.

Student: In five steps. I would say: pick up the dog, take him to... oh, wait... pick up the dog and take him to the shower.

Interviewer: Okay, that's just one.

Student: Yeah, uh-huh. Wet... I mean, grab the towel, grab the blow dryer.

Interviewer: Get the things from the bathroom.

Student: Yeah, get the bath stuff. Hmm... Wet the pet and... turn on the shower, wet him.

Interviewer: Okay, only two left!

Student: So... Two? But... Oh, right! I would now lather it up, rinse it off and dry it!

Interviewer: No... You're going to have to change this one a little bit. You started by explaining too much and ended up explaining too little. You have to explain everything in the same level of detail.

Student: Okay, okay!

Interviewer: So let's go. First?

Student: Pick up the pet and take him to the shower. Get the bath supplies.

Interviewer: Couldn't you get everything together? Because then you can use the four left over for the bath itself. So, prepare the bath, right? Get the pet and the things.

Student: Turn on the tap and wet the pet. Then soap the pet.

Interviewer: Soap the pet.

Student: Dry it.

Interviewer: Rinse, right?

Student: Yeah, rinse it. And... dry it.

Interviewer: There you go. Now perfect!

Student: But there's one thing missing! Comb her hair and make a ponytail. Put on a bow and put on a matching necklace and make a bracelet to wear! Put on shoes and clothes! [The student's dog] would be so cute if every time I put up a decoration she went there, scraped herself off the wall and took it off.

Interviewer: But there are so many things, we just want to take a bath.

Student: And whenever I show up with a dress in my hand, she goes under the table and growls when I try to get close.

Interviewer: Okay? What now? How do you explain playing in five steps?

Student: It's quite easy.

Student: Go outside with the dog.

Interviewer: One step.

Student: So what... and... wait .

Interviewer: You should have picked up a toy.

Student: I forgot! I forgot the Toy! Now let's go... again.

Interviewer: First step.

Student: Ahaha, I remember! I'm forgetful.

Interviewer: Come on, I'll put it here then, so I don't forget.

Student: First step.

Interviewer: Get the toy.

Student: I'll say! Get the toy. Second step is to take the pet outside to.

Student: The third step is to pick up the toys and play with them.

Interviewer: But you've already picked up the toy and playing with it is what you're explaining! Let's first think about what kind of game it's going to be! How do you play with it?

Student: I know.

Interviewer: Hmm?

Student: Give one of the toys to your pet.

Interviewer: Give one of the toys to the pet. Okay, so?

Student: Run after him like crazy.

Interviewer: Run after him like crazy.

Student: He'll start running and it'll look like you're playing tag.

Interviewer: Okay, but what about finally, after you run after him like crazy?

Student: Finally, take...

Interviewer: Put away your toys, right? You can't leave them lying around!

Student: I was going to say! But we went outside , I was going to get the toys and go back home, put them away, go in and get the dog too.

Interviewer: You can't forget the dog on the street!

Student: But that goes together!

Interviewer: But is this the only way to play with your dog?

Student: No. That's just a way for me to play with him.

Interviewer: And couldn't you come up with another joke?

Student: Yes!

Interviewer: So let's go...

Student: Throw the stick for him to fetch and then run away from him so he...

Interviewer: No, let's go step by step! Okay, first step then. Get a...

Student: Little stick.

Interviewer: Second step?

Student: It's... you know what it is? What's that thing where we keep saying, "catch, catch", and then we throw it?

Interviewer: Hum, call attention!

Student: With a toy!

Interviewer: Getting attention with a toy. And after getting attention?

Student: Throw the stick.

Interviewer: Throw the stick And then?

Student: Um, tell him to give it. But if... I mean... ask him to bring the toy.

Interviewer: Wait for him to bring it?

Student: But if you wait for my dog to bring it, you'll stay there until... You'll stay there until you feed her. You know why? When you feed her, she comes to eat, and then you take the stick. Because if she's supposed to bring the stick, I think she'll get a zero.

Interviewer: Okay, and after she brings it?

Student: Take the stick and go home to keep it. I don't know why, but I'm a garbage collector. When I find a feather or a stick in the park, I take it home and use it as decoration for my little house.

Interviewer: Okay, so, look? You described two ways of playing with your pet. Pick up the toy, take your pet outside, give him one of the toys and run after him like crazy, then put the toys away. And pick up a stick, get his attention by showing him the toy, throw a stick, ask him or wait for him to bring it to you, and pick up the stick and put it away.

Student: And take the dog with you! You won't leave him lying around.

Interviewer: These two are different, they are different things! But these two things, when we...

Student: They are the same things.

Interviewer: They are the same thing. It's playing with the dog. So when your dog is sad, what do you have to do?

Student: Take it outside and play.

Interviewer: Or grab a stick and play...

Student: Yeah, but the other day I threw the stick so hard, I was practicing archery, with my bow that I bought there in another... thing, that I ended up piercing my father's fireplace.

Interviewer: No way! Okay, but then I'm going to ask you this. What if it wasn't a dog?

Student: What if it was a fish?

Interviewer: What if it were a fish? How do you take care of it? When we don't think too much about the details, it's the same thing.

Student: We... are... wrong here.

Interviewer: Why?

Student: Because there's no way we can catch the fish, get a sponge to wash it!

Interviewer: No, but we clean the water!

Student: My father never cleans the water. He just wants to let the plecos suck up all the trash.

Interviewer: Ah, so the armored man cleans the water! It's not your father, but someone cleans it.

Student: The fish itself cleans the water.

Interviewer: What I'm saying is that... If we don't think about too many details, it's the same thing... when he's hungry, we give him food, when he's dirty...

Student: I want to play! How do I play?

Interviewer: I didn't finish the game! You kept bugging me!

Student: Much better! Look, one here, he's asking for food, the dog and then. And he asked and the food, asked to be washed. Hmm, perfect!

Interviewer: Okay? So, when the aquarium is dirty, we wash it.

Student: But no.

Interviewer: When the fish is sad and we play with it.

Student: How do I play with him? Throw a ball in the water?

Interviewer: You can keep touching the aquarium.

Student: But that gives fish a heart attack!

Interviewer: No... it doesn't work!

Student: My mother said! That they could die of fear.

Interviewer: Anyway, but it's the same thing. That's it, these two games.

Student: Not both! Missing the...

APPENDIX C – Cognitive Interview Transcription - Subject 2

Interviewer: Okay, so... It's a game for little kids, right? But within this game we do some challenges, okay? And this game is to introduce some computing concepts, which as the daughter of [the student's mother] you must already know. But we're going to talk a lot about abstraction and layers of abstraction here. Do you have any idea what abstraction is?

Student: No (Shakes head)

Interviewer: Abstraction is what we do when we start to forget some details of some things and focus only on the important things. When you draw your cat, for example, you abstract a little, because you only draw the shape of its body and such... You only draw the most important things. (Meow) You're not going to draw every single one of its fur, for example, you know? So (Meow), your cat here in the real world, she's concrete. She has a million details that we can't even see. And when we put it on paper, when we have an idea, when we talk about her, it's something more abstract, something that doesn't have so many details, right? And then in computing we have several layers of abstraction... (Meow, Meow)

Student: I don't know if she doesn't want to come in.

Interviewer: She wants to participate.

Student: She will tear the screen.

Interviewer: No, leave her there, I don't think she will... Will she keep bothering us and stay there?

Student: I don't know if she wants to come in. Because every time she comes in, she's a pain in the ass.

Interviewer: The screen won't tear...

Student: She already tore it.

Interviewer: Doesn't she want to go over there with [the student's mother]? Let me open the door for [the student's mother]. (Opens the door) [the student's mother], the cat is tearing the screen. Can I leave it here, with you? (Leaves the cat) Okay, then let me go back, here. This little game that we're going to play is called The Land of Abstraction. In this little game, we have several characters. These characters are superheroes, little animals and princesses, okay? These are the ones here, okay? And then, as we can see here, there are several ways of representing them, each one in a different layer of abstraction. In the higher layers, I abstract more things, I take out all

the details. All I know about them is that they are characters. So to me, they all seem the same, right? When we lower the abstraction layer a little bit, we already know a little more information, we have more details, so I can already differentiate who is a superhero, who is an animal, who is a princess. And when we lower the abstraction layer completely, then we know exactly who is who. We have all the information. So, here is going to be... a little part here. (Enter the exercise) Of these three here, can you say who is more abstract and who is more concrete?

Student: Is abstract when we don't have many details?

Interviewer: Yes, the more abstract, the fewer details, the less information I know, I'm just focusing on what's important. The more concrete, the more details, the more information, the more I can say about that thing.

Student: So the most abstract one is this (choose the correct one), which is just a card?

Interviewer: That's right. (Points to the abstraction meter in the game) There, I've already shown you this, right? In the game, we have this abstraction meter here, which we can increase to become more abstract, or decrease to become more concrete, okay? (Goes to another activity) Ah... I wanted you to separate them here... but they're already separated... and I'll skip this part. (Places them in the highest abstraction layer) Okay, I've already said that, right? In this game, it seems like everyone is the same. But wait, they're not the same! They're the same here because I'm at a very high abstraction layer. If we had all the people here in this house, at a very high abstraction layer, everyone is just a person, you know? So I don't know who's who, everyone is the same. And for many purposes, that's enough, to know that they're people. But if we lower it a little, then we already know that some are children, others are adults, right? And then you go down further and we know who is who, all the details of these people. Uh... Okay, so the first little challenge will be the following: I want you to choose one of the characters, and then I'm going to change the abstraction layer and shuffle them, and then I want you to tell me where he is, after shuffling. Why? Because when I change the abstraction layer, you lose some information, you don't know who is who anymore. So I want to see if you can identify who is who even after changing the abstraction layer. So, just choose one.

Student: I think the [fish].

Interviewer: The little fish?

Student: Yes.

Interviewer: Okay, so you chose that one. (changes the abstraction layer) Now I'm going to shuffle. (shuffles) Who's the [fish]?

Student: Okay, like, I know it's a little animal.

Interviewer: That's right. You can guess! There's no... there's not much of a problem.

Student: (points to one of the little animals) This one.

Interviewer: Do you think this is it? Or are you totally guessing?

Student: I don't know, maybe.

Interviewer: Okay. (Changes the abstraction layer) Wrong, that was the [firefighter puppy], the [fish] was there. Let's try again, with another one?

Student: Do I choose another?

Interviewer: Uh-huh.

Student: The [snow princess].

Interviewer: [snow princess]? (changes abstraction layer, shuffles and looks at **Student:**)

Student: (Points to a princess) This one!

Interviewer: This one? (changes the abstraction layer) Oooh! You got it right this time! Now I'm going to increase the difficulty a little bit more. Choose another one.

Student: [red uniform superhero].

Interviewer: The [red-suited superhero]. This time, I'm not just going to increase one layer of abstraction, I'm going to increase two! (changes the layer) Keep an eye out! (changes again and shuffles) There you go! Who's he?

Student: (points to a character)

Interviewer: (lowers the abstraction layer) It's a superhero! You're on the right track. (lowers it again) [superhero in the red uniform]! You got it right! Now I want to ask you how did you know this was the [superhero in the red uniform]?

Student: Ah... It was in the same place.

Interviewer: It was in the same place, okay, that's important! Why? The characters that are being shown here, they are always the same. We are just changing their view, what we see of their details, but they are the same. So, this here (points to a character) is something. (changes to the lowest abstraction layer) This here is the [fish]. Oh, (changes to the middle layer) regardless of whether it is a little animal in this view and a character in this (changes to the top layer), they are always the same thing. So let's move on to the next challenge. Okay, here I want you to create three groups, and separate them... separate them into three groups: the superheroes, the princesses and the little animals. Okay, you can change it as you want, you can change the abstraction layer. I just want you to... just drag it to the side, leave the three together, okay?

Student: Can I change it here? (points to the abstraction meter)

Interviewer: You can, whatever you want, just make the three groups.

Student: (switch to the middle layer and separate the groups)

Interviewer: It seemed easy, right? But I want you to tell me why you chose this layer (in the middle) and not this one (in the lowest one), for example?

Student: Because I was... to me, it seems like it was easier to differentiate (with the middle layer) than with it (the lowest one).

Interviewer: Yes, it's easier because here, the detail you have is what matters for this problem, right? Separating princess, superhero and animal. So, although here (enters the lowest layer) you can also do it, because here you know who is an animal, superhero and so on... here (enters the middle layer) it's much easier, much more direct, right? (changes to the highest layer) And if you had to do the same thing, but now you have to separate them by their main color. What color are they wearing... For example, the [superhero in the green uniform] is green, the [superhero in the blue uniform] is blue, the clothing part. Again, just separate the groups, but now by color.

Student: Uh-huh. (switches to the lowest layer) Blue here on the side. (continues separating in silence).

Interviewer: I saw that you went straight to the lowest layer, right? And if I asked you that you couldn't use the lowest layer, that you could only use this one (enter the middle one), for example? Would you be able to?

Student: (shakes head) No.

Interviewer: Couldn't you?

Student: There is not much detail.

Interviewer: Yes, the detail you need to solve this problem doesn't exist in this layer (in the middle), right? So it's very important to sometimes lower the layer, right? And it's also important to sometimes increase it, like in the previous problem, which you found much faster, right? (Continue to the next exercise) Here's the deal: they were all invited to a party. And then each of them received an invitation with the right to a guest. But this invitation is different for all of them. Your task is to get everyone to go to the party except one, who will be the person who will give the speech at the party. So everyone will go in pairs, and one person will be left out to give the speech at the party, okay?

Student: Yes.

Interviewer: How do pairs work? (enters the rules selection) Look, here is everyone's invitation. If you want to see [fish's] invitation, for example. Go here. (selects [fish's] rule) This is [fish's] invitation. (points to the graph shown by the rule) This means that [fish] was invited to go with a princess. So, they all received an invitation, but not everyone needs to use your invitation, because you can go as someone else's companion. So if [fish] invites a princess, [snow princess] for example. [snow princess] doesn't need to use her invitation, because she's already going as her companion, understand? So some won't need their invitations. But you have to take everyone except one. Okay? So you can see all the invitations here (enters the rules selection) and choose how you're going to set these things up. And you can change the abstraction layer there as you want, too.

Student: (starts to separate the pairs, an X appears on the platform and the student looks confused at the examiner)

Interviewer: No, but that... you can ignore that, okay. That's how it is, okay. It's just that... it's showing a program error.

Student: How do I select?

Interviewer: It's here (points to the select rules button).

Student: (enters the selection)

Interviewer: That's it, then you choose which invitation...

Student: (choose the [red uniform superhero] rule)

Interviewer: Invitation from [red uniform superhero].

Student: So like... does it, like... change?

Interviewer: Change what?

Student: Like, is that right?

Interviewer: You have to be able to get everyone with the invitations they have.

Student: I'll change it, I'll see later. (changes the characters and enters the rule of [superhero in blue uniform])

Interviewer: Okay. Invitation from [the superhero in the blue uniform].

Student: (keeps moving until you find an invitation that wouldn't work with the current pairs) No.

Interviewer: No, that's right, but he doesn't need to use his own invitation! If he's already going as someone else's guest, that's fine! Got it?

Student: I don't know what it's like here...

Interviewer: No, you just need to bring everyone. Everyone has to come to the party, if you can form all the pairs, that's fine. But, he doesn't... not everyone has to use the invitations, you know?

Student: But here I think it got stuck in the invitations. (referring to characters that ended up behind the invitation rule interface)

Interviewer: Go a little lower and drag a little higher.

Student: (drag and change some pairs)

Interviewer: Now the [superhero in the blue uniform] is alone, he's not going to the party anymore, so... Your job is just to make sure everyone can go.

Student: It's not me... This one here... okay.

Interviewer: Yeah, this one is with a little animal, okay?

Student: This one is with a little animal, this one is with...

Interviewer: Superhero. He's already here, you don't need to worry about him anymore, understand? Worry about those who aren't here yet.

Student: Okay. (continues doing the activity in silence)

Interviewer: The [snow princess] has to be with a princess and the [fish] has to give a speech. It worked! Right? I understood that you wanted to combine the two invitations.

Student: Yeah, but... It should match. It didn't work.

Interviewer: Yes, you can do that. It's possible, okay? But it's not necessary. You've already done this one. But why didn't you ever come back here to the other layer of abstraction (in the middle)?

Student: (shakes head)

Interviewer: You didn't need to because you already remembered, right, that everyone who was a princess was a princess, everyone who was a superhero was super...

Student: Yes!

Interviewer: But, you saw that the invitations were always concrete, very specific, and generic, more abstract, right?

Student: Yes.

Interviewer: So sometimes we have to deal with two different layers at the same time. But you managed it, okay? Everyone's at the party. (proceeds to the next activity) Let's move on to the second phase. (The program gives an error) And it didn't work. Oh, I can't believe it. Technical problems. Okay, let's skip this phase. Now I'm going to go to another game, okay? (enters a new game) Which is a virtual pet, okay? This game is really silly, really easy, okay? You have a virtual pet here and you have to meet all the needs that will appear here, some little balloons. And depending on its need, you have to choose the rules to meet its needs. So look, it's hungry, so you have to come here (enters the rules selection), and what do you do when it's hungry? (shows the rules and selects one) Give it food. And when it appears here (the need balloon), you have to click, okay? Later. So you'll always come here, see its need and click there.

Student: I didn't see what it was.

Interviewer: He's with the three of them, so whatever.

Student: (selects to give food)

Interviewer: Then you click. That's it, now he's not hungry anymore. Now there's something else.

Student: (selects another rule, about bathing)

Interviewer: Uh-huh. You can click it again because it's still dirty.

Student: (click on the balloons)

Interviewer: He's sad, so you have to play.

Student: This one? (selects the play rule and clicks on the sadness balloon)

Interviewer: That's it. So this little game is just for you to understand the concept. On an abstract level, how do we take care of a pet? When it's hungry, we give it food... when it's dirty, we give it a bath... and when it's sad, we play with it. Okay? But in real life, we know that taking care of a pet isn't that simple, right? For example, to give food to a pet, you have to do several things, right? How do you give food to a pet? Oh, I don't know, if you see a bowl of food up high, you have to raise your hand, put your

hand on top of the bowl, raise your hand holding the bowl, lower your arm, bring your arm with the food back, then you have to squat... And then you're already realizing that I'm giving WAY too much detail to explain this, right? I didn't need to. So, how much detail do you need to explain to someone? Could you explain how to feed a pet in five steps? Let's try by talking. Let me see if I can help you, just to mark it here. (enters the creation of a new grammar in the program) Okay, first step!?

Student: Ah, it will be with the dog, because with the cat it is very difficult.

Interviewer: Okay.

Student: I get the food from the cupboard.

Interviewer: Get the food from the cupboard...?

Student: Uh-huh.

Interviewer: Okay, first ball, get the food from the cupboard. Let's move on to the second. Second step!?

Student: Open it and put half of the pot in...

Interviewer: Put the food... in the pot?

Student: Yeah, in the pot... But that's basically it.

Interviewer: Okay...

Student: I had to give more details, if...

Interviewer: Yes, that's right! So we'll have to go back and somehow divide this up so that there are five. But I noticed that so far none of the pets have eaten anything, you just picked it up and put it in the bowl. We could include, for example, waiting for it to eat, then cleaning the bowl...

Student: Uh-huh. Call him to eat?

Interviewer: That's it, good!

Student: In this case, I call him before putting it in the pot, but...

Interviewer: Maybe! So you call first, then you go get the food from the cupboard... Or you get it first, call...

Student: I get it first, then call, because otherwise he doesn't hear the noise (of the food)...

Interviewer: Great, look, I think there's another step then. You take it from the cupboard, then you shake it... to make a noise that drives him crazy. (writes the steps) Then call the pet to eat... Actually, these steps are kind of together, right? Put food in the bowl.

Student: And wait for him to eat.

Interviewer: And wait for him to eat. So you saw that the first time I asked you to describe it in five steps, you gave two steps and, my God, it's already over, right? Because you gave few details, you were at a very high level of abstraction. Uh... So I'm going to ask you to measure this abstraction, measure these details that you're going to describe again, but now for... bathing the pet, which is a little more complex, involves

more steps.

Student: So, I don't do it myself... we don't... we take it to the pet shop.

Interviewer: Okay, but you can imagine, you know more or less the steps involved. Or, you can even describe it by going to the pet shop with them, if you want. The process you do. How do you get them to bathe?

Student: My father takes it.

Interviewer: Okay, but let's imagine that you are the one responsible.

Student: I... I have to... uh, first I have to find it, in the house.

Interviewer: Finding the pet is a good... a good start.

Student: And... uh, get the collar... get the collar to attach to him.

Interviewer: Grab the leash, that's a good thing. Okay.

Student: I said it in the sense... as if the bath had already been scheduled.

Interviewer: Okay.

Student: Lock him up, otherwise he'll bite my hand.

Interviewer: Leash him? Or where to leash him?

Student: On a leash.

Interviewer: Okay.

Student: Call someone to open the house for me, because I don't have a key.

Interviewer: Okay, I'll put "open the house", and we'll manage, okay? But now there have been four.

Student: Yeah... And take him there.

Interviewer: Take him to the pet shop. Okay, it was five steps, but like... Find the pet, get the collar, put him on the collar and open the house...

Student: It's not exactly a bath, right? But...

Interviewer: Yes, but that's not the problem, it's just that they are all very specific tasks. And then you take him, bathe him and that's it all together.

Student: Yes, it's because taking him is basically leaving the house and leaving him there, because there's nothing to do when I get there.

Interviewer: Yes, and then you wait for him to finish...

Student: AND...

Interviewer: I'm not saying you're going to sit there and do nothing. But the process of bathing him involves you waiting for him to finish and then picking him up, right? But okay, if you were to describe it in these five steps, do you think these five steps would be good?

Student: Ah, the last one not so much... it's not...

Interviewer: And how would you change that?

Student: I would have to add more stuff.

Interviewer: And which one would you take? Or could you put two together to be more or less on the same level?

Student: I think so. The "find the pet" and "put on the collar" here...

Interviewer: Are they the same thing? Actually I would say that taking the leash and attaching it to the collar is the same step.

Student: Yes. It's because it's a function to arrest him.

Interviewer: Okay, and then open the house, take him to the pet shop.

Student: Wait for the bath to finish before going to get it.

Interviewer: (writing) Wait until the bath is over and go back with him. He's a bit big, right? But he's done. Let's pretend that these two ("find the pet" and "put on the collar") are just one. And I'm going to ask you this: what if you didn't have a dog? If you had... well, you have... a cat? How would you take care of him? At that level of the game... Oh, he has a need, what do I do? He's hungry, I feed him. He's dirty, I bathe him. Is it the same thing for both the dog and the cat?

Student: No.

Interviewer: No? It's different?

Student: Look, like, when she's hungry, yes, she gives her food. Nowadays, in fact, when I had to give her food, it was a job. To give her a bath... we never gave her a bath!

Interviewer: Oh, does it clean itself? That's true...

Student: Yeah. Just like, sometimes, passing a towel, when there are some... like, not a wet towel, right? Just some...

Interviewer: So do some cleaning, right? Okay, and to play? When she's sad or needy?

Student: Her game is to sit next to us and sleep. So just open the door and let her...

Interviewer: Okay, but at a high level they're not like that... they're basically the same thing, right? You have a problem, you go there and solve it. You have a problem, you go there and solve it. But then when we lower the level of abstraction a little bit, these same things are different, right? How you feed your dog is not the same as how you feed your cat, right? So, what happens? We can solve your care problem in the same way as a dog for a cat if we're talking at a high level of abstraction, without many details. Without knowing exactly what's happening, step by step. But when we lower the level, it can be something completely different, that same plan of "oh, when he's hungry I'll give him food" can be something very different for both a cat and a cat, right? So... yeah, we forgot about playing with him here. Let's do it quickly to speed things up. Can you think of two different ways to play with your dog?

Student: Let's see... Take him for a walk, right? To the parks, anyway. Or just stay at home with him playing.

Interviewer: Playing like what? With a little ball? Throwing the ball for him to catch, right?

Student: AND.

Interviewer: So you have to take it for a walk and have a ball, right? When you have a problem with your dog being sad and wanting to play with it, how could we describe this "playing with it"? Like taking it for a walk, or with a ball?

Student: Little ball.

Interviewer: And why not take it for a walk?

Student: Because it gives a lot of function.

Interviewer: No, okay. But thinking about the problem...

Student: But this... Look, when he's needy, I prefer to play with him here at home, because, like, he usually stays lying down. And to go for a walk... like, he doesn't... he has to walk, and when he walks, he kind of falls over.

Interviewer: Yes, they are different ways of playing with it, but both are the same "ways of playing". The same, as long as all you know about it is that they are a way of playing. Right? So, high level of abstraction, right? So that's basically it, okay? Actually, we've already finished with the little challenges. We skipped a little phase, but it's not worth it, because it's not working. Ah, the other little phase here was basically for you to take the little animals here... let me separate... These three little animals, they need to go to a certain place, right? And then here I'll have three... three... four rules to move these little animals. So I can move them... I can move any little animal from one place to another, or I can move a specific little animal, like the [fish]. The [fish] goes from here to another. And then you would have to move all three. Which rule would you choose? Would you take one for each? Would you keep switching?

Student: Do you have to move all the little animals?

Interviewer: That.

Student: I don't know, and does it have to be one at a time? Or like, all together?

Interviewer: No, in reality, you have a rule that is at a higher level of abstraction, so it doesn't care who the little animal is, it can take any little animal from one place to the next, okay? So here I could move this little animal here, for example. Or this one... or this one. Okay? But, these other rules. For example, the one about moving the [fish], I can only move the [fish], only the [fish] goes up. So I can't use it on the [green lizard]. And I can only see... use it if I can see the [fish]. Look, this isn't just any little animal, this is the [fish]. So you would have this problem, you would have to move the three, what would you do? Would you come here, see the [fish] and take the rule of the [fish] and apply... then the [green lizard], then the [firefighter dog], or would you apply it to this one?

Student: That's it... I was going to apply just for the little thing.

Interviewer: Just for the little animal? So would there be any problem with you doing that?

Student: I don't think so.

Interviewer: You don't think so? Okay, there would be no problem doing that. Because all you want to do is move the little animals. To move the little animals, you don't need to know who is who and use their specific rule, they are all little animals, they can all walk, that's what matters to you. Uh... After that, the other rule was a rule for them to eat their food. And then I have three rules, because each one eats a different food, but, at a higher level of abstraction, these three rules seem to be the same thing. A little animal that eats a food. So regardless of whether it's the [firefighter dog's] food, or the [fish's] food, here (in the highest layer) it's always the same thing. Okay? It only seems different when you come here (lower layer), you understand? And then, here in the game, you wouldn't know which rule is which at first, right? So you could try to apply this rule, try to eat and it wouldn't work. If that didn't work, what would you do? Why wouldn't it work? Because here you would see that there are three little animals and three foods, and then you would try to make one of the little animals eat and it wouldn't work. What would you think if it didn't work?

Student: That the little animal has changed.

Interviewer: Yes and... Because, here (highest layer) it's the same, and that's what's there. But when you lower the level of abstraction, things are different, and you don't know that they're different if you're only at the highest level. Got it? So sometimes things can go wrong when we're only seeing a few details, and we don't understand why it's going wrong. Because it doesn't make sense... it's a little animal and it's a food. But, for it to make sense, we have to lower the level of abstraction and we have to see that, in fact, that food isn't the same for everyone. That each little animal only has its own food. Sorry, this part was supposed to be a little game and you were supposed to see these things, but it didn't work, so I had to do it by force. But that was it, okay? What matters most to me are your answers. In the challenges, I think you did very well! Uh, I don't know if you want to... uh, tell me what you understood from all this, about abstraction, about details?

Student: No, you can understand that sometimes the highest abstraction is easier to differentiate some things and the lowest is the same thing, but when you look at it you can differentiate with something more specific.

Interviewer: Uh-huh, the lower the abstraction, the more concrete it is, the more details it has, the more you can differentiate, that's it.

APPENDIX D – Cognitive Interview Transcription - Subject 3

Interviewer: [student] is your name? Okay, let me just put it here, real quick.

Student: I didn't know that.

Interviewer: That you are [the student's friend]'s best friend?

Student: I didn't know that.

Interviewer: Okay. So today we're going to talk about a very different word: Abstraction. Have you heard of this word before?

Student: No

Interviewer: So abstraction is when you try to forget about the details for a while and focus only on what matters to you, okay?

Student: Uh-huh

Interviewer: Like, do you like to draw?

Student: Yes

Interviewer: When you're going to draw, I don't know, a cat, a dog, you draw its snout, its ears, its four paws and everything else, but you don't draw every single strand of hair there, right?

Student: No.

Interviewer: Because otherwise you would spend a year there, drawing, right?

Student: AND.

Interviewer: So, when we do this, when we forget about these details and focus only on the important things, we say that we are abstracting. This is abstraction, okay? In this little game here, we will work with layers of abstraction. So it is as if you were in slices like this and you were looking at the same things, but with a different level of detail, you know?

Student: Oh yes!

Interviewer: So, when we are in a high, very abstract layer, we...

Student: There is a lot of detail.

Interviewer: On the contrary. When we are abstracting too much, we are forgetting a lot of details. So there is almost no information, right? So, here, for example, there are several characters, but you don't know anything about them. You only know that each one is a character.

Student: Okay.

Interviewer: When we lower the abstraction layer a little bit here, we already know

a little bit more about them. Here we know that this character is a princess, this one is a superhero, this one is a little animal, okay? But we still don't know much. When we lower it a little bit more, which becomes more concrete, closer to reality, then we already know a little bit more about each one of them, there is more information, more individual things, okay? So just open it to see if you more or less understood the concept. Can you tell who is the most abstract here? Who is the...

Student: The topmost one is the most, and the one on the right is the least.

Interviewer: Okay? And can you explain to me why?

Student: Uh-huh. This one has the most details and the other one has the least.

Interviewer: Uh-huh, so you can tell a lot more about this character here than by looking at this one, right?

Student: Yes

Interviewer: So, let's go. Let's go to the first challenge. The first challenge, here look. Here in this game we have this abstraction meter, where you can go through the abstraction layers and change, okay?

Student: Uh-huh.

Interviewer: Here you can use any of them, whichever you want to choose. And I want you to organize them by group. The princesses on one side, the superheroes on the other, and the animals on another. Okay? Then just grab this and drag it the way you want. Then make three groups like this, okay? But you need to organize them according to what they are: superheroes on one side, princesses on the other, and animals on the other. Can you do it?

Student: Uh-huh. (organizes)

Interviewer: Okay, what now? How do you know that these are princesses? These are little animals and these are superheroes?

Student: I don't know.

Interviewer: So why did you organize this here in this layer of abstraction?

Student: First I was guessing.

Interviewer: You were guessing. You can't guess, you have to solve the problem. Let's go. If you could touch here, use any of the layers, which one would you use? This top one? This one in the middle?

Student: This one here.

Interviewer: Would you use this one? Why?

Student: Because it is neither too easy nor too difficult.

Interviewer: Yeah, you have the information you need and there's not too much information, right? So, organize it now. You saw that you had organized it wrong.

Student: (organizes)

Interviewer: That's it, okay. Now your task is to organize them by their main color. What color do they have the most?

Student: Make groups by color?

Interviewer: (nods) By the colors.

Student: Should we change (the abstraction layer) or leave it like this?

Interviewer: You can change, it's up to you to decide.

Student: (change layer and organize)

Interviewer: Okay, that's right, that's it. Now here's the deal: they've all been invited to a party...

Student: Uh-huh.

Interviewer: And then each of them received an invitation, with the right to a companion.

Student: Hmm.

Interviewer: Okay, then you'll be able to see each of their invitations, okay? Not everyone needs to use your invitations, because if you're invited by someone else, you can go as someone else's companion and then you won't use your invitation, understand?

Student: Uh-huh

Interviewer: But each one has a different invitation, look (enters the rules selection). Let's see the invitation from [fish], from the little fish (selects the rule from [fish]). Her invitation says that she can go with a princess as a companion.

Student: Uh-huh.

Interviewer: So you can take her and tell her to go with any princess. Don't pay attention to this X, okay? It's just nonsense (program error). And then here you can see someone else's invitation and see how that other person got invited.

Student: Ah, like a puzzle, each one with its pair!

Interviewer: That's right, that's right. You need everyone to go, except one, because there will be one left over, since there are nine. But you don't need the two invitations to be... to match each other, okay? If... for example, if the [snow princess] has to go with a princess and she goes with the [hero in the blue uniform], there's no problem IF the [hero in the blue uniform]'s invitation allows her to go with him as a date, understand?

Student: Okay.

Interviewer: So here (shows where you enter the rules selection) you can choose which invitation you want to see and then you just put them up in pairs, ok?

Student: Okay. (starts to organize and look at the invitations, until he organizes three together)

Interviewer: No. But they can only go in pairs. Each one is with only one other.

Student: (tidies up and continues organizing in pairs)

Interviewer: Uh-huh. Okay. I'm going to ask you why you didn't switch at any point to other levels of abstraction here? For other layers, you just stayed at this one.

Student: Because I had forgotten it existed.

Interviewer: Ahaha, you had forgotten. But that's because you already knew, for example, that the [green lizard] is a little animal... the [hero in the red uniform] is a superhero... so you didn't need to come back here to find out, right? But, as you can see here, he's not asking for the same layer, right? This one has more details. This one has neither too much nor too little, right? So, sometimes we need to deal with this, right? With two different layers at the same time.

Student: Uh-huh.

Interviewer: But you understood that the specific can go with the generic there. Now we're going to move on to another phase. Here, look, we have the three little animals here, but this time I only have two layers, look. The little animals, generic, generic things, and more specific things. And here you have these four rules here to move them.

Student: Uh-huh.

Interviewer: . Each one of these is for moving only a specific object. So this one moves the [green lizard]. This one moves the [fish]. And this one moves the dog. And this one can move anyone, so you have to use it at a higher level of abstraction. You can forget about the details, you know?

Student: Uh-huh.

Interviewer: You need to get these three little animals up here, okay? So I'm going to ask you what rule you're going to use? What's your strategy to get everyone up there?

Student: I'll go one by one, like this.

Interviewer: It could be one for one, but why would you choose one for one? And what is the advantage of one for one?

Student: Isn't it so that everyone can get their food? This one, this one and this one?

Interviewer: Uh-huh. But if you can use this same rule, which is more generic, which can be... which doesn't matter to anyone. Why would you use... would you keep changing rules and going one by one? Isn't it easier... Here you have to click... no... here look. Okay, no, it worked. Because it's much easier to do everything here at this level of abstraction where you don't need to know who's who, right? You just need to know that it's a little animal and move on. So isn't it easier, faster, to use just one rule, which is always the same? Than you keep changing... going to the [green lizard], then to the [firefighter dog]... And then after they get here to the food you have to feed them, but each one eats a different food. Here it has the three rules for eating. And then when you look at how you use this from here at a higher level of abstraction, without much detail. The three are the same, the three are like this, only here you can't see them like this, because when we lower the level of abstraction, they are different, each

one eats something different. So you can't try to make the [firefighter puppy] eat his food using here, because here it's not the [firefighter puppy], here it's the [green lizard].

Student: Uh-huh.

Interviewer: Got it? So, even though it all seems the same when we don't pay much attention to the details...

Student: It is not.

Interviewer: It's not really! So how would you get everyone to eat their food now?

Student: (starts using the rules) Oh no, wait, this one, look...

Interviewer: That's right, but you can't do that by looking at it at this level of abstraction, because you don't know who's who.

Student: (changes the level of abstraction)

Interviewer: That's it, you have to do it like this. Okay, okay.

Student: No, wait.

Interviewer: That's it. Then you have to change it, you have to come here.

Student: Hmm.

Interviewer: That's it, you're clicking on the other one. Put it to the side, and then click on the [green lizard]. That's it, now yes, now... no, on the mushroom. That's it, it worked.

Student: Let me just put it here on the side.

Interviewer: Okay, let's move on to the next one then.

Student: I haven't put it there yet.

Interviewer: No, that's okay. You already understood, that's what I needed.

Student: Is it here?

Interviewer: That's it, you scroll up a little bit and then you'll get to these three. That's it, scroll up there. More. And then it goes in. That's it, you're done. Now we're going to the last one. Well, this last one is... just so you understand the concept, okay? There's not much to do in the game. Basically, this game is for you to take care of a virtual pet. So you have this puppy here and then over time he gets hungry, he gets dirty...

Student: Like pow?

Interviewer: That's it. That's right. And then when he gets hungry, you have to come here, and what do you have to do? When he's hungry?

Student: Give food.

Interviewer: Food, that's it. Then you click on his hunger, until it's gone. That's it, look, he's already dirty and sad. That's it, that's it. It looks like you got this one easy. Then he's hungry again. It's kind of fast. He's done, he won. But in real life, we know that it's not that easy to take care of him...

Student: It's not that fast!

Interviewer: Yes, and it's not that easy, right? He's hungry, you go there and click on something and he's not hungry anymore? It's not like that, right? So we know that to take care of your pet's hunger you have to do a lot of things, right? Could you explain to me what you need to do to feed your dog?

Student: Mm, uhm.

Interviewer: But I want you to think in five steps, okay? So you have to be careful to give enough details to make it five steps.

Student: Uh-huh, can I talk now?

Interviewer: Then I'll write here and then you'll tell me the steps.

Student: Let me write?

Interviewer: Could it be.

Student: (starts writing)

Interviewer: (reading) "Buy food". We're going to do it like this, the little ball will appear here and then we're going to make another one.

Student: (starts writing)

Interviewer: (reading) "Have a pet", great. "Wait for him to get hungry". There you go, that's it, you can click. "Feed him". There were four. One is missing. So far you have "buy food", "have a pet", "wait for him to get hungry", "feed him"...

Student: Hmm...

Interviewer: Maybe you could split that "feeding" into two. Then you could "take the food" and "put it in the jar", for example.

Student: (starts writing)

Interviewer: That's it. Perfect. Now I'm going to ask you to do the same thing, but to explain how to "bathe" your pet. This is a slightly more difficult task, and involves a few more details. But you also have to describe it in five steps.

Student: Hmm. Do you do it down here?

Interviewer: It could be. So you have to think carefully about how you're going to... how many details you're going to do. It doesn't matter how you're going to leave it there. I'll fix it later.

Student: (starts writing)

Interviewer: (reading) "Having a pet". "Him being dirty". "Buying soap". "Taking him to the bathroom". "Turning on the shower". "Giving him a bath". It's good to dry him afterwards too, right?

Student: One, two, three, four, five, six.

Interviewer: Uh-huh, look, there are already six steps, so you should change it a little bit. Do you know what you could do? You could think about how you can combine two of these, for example. Look, "having a pet" and "it's dirty" are just conditions, right? So you could do that and leave the two together, saying that these are just one. So this is a step. The first step is for you to have a pet that's dirty, and then you're going to buy

soap, take it to the bathroom, bathe it, rinse it, right? And... lastly?

Student: (writes dry)

Interviewer: That's it.

Student: (retraces steps and deletes old ones) Trash, trash, trash, here.

Interviewer: Now, yes, very neat! Okay, and now the last challenge. Actually, there are two. Remember that in that Pet game he also got sad and you had to play with him?

Student: Uh-huh.

Interviewer: So you have to describe in five steps how you can play with your pet? But then I'm going to ask you to write another game. So you have to think of two.

Student: (starts writing)

Interviewer: (reading) "Having a sad little animal". "Having a little ball". "He liked the ball". "Throwing it for him to fetch". And lastly... "He fetches it and brings it back". Perfect. Now can you think of a game other than the ball game? To play with him?

Student: Hmm...

Interviewer: Run with him in the park?

Student: (starts writing)

Interviewer: (reading) "Have a sad dog". "Take him outside". "Take a ball with you". "Take two slippers and put them on the floor separately". Oh, are you going to play soccer with the dog? "He'll be the goalkeeper". Cool!

Student: I can't do that because my dog, when I arrive with the ball in the yard, he already starts asking to come in.

Interviewer: Doesn't he wait until you score the goal?

Student: Then I give the first kick and he starts to get desperate. He wants to go in.

Interviewer: Okay, let me try to find where you did the other one here. Here! You saw that they both started the same, right?

Student: Uh-huh.

Interviewer: Okay. So, remember your little game? We're doing these processes. We're describing everything in five steps, but they're all the same things you did in the game: you're sad, you play. You're just explaining it better, giving more details about how you feed him, how you play with him, right?

Student: Uh-huh.

Interviewer: So there you had to play with him when he was sad. So if he is sad, what do you have to do?

Student: Click on the balloon.

Interviewer: Yes, but the way you explained it, will you have to do the process you explained above of throwing the slippers or the one below of playing ball?

Student: Anyone.

Interviewer: Anyone, exactly. Why anyone?

Student: Because they both work.

Interviewer: They both work. Why do they both work?

Student: Because they both entertain the dog... the animal?

Interviewer: Exactly. Both start with the dog sad and end with him happy because he had fun playing. That's right. And if we didn't have a dog, but had a fish, for example, would it be the same?

Student: No.

Interviewer: But what about that little game over there? Oh, the fish is hungry? What do you do?

Student: (no answer)

Interviewer: You give it food, the same way, right? The dog is hungry, you have to give it food, the fish is hungry, you have to give it food, the fish is sad, what do you have to do? Play with it.

Student: I don't know, how do you play with a fish?

Interviewer: Well, how you play with a fish is different from how you play with a dog.

Student: Grab a fishing rod and start playing tag with him.

Interviewer: Uh-huh, that's right. So, the process here in more detail will be different, but the one at a higher level of abstraction, where you don't care much about the details, is the same thing, right? You're just saying: oh, play with it, right? So we can have these situations where: when we talk about things without going into too much detail, they are the same thing; but when we explain exactly, step by step, what needs to be done, they can be completely different things, right? You'll never be able to play soccer with a fish, for example, but you'd be playing with it just the same. So, when we think without any detail, the two pets are the same.

Student: s No... (shakes head in denial)

Interviewer: Ah, but they are similar, in the sense that they have needs: he is hungry, he is sad, he gets dirty, and you have to solve them. You have to solve them by giving him food, or playing with him or cleaning him. This is the same for everyone, but how is it done...

Student: It's different!

Interviewer: ...it changes a lot, right? Exactly. And even when it's the same, because you can have different ways of playing with it, right?

Student: Yes.

Interviewer: So that was it. Those were the little challenges. And I wanted to ask you then, if you could summarize what you understood about the abstraction of this issue of layers?

Student: Hmm.

Interviewer: Yeah, I know it's hard. You see, when we have...

Student: That when it's... within a more detailed layer like that, it becomes... it becomes much more specific what should be done or what is done. Than when you just say, like... very abstract.

Interviewer: Uh-huh, exactly, and why sometimes we deal with things with few details, without giving any information and sometimes we have to talk step by step...

Student: Because there are times when it is not necessary to give so many details.

Interviewer: And then it's much faster...

Student: It's faster there.

Interviewer: That's it, that's it. So that's it, thanks for participating.

Student: Thank you!

APPENDIX E – Cognitive Interview Transcription - Subject 4

Interviewer: Your name is [student]?

Student: (Nods)

Interviewer: How old are you?

Student: Ten.

Interviewer: And in which grade are you in?

Student: I'm on the fifth.

Interviewer: What I want to play with you is about something called abstraction. Have you ever heard of that word?

Student: (Shakes his head)

Interviewer: So, it's quite different, quite crazy, right? Abstraction is basically when we think about things without worrying too much about the details. We only focus on the things that matter. Like, I don't know, when you're going to draw a cat or a dog, you only draw the things that matter about that dog, draw its ears, its snout, its eyes... You don't draw every single detail. Like, you're not going to draw every single hair on that dog in the drawing. So, when you're drawing, you're abstracting. The more you abstract, the more you stay in the world of ideas, things that require no existence. You're only focusing on the things that matter. The less you abstract, the closer you get to the real world, to the concrete things that exist here, with more details, right? More information. (puts on the game) Here we have a little game, the land of abstraction. In this game we have several characters and we have an abstraction meter here on the side. So when I go to the highest level of abstraction, I forget all the details. And all I know about these characters is that they are characters.

Student: Okay.

Interviewer: When I go a little lower in abstraction level, then I know more details. Then I already know that some characters are princesses, others are little animals and others are superheroes. Okay? And then, when it goes a little lower in abstraction, then I know everything about them. Then I know every little detail about them, I see that they are all very different. So, just to see if you understand more or less the concept of abstraction, can you tell who is more abstract, who is less abstract here?

Student: (hesitates and looks confused)

Interviewer: Don't worry, there is no right or wrong answer. I just want to know if you understood more or less what I told you.

Student: That!

Interviewer: What is this? The most abstract or the least abstract?

Student: I don't know.

Interviewer: Is this one more detailed or less detailed? Do you know more about it or less?

Student: Less!

Interviewer: Less. If there are fewer details, then you are abstracting more, it is the most abstract. And who is the least abstract?

Student: (points to one)

Interviewer: This one, because it has more details, so I'm abstracting less, it's closer to the real world, right?

Student: He nods.

Interviewer: Now... I'm going to ask you to separate the most abstract ones to one side, the least abstract ones to the other, and the ones that are neither too much nor too little to the other side. To whichever side you want, just drag it.

Student: (drags) Yeah, I think it worked.

Interviewer: Here you made four groups, is that it?

Student: (affirms with the person)

Interviewer: Okay, why did you leave those two together?

Student: Because they are the most abstract.

Interviewer: That, and those two?

Student: Ah... why... no... it's both of them together.

Interviewer: Yes, they look alike, right?

Student: I didn't know if this one was here.

Interviewer: Okay, do you think here is here, or is it here?

Student: Here!

Interviewer: Here, why?

Student: Because it has more details.

Interviewer: Because there are more details than this one, right?

Student: Uh-huh

Interviewer: We know more about her. Okay. Okay, so now I'm going to give you a little challenge. I want you to choose one of these characters, and then I'm going to increase the level of abstraction a little bit and I'm going to shuffle them around. And then I'm going to come back here and you're going to have to tell me who's who, okay? You're going to have to find the one that chose you. Who do you want?

Student: (points to one of the characters) Her.

Interviewer: [mermaid princess]?

Student: (nods) Yeah!

Interviewer: Okay, so keep an eye out. (changes level of abstraction) Do you know who the [mermaid princess] here is?

Student: (points to one of the characters) This is the one.

Interviewer: Is that the princess? (goes back to the bottom layer) That's it. So, [mermaid princess] (changes the top layer). Now I'm going to shuffle (shuffles). Who is the [mermaid princess]?

Student: (points to one of the characters)

Interviewer: This one? Why is this one?

Student: Because since you started shuffling, I've been following you.

Interviewer: Okay, so you knew she was this one. When we raised the level of abstraction, you kept an eye on the one that was in the same place. So I'm going to make it a little more difficult now. I want you to choose one again. And then we're going to raise the level of abstraction to here (highest layer) and then we won't know much about the other characters anymore. Let me separate them more so you can tell more easily who's who. (separates the characters and looks at the student)

Student: (points to one of the characters) [firefighter puppy].

Interviewer: [firefighter puppy]? Okay. (switches to the middle abstraction layer)

Student: Calm down, I haven't recorded where he was yet.

Interviewer: (back to the lowest layer)

Student: Okay

Interviewer: (raise the layer, wait a second, raise it again and shuffle)

Student: (points to a character)

Interviewer: Is that the last one?

Student: Uh-huh

Interviewer: (goes down to the middle layer) Still think this is it?

Student: Uh-huh.

Interviewer: (changes to the lowest layer) Great, you got it right! So, here's what we saw: in fact, you don't have... (multiple characters). They're always the same characters. But when we change here (in the abstraction meter), we're just changing the view of them, they're still the same, right? So the [firefighter puppy] here (in the lowest layer) continued to be the puppy here (changes to the middle layer), and continued to be a character here (changes to the highest layer). So that's why you were able to find him even without seeing him, right? Okay, now, I'll leave that here. And I'll tell you this: there's going to be a party, all the characters are going, but they need to go in an organized way. So you need to have the little animals on one side, the princesses on another side, and the superheroes on another. Three separate groups. You need to organize them and you can use any level of abstraction here that you want.

Student: Even the lowest?

Interviewer: Even the lowest. And then I'm going to ask you to choose a (level of abstraction) and organize them.

Student: (choose the lowest level and organize)

Interviewer: So, do you think you got it right?

Student: I don't know.

Interviewer: Why did you separate like that?

Student: Here are the princesses. Here are the superheroes. Here... I didn't know about this one.

Interviewer: Oh, that's what I was going to ask you: were you in doubt about any of them, whether they should be a superhero, a princess or an animal?

Student: I've never seen him.

Interviewer: Well, you don't know some of these, so you didn't know.

Student: No, just this one!

Interviewer: Okay, and why didn't you use this level of abstraction? (switches to the middle layer)

Student: I don't know.

Interviewer: Your task was to separate them into princesses, superheroes and pets. When you are at this level of abstraction (in the middle), all you know about the character is whether they are a superhero, pet or princess, right? And this one here (changes to the lower layer) you even know, but you have to know (the character), right? Just like you were in doubt about this one, so wouldn't it be easier for you to solve this problem at this second level of abstraction?

Student: (reluctantly) No.

Interviewer: Do you disagree or do you agree?

Student: I agree.

Interviewer: Okay, now what if I ask you to organize them again? Only instead of leaving the superheroes on one side, the princesses on another side and the little animals on another side, you organize them by color: people with the color blue on one side, green on another and red on another.

Student: There is no way to.

Interviewer: There is no way? Even if you change the level of abstraction?

Student: If I change, yes.

Interviewer: And which one can you do?

Student: The last one.

Interviewer: The lowest one? You can't get it in that (intermediate)?

Student: (shakes his head)

Interviewer: Okay, and in this (changes to a more concrete level) can you do it? You can do it then.

Student: (starts to separate and stops on character that has two colors) Does this count as blue or red?

Interviewer: Red. That's right. Now, here's the thing: they all received an invitation to go to the party. But each of their invitations entitles them to one companion. So they can go and bring one person along, okay? But each one's invitation is a different invitation, which entitles them to a different companion. So here, look, (opens the rules selection) here are all the invitations, okay? On the invitation here for [fish] (selects one of the rules), the little fish, she and a princess can go. On the invitation here for [the superhero in the red uniform], he and a little animal can go, okay? You don't need to use everyone's invitation, but you need everyone to go to the party. Either as the main character or as a guest, okay? So I want you to organize here, for example, the [superhero in the red uniform] and a little animal. So, here I'm already saying: Oh, they're going to the party together with the invitation from [the superhero in the red uniform]. Okay? That's for everyone, except one. One will have to stay out.

Student: Yeah, because I saw, like: this one goes with... let's say, when this one goes, it goes with this one. When this one goes, it goes with this one. When this one goes, it goes with this one. When it's this one, it goes with this one. (referring to several different pairs) And then there won't be any to go with this one. (referring to the last one that would be left because it's an odd number of characters)

Interviewer: That's right, one of them will be left out. Okay, and then I want you to do something here, look (go to the rules selection). You can come here and look at the invitation for each of them, okay? Just click here (click on one of the rules) to open the invitation. And you can organize it any way you want. You just need to make the pairs according to the invitations.

Student: I won't know.

Interviewer: This one is harder. But you can change the level here (shows the abstraction meter) to whatever you want.

Student: Oh, great. (starts to fix it)

Interviewer: Why are you trading one pet for another?

Student: It's not working!

Interviewer: No, you can ignore that "x" (referring to the platform's error indicator), but why are you swapping one little animal for another?

Student: I don't know, so I can see which one is correct. (the student was in the intermediate layer exchanging the little animals in the hope that by placing one of them the platform would indicate success, but success would only be obtained by placing the [green lizard], which is a little animal, in the most concrete layer, where it is seen as the [green lizard] in fact, and not as a little animal)

Interviewer: You don't know what the [green lizard] is?

Student: AND!

Interviewer: So what do you have to do to find out who the [green lizard] is?

Student: I don't know.

Interviewer: You have to change the level of abstraction here (points to the meter). At what level can you tell who the [green lizard] is?

Student: (switches to the more concrete layer without saying anything)

Interviewer: That!

Student: (makes the pair) And so?

Interviewer: That's right! Now, look: these two are already going to the party, these two are missing (points to the other characters). Now you can see someone else's invitation. Whose invitation do you want to see?

Student: From [mermaid princess]. (enter the selection, choose the one from [mermaid princess] and assemble the pair accordingly)

Interviewer: That's it, now it's someone else's invitation.

Student: (continues organizing the pairs)

Interviewer: They're getting in, but there are still two left.

Student: Okay. (continues separating the pairs, but hesitates for a while)

Interviewer: Why did you stop? What confused you?

Student: It's because I thought there was only this (character) of this (type, classification) to go, but I saw that there was another.

Interviewer: I understand, then, since you already knew here (from the most concrete appearances), you didn't even need to come back here (in the intermediate layer that shows the character classifications), right? But you saw that here (points to the [green lizard]) you couldn't solve it because you didn't know who it was in more detail.

Student: Uh-huh.

Interviewer: Okay, that's it. Let me see what's next. Okay, here, look, we have the three little animals: the [green lizard], the [firefighter dog] and the [fish]. And they have to go home, they have to come up here (points to the game board) to eat.

Student: Wait, it's the [green lizard] (points to one of the little animals), the... (points to another)

Interviewer: You can change (the abstraction layer) here (in the abstraction meter) whenever you want. You can go back and check, okay? So, here, in this game: to move up a little space, you need to come here (in the rules selection) and use a specific rule. So you can use... To move the dog up, here (select one of the rules). And here (apply the rule) for it to move up one space. And then to move the [green lizard], there's the [green lizard] rule, okay? But in addition to these three, one for each, there's this one for moving animals. You don't know who the animal is, you're moving at a higher level of abstraction, so you can use it on any animal, okay? If you have to make the three of them move up here, which of these rules would you use? Which of these actions?

Student: I think this one (points to the generic rule).

Interviewer: Of the little animal? Why?

Student: Because I don't know. To move any animal.

Interviewer: Uh-huh, you have to move the three of them, right? And it doesn't matter which rule you use, so if you use the one about moving animals, you just keep applying it, you don't need to keep changing it, right? You just keep walking and at some point you'll get here, with the three of them. So they need to eat, okay? But each one eats something different, okay? Let's go. The [fish] eats fish food. The dog eats dog food. And the [green lizard] over there eats the mushroom. But in these rules each animal eats a specific thing. So could you use them like this? In this layer of abstraction (the intermediate one)?

Student: (no answer)

Interviewer: Because you saw, the three rules, when we look at them, without details, they are the same. It's a little animal and a food.

Student: Like, when the animal arrives here, does the food change? Or does it stay the same?

Interviewer: No, it's the same.

Student: Ah, so it's the mushroom (points to one of the foods on the board), the dog food (points to another) and the fish food (points to the rest).

Interviewer: That's it. But can you understand that these three rules are different? But you can only see that they are different when you lower the level of abstraction. When you don't care too much about details, they are the same thing, right? Do you agree?

Student: (nods)

Interviewer: Okay. That was it in that game. Now, the second game. This second game is pretty silly, okay? It's just to introduce a concept. Which is for you to take care of a virtual pet. Basically, when you start playing... It's kind of fast, so you have to run. But when you start playing, you'll have this puppy and it'll show you little need bubbles: oh, I'm hungry; oh, I'm... dirty, I'm sad, I want to play. And then, look: when it's hungry, you have to come here (in the rules) and feed it (select a rule) and then just click here (click on the need bubble to apply the rule), okay?

Student: Uh-huh.

Interviewer: And then when he's dirty (indicates another need balloon), you have to give him a bath (shows another rule), when he's sad (indicates another balloon), you have to play with him (shows another rule). So that's it, you can try to do it. But you have to be a little quick because he'll have other needs again.

Student: (complete the activity silently)

Interviewer: That's it! So that was just to tell you that the idea of taking care of a pet is this: if it's hungry, you go and feed it; if it's sad, you go and play with it; if it's dirty,

you go and bathe it. But this is taking things too far, without worrying about the details. In real life, with so many details, we know that it's not that easy, right? Each of these things actually means a lot of things. For example, how do you feed your pet?

Student: Like this?

Interviewer: Can you explain to someone how you go there and give food to your pet when he is hungry?

Student: No.

Interviewer: Describe a step. For example, if I were to describe how I feed my pet, I would say that I have to take a step toward my laundry room, then take another step, then take another step, then raise my hand, then grab the little bowl of food, then lower my hand, then turn around, then take another step... But you're understanding too much detail, right? It's too complicated to explain it to you like that, and everyone knows what it's like to go to the laundry room and grab a little bowl, right?

Student: Uh-huh.

Interviewer: So there are several ways to explain the same thing: we can give a lot of details, which is a pain to listen to one by one, right? Or we can simply say: oh, give him food. But what if the person doesn't know what giving food means? Doesn't know how to do it? You have to explain a little more details, you don't need to give as many as I did now, which is really boring to listen to, but you need to give more than just saying: oh, give him food.

Student: Do you have to write?

Interviewer: No, now I'm going to challenge you to try to explain how to feed your pet in 5 steps. And then you have to think about giving enough details to give 5 steps correctly. I'll write it down for you, okay? The first step. (looks at the student)

Student: (no reaction)

Interviewer: You're here, and your pet is there barking for food, what do you do? The first thing you do?

Student: I get the food.

Interviewer: Get the food (writes). Okay, second step?

Student: Yeah... I put it in a little pot.

Interviewer: Put it in a little pot (writes). We already have 2 steps. Third step?

Student: I... uh... I don't know.

Interviewer: Call the dog to eat?

Student: AND.

Interviewer: (writes) So, what else?

Student: There is no more.

Interviewer: There are no more? So you took three steps. Could I not describe these three steps in a little more detail to make five?

Student: Can you just erase everything? And start from scratch.

Interviewer: Yes, you can erase everything (erase). First step?

Student: Calm down, don't write yet.

Interviewer: No pressure.

Student: I take the... the little thing where I take the food in strips.

Interviewer: Get the dispenser?

Student: That.

Interviewer: And now?

Student: I put the food.

Interviewer: Put the food... in the dispenser? (writes)

Student: Uh-huh.

Interviewer: Okay, what now?

Student: I put it in her little pot.

Interviewer: Put it in the jar (writes).

Student: I didn't understand you calling.

Interviewer: Call (writes).

Student: But it won't give five.

Interviewer: Well, it's already improved, look: now you have four steps. What else could you share here? Explain in a little more detail?

Student: Can I take one more and put it here, at the beginning?

Interviewer: There is?

Student: Yeah... can I talk? Yeah... can I go to the feed.

Interviewer: Go to the feed... find the feed?

Student: AND!

Interviewer: (writes and starts reading all the steps) Find the food, get the dispenser, put the food in the dispenser, then put it from the dispenser into the bowl and call the dog to eat. Five steps on how to feed your pet. And if I asked you to do the same thing in relation to bathing, it is also a little more complicated, there are more steps, so it should be a little easier for you to put it in five steps to solve. Let's go?

Student: I'll get the dog.

Interviewer: Catch the dog (writes).

Student: It's like... anything here? Get the shampoo.

Interviewer: Get the shampoo (writes). You've already got the dog and you've already got the shampoo.

Student: Tap or shower, right? Turn it on.

Interviewer: Turn on the tap or shower (write).

Student: Soap the dog.

Interviewer: Soap the dog (writes). One missing.

Student: And then remove it with water...

Interviewer: Rinse?

Student: Rinse!

Interviewer: Did it work? Are you satisfied? Is it good? Five steps of the same size: Grab the dog, get the shampoo, turn on the shower, lather up and rinse. Okay. And now I'm going to ask you to take five steps to play with the dog.

Student: Okay. Pick up the toys.

Interviewer: Pick up the toys (writes). What else?

Student: Call the dog.

Interviewer: Call the dog (writes and looks at the student).

Student: I don't know.

Interviewer: How are you going to play with him? What do you do when you play with the dog?

Student: Play with... with a ball.

Interviewer: With a ball, okay.

Student: Catch the ball!

Interviewer: So we can now say that it's about catching the ball (edits the first step). Okay, (reading) catching the ball, calling the dog?

Student: To play?

Interviewer: Playing is what you are explaining.

Student: Play with the dog.

Interviewer: Yes, but how? You picked up the ball and called the dog, what do you do now?

Student: Play with... throw the ball.

Interviewer: Throw the ball, good! (writes) After you throw the ball, what do you do?

Student: The dog catches it.

Interviewer: The dog catches it, okay. (writes) And lastly?

Student: Yeah... I just pick up the ball again, in this case, right? The dog gives me... there are some dogs that give it, right?

Interviewer: So you call him back to...

Student: Uh-huh

Interviewer: Call to play again (write). Come on, so now this one is like this: grab the bubble, call the dog, throw the ball, the dog catches it and you call the dog with the ball to play again. Didn't it seem like this one (last step) was too long compared to the others?

Student: Yes.

Interviewer: So do you think you could break this one in two? And join two of these together? To make... because there have to be five, but the five should be more or less the same size. What do you think you can do?

Student: You can... It's not here. Just put "call the dog with the ball".

Interviewer: And then you don't play again? It's just ONE game, right? Okay then. That's it. Now it's a more difficult challenge. Can you think of five steps to play with your dog that aren't the five steps you mentioned just now?

Student: No.

Interviewer: Another joke.

Student: Pick up a stuffed animal, but I already put it there.

Interviewer: Okay, but then we're going to start another process. What do you do with the stuffed animal?

Student: I throw it for him to catch.

Interviewer: Same game? Okay. What else can you do with a dog to make him happy?

Student: Run.

Interviewer: Run. Okay, we can run...

Student: But I only know this.

Interviewer: Walk with him?

Student: Take a walk then.

Interviewer: Okay, let's write walking your dog in 5 steps. What do you have to do to walk your dog?

Student: Put on a collar.

Interviewer: Look, that's a good one! So you start by putting on the collar?

Student: (nods reluctantly)

Interviewer: Put on the collar (writes), and then?

Student: Get the dog. Can you put it first, get the dog? Call the dog?

Interviewer: Ahh! Yes, you can. Call the dog (writes).

Student: There's no way to put on the collar if the dog isn't there, right?

Interviewer: Oh, great! I'm glad you thought of that! It would be a problem! Okay, so you call the dog, put the leash on him... (looks at the student)

Student: Yeah... open the gate.

Interviewer: Open the gate (write), and then?

Student: Get out.

Interviewer: Go for a walk with him (writes).

Student: Can you put "close the gate"?

Interviewer: Can you? Before you leave?

Student: No. Then close the gate.

Interviewer: Oh, okay, you went out and then you close it. Okay, you called the dog, put on the leash, open the gate, go for a walk, close the gate... and that's it?

Student: Yeah... then come back too

Interviewer: Yeah, well, if you want to include "back" here, then maybe you're going to have to tone down the details here a bit.

Student: Oh no, huhum.

Interviewer: So, is that okay? And then I want you to think about the following: what if you didn't have a dog? If you had a cat, or a fish? Would taking care of it be different?

Student: Perhaps.

Interviewer: Perhaps?

Student: Perhaps yes.

Interviewer: So I want you to think about it like those things about layers of abstraction, about details. If we think about very few details, isn't it the same thing? It's the same thing for a dog, a cat or something. Oh, if it's hungry, go there and feed it. If it's dirty, you go there and bathe it, clean it. If it's sad or needy, you go there and play with it. It's the same thing for any animal, right?

Student: Yes.

Interviewer: . But then, how you feed a dog may be different from giving food to a cat or a fish, right? Let's see (go back to the steps for feeding)? If it were a fish, you would (reading) find the food, get the dispenser, put the food in the dispenser, put it in a little bowl... Oops! There are no little bowls for fish.

Student: Put it in the aquarium, right?

Interviewer: Putting it in the aquarium, okay. Just a little different, right? And calling the dog, you don't need to call the fish, right? So it's a little different. And what if... what if the fish is dirty?

Student: Fish don't get dirty.

Interviewer: Fish don't get dirty, what gets dirty is the aquarium water.

Student: AND.

Interviewer: You have to clean the aquarium water, or have something cleaning it for you. But what about playing? How do you play with a fish?

Student: There's no way to play with a fish.

Interviewer: Yes, at most you can keep hitting the aquarium, observing... right? But you saw that the process is different, right?

Student: Uh-huh.

Interviewer: So, that's basically it, okay? These are the challenges and activities here. I just wanted to ask you if you understood anything about this issue of abstraction layers, more details, fewer details... what would you say you learned? If you think you learned anything at all.

Student: I learned that more detailed is not having... What is the name?

Interviewer: Abstraction

Student: Abstraction! And with fewer details more abstract.

Interviewer: That's it! And you see... the main thing I wanted you to see is that when we're not worrying too much about details, some things seem the same, right?

Oh, every little animal there is an animal, but each one eats something different. That's when we have to look at it in more detail to deal with this. Every little animal has to eat when it's hungry, but each one eats something different, we feed them in a different way...

Student: Like the [green lizard].

Interviewer: That's just like the [green lizard]. So... that's it, okay? That's the activity! Thanks for participating!

Student: You're welcome!

Annexes

ANNEX A – Informed Consent Form

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS
CENTRO DE DESENVOLVIMENTO TECNOLÓGICO

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

O menor sob sua guarda está sendo convidado(a) a participar de um projeto de pesquisa. Após a leitura e esclarecimento sobre as informações a seguir, no caso de aceitar que o menor faça parte do estudo, **rubrique a primeira página e assine no final deste documento**, que está em duas vias. Uma delas é sua e outra é do pesquisador.

Projeto ExpPC - Explorando o Pensamento Computacional para a Qualificação do Ensino Fundamental

Pesquisadora responsável: Profa. Dra. Simone André da Costa Cavalheiro
Telefone para contato: (53) 3284 3860

Este projeto tem como finalidade criar uma rede educacional para consolidar o Pensamento Computacional no âmbito do ensino fundamental, o qual está sendo desenvolvido por professores e alunos dos cursos de Computação da Universidade Federal de Pelotas. Participarão deste projeto em torno de 300 alunos do ensino fundamental de escolas da rede municipal de Pelotas. Ao participar deste projeto, o menor sob sua responsabilidade irá realizar atividades didático-pedagógicas que abordam conceitos da Computação, as quais serão observadas e avaliadas por pesquisadores que compõem a equipe do projeto. Além das atividades, o jovem também receberá um questionário socioeconômico e cultural, que deverá ser respondido pelos pais ou responsáveis. O Senhor(a) tem a liberdade de se recusar a autorizar o jovem a participar e o jovem tem a liberdade de desistir de participar em qualquer momento sem qualquer prejuízo. No entanto solicitamos sua colaboração para que possamos obter melhores resultados no projeto. Sempre que o Senhor(a) e/ou o jovem queiram mais informações podem

entrar em contato diretamente com a pesquisadora responsável. A participação neste projeto não traz complicações legais de nenhuma ordem e os procedimentos utilizados obedecem aos critérios da ética na Pesquisa com Seres Humanos. O estudo apresenta riscos mínimos, pois o preenchimento do questionário socioeconômico poderá acarretar constrangimento aos participantes, podendo ser interrompido a qualquer momento. Todas as informações coletadas nesta investigação são estritamente confidenciais. Ao participar desta pesquisa, o jovem terá a oportunidade de desenvolver algumas habilidades que auxiliam na construção de um raciocínio lógico necessário para a solução de problemas. Você não terá nenhum tipo de despesa por participar deste estudo, bem como não receberá nenhum tipo de pagamento por sua participação. Após estes esclarecimentos, solicitamos o seu consentimento de forma livre para que o menor sob sua responsabilidade participe desta pesquisa. Para tanto, preencha os itens que se seguem:

CONSENTIMENTO LIVRE E ESCLARECIDO

Tendo em vista os itens acima apresentados, eu, de forma livre e esclarecida, autorizo o menor sob minha responsabilidade a participar deste projeto.

Nome do menor: _____

Nome do responsável: _____

Telefone do responsável: _____

Local e data: Pelotas, _____ de _____ de _____.

_____

Assinatura do Responsável

_____

Coordenadora do Projeto

ANNEX B – Image and Voice Terms

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS
CENTRO DE DESENVOLVIMENTO TECNOLÓGICO

TERMO DE CONSENTIMENTO PARA USO DE IMAGEM E VOZ

Projeto: ExpPC - Explorando o Pensamento Computacional para a Qualificação do Ensino Fundamental
Pesquisadora responsável: Profa. Dra. Simone André da Costa Cavalheiro

Eu, ... permito que o pesquisador relacionado acima obtenha fotografias e filmagens do menor sob minha guarda para fins de divulgação do projeto e pesquisa científica/educacional.

Concordo que o material e as informações obtidas relacionadas ao menor participante do estudo possam ser publicados no site do projeto, em vídeos de divulgação do projeto, aulas, congressos, eventos científicos, palestras, periódicos científicos ou afins.

As fotografias e vídeos ficarão sob a propriedade do grupo de pesquisadores pertinentes ao estudo e sob sua guarda.

Nome do menor: _____

Nome do responsável: _____

Telefone do responsável: _____

Local e data:

_____
Assinatura do Responsável pelo Participante da Pesquisa

ANNEX C – Interview Assessment by LLM

Overview F8:

The interview captures a range of competencies related to Layers of Abstraction (LoA), particularly how students engage with abstract and concrete concepts. While the Applicator introduces key abstraction principles, such as distinguishing between abstract and concrete elements and navigating between layers, the student's responses reflect varying degrees of grasp on these concepts.

1. **LoA Recognition**

- **1.1 Distinguish between abstract and concrete:** The Applicator explains the distinction between abstraction and concrete ideas using the student's drawings, e.g., drawing a dog with or without minute details. The student shows understanding when acknowledging that drawing every hair would be more concrete but that it's unnecessary. This subskill was strongly evident.

- **1.2 Connect multiple representations:** The task of grouping characters by abstraction, such as identifying those with more details (e.g., the mermaid princess), demonstrates the student's ability to switch between different representations. The student connects visual elements and degrees of abstraction. This was moderately visible.

- **1.3 Visualize in layers of abstraction:** The student's ability to recognize that characters can have multiple levels of detail and group them according to abstraction levels indicates competence in visualizing abstraction layers. This skill was somewhat evident during the interview.

- **1.4 Acknowledge information hiding:** This was weakly present. The concept of information hiding wasn't directly addressed or discussed by the student in any significant way.

2. **LoA Calibration**

- **2.1 Assess relevance of details:** The student understands when details are important or irrelevant, such as when organizing characters based on their main attributes (e.g., princesses and superheroes) rather than minor features. This skill was moderately apparent.

- **2.2 Navigate layers of abstraction:** The interaction involving the abstraction meter, where more abstract versions of characters are displayed with fewer details, shows the student's ability to navigate between different levels. This subskill became clear during the grouping task.

- **2.3 Evaluate layers of abstraction:** The student effectively differentiates characters based on the level of abstraction and groups them accordingly. This was evident during the tasks but not deeply explored.

- **2.4 Select the best layer of abstraction:** The decision to use a more detailed or less detailed view to organize characters indicates the student's emerging ability to select a suitable layer for a given task. This subskill was evident, though the student displayed some indecision.

3. **LoA Interaction**

- **3.1 Cross layers of abstraction:** The student's ability to recognize characters despite changes in abstraction level (e.g., identifying the firefighter puppy after shuffling) shows an understanding of how to move between abstraction layers. This subskill was apparent.

- **3.2 Identify how they are connected:** The student recognizes that the same character can appear at different abstraction levels, showing that they understand the connections between these layers. This subskill was clearly demonstrated.

- **3.3 Analyze the impact on others:** This subskill wasn't strongly evident. The tasks were focused on identifying abstraction levels rather than assessing the impact of abstraction on other characters or systems.

- **3.4 Replace layers of abstraction:** The interview did not show any significant interaction involving replacing one layer of abstraction with another. This skill was not evident.

4. **LoA Modeling**

- **4.1 Simplify by removing unnecessary details:** The student is introduced to simplification but hesitates to remove details in some cases, such as wanting to include more features when describing their dog. This subskill appeared somewhat but with reluctance.

- **4.2 Generalize recognizing patterns:** The student's ability to group characters based on abstract patterns (e.g., grouping by color or role) reflects a developing ability to generalize across abstractions. This subskill was moderately demonstrated.

- **4.3 Refine by adding necessary details:** When tasked with refining instructions for giving a bath or organizing characters based on colors, the student demonstrates the ability to add appropriate details where necessary. This subskill was visible but could be expanded further.

- **4.4 Decompose dealing with complexity:** The student was guided through decomposition tasks, such as breaking down steps for playing with a pet or organizing characters. The ability to decompose was demonstrated clearly during these tasks.

Summary of Strengths and Gaps:

- **Most evident subskills:** LoA Recognition (1.1), LoA Calibration (2.1, 2.2), and LoA Interaction (3.1, 3.2) were the most evident, as the student could differentiate between abstraction layers and connect representations.

- **Less evident subskills:** LoA Recognition (1.4 - Information hiding) and LoA Interaction (3.3, 3.4) were underexplored. Additionally, LoA Modeling (4.1 - Simplify) and (4.4 - Decompose) appeared but needed more support to be fully demonstrated.

The student's engagement with the concepts was solid but requires further development in more complex interactions across abstraction layers and refining their approach to simplification and decomposition.

Overview F11:

The interview demonstrates how the student engages with key competencies related to Layers of Abstraction (LoA) while interacting with a variety of tasks. The student shows an understanding of abstraction through practical examples and actively adjusts to different levels of complexity and detail. Below is a breakdown of each competency and its related subskills based on the student's responses.

1. **LoA Recognition**

- **1.1 Distinguish between abstract and concrete:** The student understood the concept of abstraction, recognizing that abstract representations have fewer details compared to more concrete ones. For example, the student correctly identified the most abstract card in a set as the one with fewer details【9†source】. This subskill was clearly demonstrated.

- **1.2 Connect multiple representations:** The student had some difficulty connecting different representations when the abstraction level changed. During tasks like identifying shuffled characters at different abstraction levels, the student was able to identify characters through place memory but struggled with the shift in abstract representation【9†source】. This skill was moderately evident.

- **1.3 Visualize in layers of abstraction:** The student showed some ability to visualize and differentiate between layers of abstraction. When sorting characters into groups (superheroes, animals, and princesses), the student opted for the middle abstraction layer, explaining that it made the task easier to differentiate characters【9†source】. This subskill was fairly evident.

- **1.4 Acknowledge information hiding:** This subskill appeared subtly when the student struggled to identify characters as the abstraction level changed. The notion that details could be hidden within higher levels of abstraction was not explicitly acknowledged by the student but was introduced by the Applicator. This subskill was less evident.

2. **LoA Calibration**

- **2.1 Assess relevance of details:** The student effectively assessed which details were relevant for different tasks. For example, when asked to separate characters by their main colors, the student chose the lowest abstraction layer to reveal the necessary visual details, showing an understanding of what was required for the task【9†source】. This subskill was well demonstrated.

- **2.2 Navigate layers of abstraction:** The student actively adjusted between layers depending on the task. The choice of using different abstraction layers to simplify or differentiate between characters shows that the student was able to navigate between levels as needed【9†source】. This was evident.

- **2.3 Evaluate layers of abstraction:** The student evaluated layers based on the complexity of the task, as shown when selecting the middle abstraction layer for grouping characters. This decision was made based on the clarity provided by the layer's details【9†source】. This subskill was demonstrated but could benefit from further reflection on why certain layers were easier for specific tasks.

- **2.4 Select the best layer of abstraction:** The student displayed this subskill when choosing the middle abstraction layer for organizing characters, explaining that it was simpler to differentiate between the categories using that level of detail【9†source】. The decision-making process was clear, so this subskill was effectively demonstrated.

3. **LoA Interaction**

- **3.1 Cross layers of abstraction:** The student crossed layers effectively, especially when performing tasks such as identifying characters after the abstraction level was shifted. The student was able to adapt to varying levels of detail【9†source】. This subskill was evident.

- **3.2 Identify how they are connected:** The student recognized the connection between abstraction levels and the characters' identities, acknowledging that they were the same characters despite fewer details being shown at higher abstraction levels【9†source】. This subskill was demonstrated.

- **3.3 Analyze the impact on others:** The interview did not emphasize this subskill, as the focus was more on individual problem-solving and recognition, rather than understanding how abstraction impacts other elements in the system. This subskill was not evident.

- **3.4 Replace layers of abstraction:** While the student didn't actively replace one layer with another, the process of selecting and shifting between layers during the interview hints at a developing understanding of when to apply different layers. This subskill was weakly demonstrated.

4. **LoA Modeling**

- **4.1 Simplify by removing unnecessary details:** The student demonstrated simplification during the "pet care" task, describing a five-step process for feeding or bathing a pet. The student struggled initially to include enough detail, indicating a natural tendency toward abstraction【9†source】. This subskill was evident but could benefit from more practice.

- **4.2 Generalize recognizing patterns:** The student generalized well when grouping characters into broad categories such as superheroes, animals, and princesses. The recognition of these patterns across abstraction layers indicates a solid ability to generalize【9†source】. This subskill was demonstrated.

- **4.3 Refine by adding necessary details:** When asked to refine the description of feeding a pet, the student was able to add more steps to the process, showing an understanding of when to add necessary details【9†source】. This subskill was well demonstrated.

- **4.4 Decompose dealing with complexity:** The student successfully decomposed tasks, such as feeding or bathing a pet, into smaller steps. However, there was some hesitation in providing a sufficient number of details at the beginning, indicating a need for further development in dealing with complexity【9†source】. This subskill was evident but could be strengthened.

Summary of Strengths and Gaps:

- **Most evident subskills:** LoA Recognition (1.1), LoA Calibration (2.1, 2.2), LoA Interaction (3.1, 3.2), and LoA Modeling (4.1, 4.3) were well demonstrated. The student was proficient at distinguishing between abstraction layers, navigating them, and simplifying tasks.

- **Less evident subskills:** LoA Recognition (1.4 - Information hiding), LoA Interaction (3.3, 3.4), and LoA Modeling (4.4 - Decompose) appeared weakly or not at all. These areas, particularly understanding how abstraction impacts interconnected elements and dealing with complexity, need further development.

Overall, the student shows a solid understanding of abstraction concepts but could benefit from more explicit tasks that focus on information hiding, interconnection impacts, and decomposing complexity into finer details.

Overview M12:

The interview reveals how the student navigates competencies related to Layers of Abstraction (LoA) through a series of guided tasks. The student demonstrates an understanding of abstract versus concrete concepts, occasionally needs prompts, and provides explanations for choices. Below is a detailed breakdown of each competency and its subskills based on the student's responses.

1. **LoA Recognition**

- **1.1 Distinguish between abstract and concrete:** The student quickly grasped the difference between abstract and concrete representations, correctly identifying the more abstract characters as those with fewer details【14†source】. The explanation that “this one has the most details and the other one has the least” was accurate, showing clear understanding. This subskill was evident.

- **1.2 Connect multiple representations:** The student was able to connect different layers of abstraction during tasks such as organizing characters by type (princesses, superheroes, and animals) and by color. The transition between abstract and more concrete layers was managed with ease, although some guessing occurred initially【14†source】. This subskill was moderately demonstrated.

- **1.3 Visualize in layers of abstraction:** While the student successfully visualized characters at different abstraction levels, there was occasional hesitation about which layer to use when organizing characters. Once prompted, the student used layers effectively based on task needs【14†source】. This subskill was visible but could benefit from more proactive layer-switching.

- **1.4 Acknowledge information hiding:** The idea of information being hidden in higher abstraction layers was touched upon but not deeply explored by the student. They understood that different layers provide different amounts of information, but did not explicitly acknowledge the concept of information hiding【14†source】. This subskill was weakly evident.

2. **LoA Calibration**

- **2.1 Assess relevance of details:** The student demonstrated the ability to assess which level of detail was appropriate for different tasks, such as organizing characters by color. The choice to use certain abstraction layers for specific problems indicates a good sense of when details were necessary【14†source】. This subskill was clearly evident.

- **2.2 Navigate layers of abstraction:** The student was comfortable navigating between layers of abstraction when prompted, though they sometimes forgot about the existence of other layers. When asked why they didn't switch to different layers, the student admitted to forgetting but generally adjusted when reminded【14†source】. This subskill was demonstrated, though it could be more naturally integrated.

- **2.3 Evaluate layers of abstraction:** When organizing characters, the student selected layers that were neither too detailed nor too abstract, showing an understanding of which layers provided the most useful information for the task at hand【14†source】. This subskill was evident.

- **2.4 Select the best layer of abstraction:** The student effectively selected the middle abstraction layer for organizing tasks, explaining that it was easier to work with without being overwhelmed by details. This shows an ability to identify the most efficient abstraction layer【14†source】. This subskill was well demonstrated.

3. **LoA Interaction**

- **3.1 Cross layers of abstraction:** The student frequently crossed between abstraction layers during tasks like organizing characters and solving the invitation puzzle, though they sometimes needed reminders to switch layers when appropriate【14†source】. This subskill was evident.

- **3.2 Identify how they are connected:** The student understood that the same character appeared differently across abstraction layers, recognizing that they were still dealing with the same entities regardless of how much detail was visible. The Applicator reinforced this concept, which the student grasped【14†source】. This subskill was well demonstrated.

- **3.3 Analyze the impact on others:** The interview did not focus heavily on the student analyzing the impact of abstraction changes on other elements in the system. This subskill was not clearly evident.

- **3.4 Replace layers of abstraction:** The student did not explicitly replace one layer with another but did shift between abstraction layers when organizing characters or performing tasks. This subskill was weakly demonstrated, with more support from the Applicator.

4. **LoA Modeling**

- **4.1 Simplify by removing unnecessary details:** The student effectively simplified tasks when asked to describe processes like feeding or bathing a pet. Initially, they tended to give fewer details but adjusted when prompted to break down the steps further【14†source】. This subskill was evident, though the student needed some guidance.

- **4.2 Generalize recognizing patterns:** The student demonstrated pattern recognition when asked to create groups based on color and character type. They applied general rules to categorize items, showing a solid ability to generalize across abstraction layers【14†source】. This subskill was well demonstrated.

- **4.3 Refine by adding necessary details:** The student refined their descriptions when asked to add more detail to the processes, such as expanding the steps involved in feeding or bathing a pet. They initially provided minimal details but understood how to refine and adjust their descriptions to meet the task's needs【14†source】. This subskill was evident.

- **4.4 Decompose dealing with complexity:** The student decomposed tasks like feeding, bathing, and playing with a pet into smaller, manageable steps. They occasionally struggled to break down the processes into enough detail, indicating room for improvement in dealing with complexity【14†source】. This subskill was moderately evident.

Summary of Strengths and Gaps:

- **Most evident subskills:** LoA Recognition (1.1), LoA Calibration (2.1, 2.2, 2.4), LoA Interaction (3.1, 3.2), and LoA Modeling (4.1, 4.2) were the most evident. The student showed proficiency in distinguishing between abstraction layers, navigating them for different tasks, and simplifying processes when necessary.

- **Less evident subskills:** LoA Recognition (1.4 - Information hiding), LoA Interaction (3.3 - Impact analysis), and LoA Modeling (4.4 - Decompose) were less evident. These areas need more attention, especially when it comes to understanding hidden information in abstraction layers and breaking down complex tasks into detailed steps.

Overall, the student demonstrated a solid understanding of abstraction principles, though more focus on dealing with complexity and understanding hidden information within layers could help enhance their skills.

Overview F10:

The interview transcript reflects how the student interacts with the concept of Layers of Abstraction (LoA), especially when introduced to it for the first time. The student displays varying levels of competency across the four main LoA skills and their subskills. Here's a breakdown based on the competencies requested.

---

1. **LoA Recognition**

1. **Distinguish between abstract and concrete (1.1)**: The student struggled initially but gradually recognized differences between abstract and concrete representations when prompted by the applicator's examples. They eventually understood that fewer details represent higher abstraction levels (e.g., identifying fewer details as more abstract).

2. **Connect multiple representations (1.2)**: The student was moderately able to connect the multiple levels of abstraction when switching between layers in the game (e.g., recognizing the same character across layers), though they hesitated when shifting from one representation to another.

3. **Visualize in layers of abstraction (1.3)**: This was somewhat underdeveloped. The student required frequent reminders to switch layers and recognize how the characters stayed the same despite abstracting different details.

4. **Acknowledge information hiding (1.4)**: This competency was less evident, as the student didn't explicitly reflect on what details were being hidden at each abstraction level without the applicator's prompts. They needed guidance to realize which characteristics were concealed.

Most Evident Subskill: Distinguishing between abstract and concrete (1.1)

Less Evident Subskill: Acknowledging information hiding (1.4)

---

2. **LoA Calibration**

1. **Assess relevance of details (2.1)**: The student occasionally assessed the relevance of details, especially when the applicator asked them to distinguish between characters based on the level of abstraction. However, they showed difficulty in independently determining which details mattered the most.

2. **Navigate layers of abstraction (2.2)**: While able to switch between layers with guidance, the student was hesitant to initiate layer navigation independently. They often stuck to one level and needed prompting to shift abstraction layers.

3. **Evaluate layers of abstraction (2.3)**: The student struggled to evaluate which abstraction layer best suited the task at hand (e.g., when separating characters by color or role, they preferred concrete levels but didn't articulate why).

4. **Select the best layer of abstraction (2.4)**: The student generally defaulted to concrete layers when organizing, even when the applicator suggested that an intermediate layer might be more efficient. They seemed reluctant to consider abstraction layers that were more abstract.

Most Evident Subskill: Assess relevance of details (2.1)

Less Evident Subskill: Select the best layer of abstraction (2.4)

---

3. **LoA Interaction**

1. **Cross layers of abstraction (3.1)**: The student demonstrated the ability to cross abstraction layers when switching between more concrete and abstract views of characters, though they needed prompting.

2. **Identify how layers are connected (3.2)**: The student recognized connections between layers when explicitly asked (e.g., linking the more abstracted character representations to the detailed ones). However, they didn't independently articulate these connections.

3. **Analyze the impact on others (3.3)**: This subskill was not addressed in the interview. The student did not discuss how changing abstraction levels might affect others.

4. **Replace layers of abstraction (3.4)**: While they participated in exercises that involved switching layers, the student didn't express an understanding of when or why one layer should be replaced by another.

Most Evident Subskill: Cross layers of abstraction (3.1)

Less Evident Subskills: Analyze the impact on others (3.3), Replace layers of abstraction (3.4)

---

4. **LoA Modeling**

1. **Simplify removing unnecessary details (4.1)**: The student showed a limited ability to simplify tasks (e.g., feeding or playing with a pet) into concise steps. When challenged to break down processes, they initially struggled, but they could simplify when guided.

2. **Generalize recognizing patterns (4.2)**: The student began to generalize the process of caring for pets by acknowledging that different animals have similar basic needs (e.g., feeding and playing), but their understanding of generalization was still developing.

3. **Refine adding necessary details (4.3)**: While the student was prompted to refine explanations by adding details (such as breaking down steps into smaller parts), they seemed to struggle with determining when and where additional details were necessary.

4. **Decompose dealing with complexity (4.4)**: The student had trouble decomposing complex processes like feeding or playing into manageable steps without guidance. When guided, they provided a basic decomposition, though it lacked complexity.

Most Evident Subskill: Simplify removing unnecessary details (4.1)

Less Evident Subskill: Decompose dealing with complexity (4.4)

Conclusion:

Overall, the student demonstrated basic recognition and interaction with layers of abstraction when guided but showed limited independent capability in higher-level skills such as calibration and modeling. Their responses suggest a foundational understanding, with stronger skills in distinguishing between abstract and concrete representations (LoA Recognition) and crossing abstraction layers (LoA Interaction), while areas like refining details and selecting appropriate abstraction layers require more development.