PACT: A Citizen Science Project for Computing Education

Braz Araujo da Silva Junior*, Simone André da Costa Cavalheiro†, Luciana Foss‡,
Ana Marilza Pernas Fleischmann§ and Andrea Volz Garcez¶
Universidade Federal de Pelotas
Pelotas, Brazil
*badsjunior@inf.ufpel.edu.br; †simone.costa@inf.ufpel.edu.br; ‡lfoss@inf.ufpel.edu.br;
§marilza@inf.ufpel.edu.br; ¶andreavolzg@gmail.com

***Abstract*—This innovative practice full paper proposes a model for centralizing, categorizing and organizing Computational Thinking resources and knowledge. The term "Computational Thinking" usually refers to problem solving skills related to Computer Science. Yet, there is neither: a consensual definition of what exactly is this term about; nor which skills are involved. Even under these conditions, several studies are conducted regarding this theme around the world, as well as informal activities applied by non-researchers enthusiasts. Thus, a model to reunite both, academic and non-scientific community, around the Computational Thinking is proposed: the Partnership for Advancing Computational Thinking, PACT. On one hand it is an effort to guide convergence in the literature, leading to more standardized definitions and spotting similarities or disagreements between authors. And on the other hand, it is a platform for sharing resources and experiences between the communities, an initiative to bring the model of research with public participation (Citizen Science) to education. An expandable ontology that is able to consider the perspective of multiple authors over the subjects is used to implement the model. Three different author views and resources are instantiated to illustrate the range, expressiveness and operation of the ontology. It is revealed able to model author views from previous models, frameworks and even discursive texts, as well as linking resources of different natures by their purpose and subjects approached.**

***Index Terms*—Computer Science, Computational Thinking, Citizen Science, Ontology.**

I. INTRODUCTION

Computational Thinking (CT) is a term that has been increasingly found in a cross of two fields: **Computer Science (CS)** and Education. Even though it has no consensual definition or model [1], CT research is kept between these fields, featuring topics as: the introduction of technology in the classroom [2]; programming [3] and robotics [4]; CS for beginners/basic CS [5]; and general purpose skills for problem solving [6]. This wide range contributed to the consolidation of CT in the scientific community. As shown in Fig. 1, CT has consistently increased in both, number of citations and publications over the last two decades.

Although having different views from multiple authors, the ongoing discussion about "what CT really is" did not disrupt the CT research. Instead, it fostered even more research by turning CT into an umbrella term [7]. Some authors proceeded the discussion in a philosophical manner, either presenting the main ideas [8] or making a more in-depth discussion of its

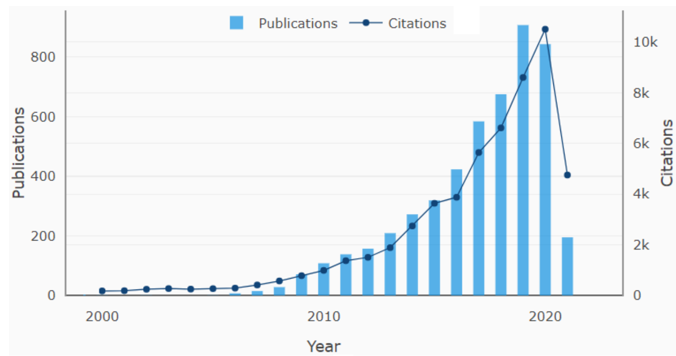

Fig. 1. Numbers of citations and publications on the Computational Thinking topic according to Microsoft Academic. Source: https://academic.microsoft.com/topic/2780368719/publication/

nature [9]. While others have targeted operationalization [10], working on theoretical models [11] and frameworks [12]. On the practical side, some tools have been built specifically to develop CT [13] and to assess it [14]. Furthermore, many experiments and case studies using other educational tools have been reported targeting CT [15]–[17].

On the other hand, all this variety of tools, diversity of approaches and multiple definitions may look messy and confuse, leading even specialists to call CT a "blurry" construct [7]. Given that one of the challenges CT researchers face are the misconceptions of teachers about the term [2], having no convergence for introducing it to newcomers becomes a problem. Flooding CT enthusiasts with many different conceptions and resources without a proper way of mapping and relating them will get them lost and directionless. Dealing with it would require a platform that is not just a searchable database to centralize, categorize and organize the storage of all kinds of resources towards CT, as already suggested [18], but something on top of it to guide newcomers and situate specialists.

This centralizing effort could reunite CT researchers, teachers, learners, practitioners and overall enthusiasts, which is an opportunity to explore what those communities has to offer to one another. People out of academic environments could get in touch with products of scientific researches, such as tests [7] and educational games [13], that wouldn't be normally found without reading scientific papers. And academics could get useful data from non-scientists. Borrowing the concept of scientific studies with public participation (popularized as Citizen Science), people around the world can acquire and generate data for research using a collaboration network [19].

Teachers may benefit from lesson plans, activities and assessment methods developed by researchers to effectively bring CT to their class. In return, they may contribute with guided experience reports, providing case studies to researchers. Learners may benefit from theoretical explanations, visual models, tools and tutorials to understand and develop CT. In return, they may contribute with linked surveys and feedback, providing user experience data to researchers. Beyond all these data for empirical analysis, scientists may further benefit from the centralization of resources: avoiding reinventing the wheel, building improvements on existing achievements, and identifying similarities, gaps, divergences, inconsistencies, trends and points of interest for future research [18].

In order to materialize this envisioned approach, the **Partnership for Advancing Computational Thinking (PACT)** is proposed. A model able to capture and integrate multiple perspectives on CT, while being capable of organizing all kinds of resources that could interest the CT academic community or CT enthusiasts out of scientific environment. The core idea of the model is represented in Fig. 2, a main structure (in gray) will define generic classifications to organize resources in a database (in green), while reuniting all CT subjects approached by the stored resources (this would expand as new subjects are found). Instead of arbitrarily defining what is CT, it would be multiply defined by various authors, by establishing hierarchies and relations between the existing subjects (in blue).

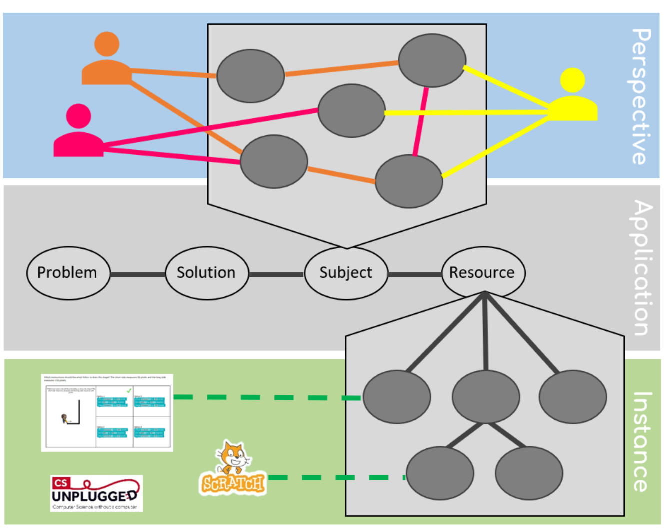

Fig. 2. Overview of the structures of OntoPACT.

The data that PACT needs to deal with is quite singular. Abstraction, sequences, parallelism, these are some of the subjects often approached in the context of CT. Lesson plans, educational games, questionnaires, these are some of the resources often wanted in the context of CT. Arguably, PACT deals with very generic, abstract concepts (including abstraction itself) at the same time it is dealing with specific, concrete things. Since PACT floats at this level of conceptualization while aiming for reaching common, clear descriptions, it is fit for the use of Ontologies, defined as the "shared explicit specifications of conceptualizations" [20].

Ontologies stand out as a convenient solution for representing knowledge in computer systems (like for a platform or a database), since they are used as shared vocabularies applied for integration and interoperation. An ontology for the PACT purposes would be able to normalize the different views and allow machine-interpretable data integration. Thereby, PACT is implemented using an ontology: the **OntoPACT**.

To illustrate the coverage and capacity of integration of OntoPACT, there are given examples of perspectives of three CT authors, whose source publications differ in nature and purpose: some influential discursive viewpoints [8], [21]; a framework centered on Scratch[1] [12]; and a model based on a systematic review [11]. Moreover, three resources, also different in nature and purpose, are instantiated: a CT skills assessment questionnaire [14]; an unplugged[2] activity; and a Scratch[1] project.

The rest of this paper is organized as follows. Section II relates other ontologies for CT. Section III discusses definitions of CT, detailing those being used as example perspectives and how they were modeled. Section IV describes the OntoPACT, its components, their relations and classifications. Section V presents the instantiation of OntoPACT, showing how the different perspectives and resources interact in a working ontology. Section VI brings the conclusions of the study.

II. Related Works

Ontologies have been used in Computer Systems for standardization of terms and interoperation for a long time, but few initiatives related to their application in CT have been found. A related research is OntoScratch [23], which proposes an ontology-based knowledge representation to characterize Scratch project. In this work, the authors explores inference rules written in SPARQL[3] query language aiming at representing learning activities designed in Scratch and identify student's level of knowledge according to CT skills' evaluation. The authors evaluate their inference rules relating the result generated by OntoScratch with the analysis of the tool Dr. Scratch[4]. It is a work focused on Scratch, exploring ontologies to represent the environment and the activities developed by the students, not proposing to represent concepts on CT. About that, the authors argue the lack of standardization in

[1]Scratch is a block-based visual programming language developed by the Lifelong Kindergarten group, for more info, access: https://scratch.mit.edu/

[2]"Unplugged" is a term coined to refer to a computer science approach that does not require a computer machine [22]

[3]https://www.w3.org/TR/rdf-sparql-query/

[4]http://www.drscratch.org/

the definition of concepts in CT, which makes difficult the evaluation between different tools.

In the search for works related to data standardization and ontologies for CT, an ontology called "Micas, the Web Platform to Support Teachers of Computing at School" [24] from a master's dissertation from the University of Minho was found. Micas is a web platform to support teachers in computer education through the collection and classification of learning resources to be used by teachers in computer classes. Underlying the web platform, Micas provides an ontology, used to categorize and research learning resources by area and by year of teaching. It arose from the idea of creating a common curriculum basis for teaching CT from the first years of teaching. While at Micas the author's idea is to assist teachers in the search for resources to be applied in the classroom for teaching computational thinking, our proposal is to link resources to a standardization between the terms that are used by the various authors and make an intersection between what each author means with each term.

III. Computational Thinking

Synthesizing the concept, "Computational Thinking's essence is thinking like a computer scientist when confronted with a problem." [25]. Going deeper, "Computational Thinking is the mental skills and practices for designing computations that get computers to do jobs for us, and explaining and interpreting the world as a complex of information processes." [9]. This abstract idea of solving problems using things related to CS is a commonplace for CT. But when authors proceed to detail what exactly is involved, they start to diverge. Some head towards programming and computing skills, while others head towards general purpose skills, but each author presents its own list of skills [18].

A. Concepts - Wing

With no intention of presenting a model at all, Jeannette Wing has written a couple of very influential viewpoints [8], [21], which grant her the credit for reviving the term and bringing back the discussions around CS education within the academic community. Heavily cited in CT studies, these viewpoints abstractly conceptualize the term, advocate its importance and present everyday examples. Given Wing's relevance for the field, her viewpoints were selected to base a model for representing her perspective over CT in the OntoPACT.

It is noteworthy that the result is bound to bias and interpretation, since it is extracted from discursive texts with persuasive goals, rather than systematic representation of the concept. Addressing to that, each subject modeled is shown in **bold** inside an adaptation of a text from the source publication. These subjects are listed as follows:

- CT involves **Understanding Human Behavior**, **Designing Systems** and **Solving Problems** drawing on **CS Concepts** [8, §3]
- **Solving Problems** might address, or lead to **Efficiency/Optimization** and **Difficulty Analysis**, which accounts for underlying **Computational Models/Methods** [8, §4]
- Several **Evaluation** questions might lead to **Efficiency/Optimization** [8, §5]
- CT is **(Re)formulating Difficult Problems** into one we know how to solve [8, §5]
- CT is using **Abstraction** and **Decomposition** when attacking such problems [8, §7]
- CT is using/applying various specific **CS Concepts**, including in everyday life [8, §6–11]
- **Abstraction** is the essence of CT [21, §3]
- An **Algorithm** is an **Abstraction** of an step-by-step procedure for taking input and producing some desired output [21, §4]
- The **Abstraction** process introduces **Layers** [21, §6]
- Computing is the **Automation** of our **Abstraction**s [21, §9]

B. Concepts - Brennan

With the intention of presenting a model fit for a specific environment (but not restricted to it), Brennan and Resnick present a framework of CT centered on programming, specially in Scratch [26]. For them, CT "involves three key dimensions: computational concepts (the concepts designers employ as they program), computational practices (the practices designers develop as they program) and computational perspectives (the perspectives designers form about the world around them and about themselves)." [12]. These concepts, practices and perspectives are listed as follows:

- **Concepts**: sequences, loops, parallelism, events, conditionals, operators, and data.
- **Practices**: being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing.
- **Perspectives**: expressing, connecting and questioning.

C. Concepts - Silva Junior

With the intention of presenting a broad model covering most of what CT studies addresses, Silva Junior build a model to organize several terms found through a systematic literature review that has the following research question: "which concepts, subjects or skills are being related to CT in the literature?" [11]. The model divides the concept of CT into "lines", namely: abstraction, decomposition, algorithm, data, automation and evaluation. Each line touches several points and they eventually intersect. For OntoPACT, The lines themselves are modeled as subjects, as well as the points touched by each line, listed as follows:

- **Abstraction**: Generalization, Pattern Recognition, Modeling/Designing.
- **Decomposition**: Reuse/Remixing, Emergence and Recursion.
- **Algorithm**: Functions/Procedures, Flow of Control and Parallelism.

- **Data**: Data Collection/Gathering, Data Visualization and Data Structures.
- **Automation**: (Software) Development, (Hardware) Tinkering, Simulation.
- **Evaluation**: Optimization, Test, Debug.

IV. OntoPACT

As presented in the Introduction, PACT was proposed for leading to a consensus or more standardized definitions related to CT, in order to enable platforms for sharing resources. Thus, ontologies were explored to define a layer for standardization and interoperation of data, giving birth to OntoPACT.

Ontology consists of the science that studies nature, beings, metaphysics. While several areas of science deal with entities belonging to specific domains of knowledge, ontologies deal with entities from different domains, studying their interrelationships, and also with entities recognized for being of common sense [27].

In CS, ontologies are in general used to provide computer systems with meta-knowledge. There are several concepts for the term ontology, but one of the most widespread define ontology as an explicit specification of a "conceptualization" [28], or as a formal explicit shared conceptualization [29].

The use of ontologies for the semantic description of a vocabulary, despite being a complex process and requiring an in-depth study about the domain to be described, provides a broad understanding of the characteristics and properties of the classes belonging to the study domain, as well as the form that they relate to each other. An ontology expresses, for a particular domain, the set of terms, entities, objects, classes and the relationships between them, providing formal definitions regarding the terms and axioms, which restrict interpretation of these terms [30].

The ontological layer proposed in this work was designed to be offered on a platform, being possible to be accessed as a web service. Thus, its development was based on the integration with Semantic Web, where they are responsible for making the contents available from machine-readable to machine-understandable [27]. Ontologies are used in the Semantic Web as taxonomies, formed by a set of inference rules [31]. The ontologies were created using **Web Ontology Language (OWL)**[5] through the framework Protégé[6], in which all vocabulary classes are defined as subclass of "Thing".

In order to allow different CT author's views, multiple **High Level Ontologies (HLO)** are used to represent the views. Then, they are imported by a **Low Level Ontology (LLO)**, the application ontology. Thus, it is possible to include new concepts in the model while maintaining its reuse and respecting different possible approaches to CT. The underlying idea is represented in Fig. 2: an application layer (in gray), use the LLO to model the CT subjects without organizing them, as well as the resources without instantiating them; a perspective layer (in blue), use HLO to organize the CT subjects according to the respective authors; and an instance layer (in green), use a database to allocate the resources, modelled by the LLO.

To start OntoPACT, three HLO were developed, representing the views of Wing [8], [21], Brennan [12] and Silva Junior [11], presented in the Section III. These three top ontologies are presented, respectively, in Fig. 3, Fig. 4 and Fig. 5, reflecting their concepts explained in Section III A, B and C.

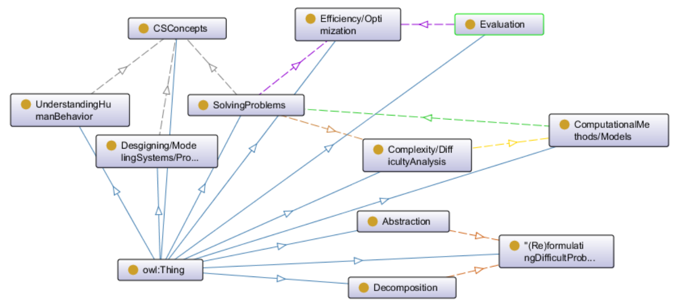

Fig. 3. High level ontology representing concepts discussed by Wing.

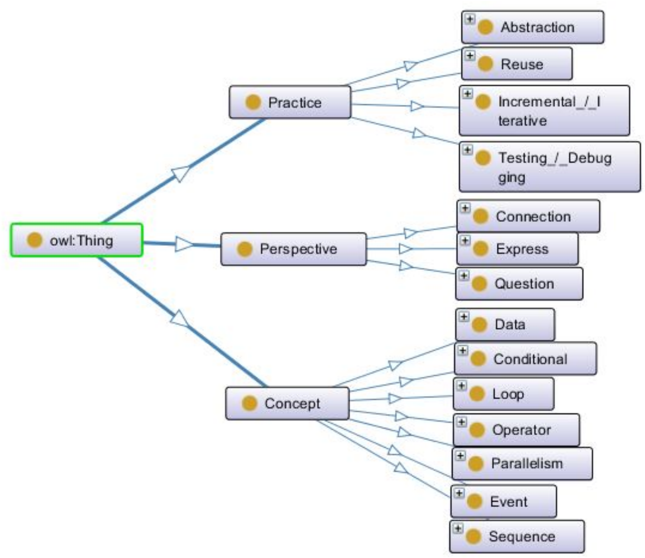

Fig. 4. High level ontology representing Brennan's framework.

While the task of modeling the concept of CT itself is left for the HLO, the LLO settles base concepts for organizing, classifying and relating CT resources that could be shared in the platform. For connecting the layers, the LLO presents a **Subject/Skill** class. This class have several unorganized children (subclasses), whose organization will be given by each author's HLO. For instance, the LLO has the classes Abstraction and Algorithm just hanging in there as children of Subject/Skill. The HLO will situate these classes: Wing will define that Algorithm is an Abstraction, which is the essence of CT; Silva Junior will define that both, Algorithm and Abstraction, are Lines of CT; and Brennan will define that Abstraction is a CT Practice, not directly mentioning

[5]https://www.w3.org/OWL/
[6]https://protege.stanford.edu/

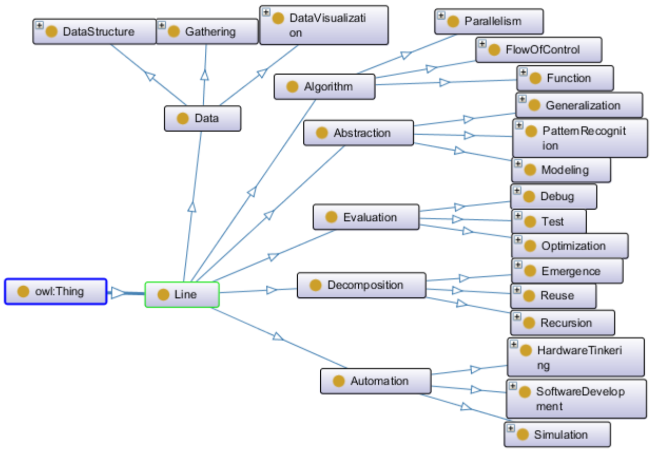

Fig. 5. High level ontology representing Silva Junior's model.

Algorithm. The the core of this model is the Subject/Skill class and its children, a non-exhaustive expansible class, that might grow as the CT literature advances, incorporating more subjects.

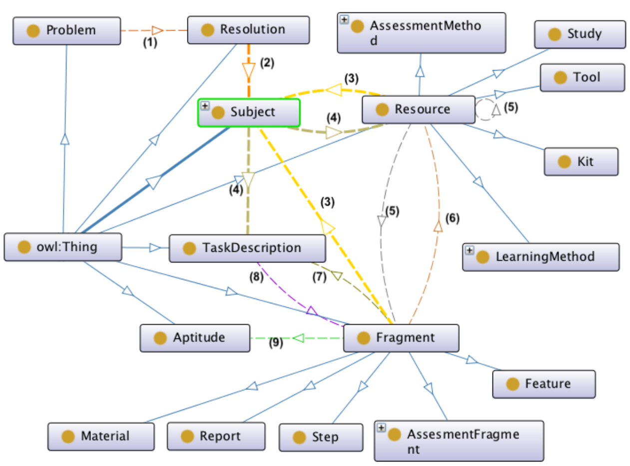

Fig. 6. Classes and relationships of OntoPACT ontology.

The following paragraphs present the other classes of the LLO (highlighted in **bold**) and their relationship (highlighted in *italic* followed by a number in parenthesis), which can be seen in Fig 6. Starting with a **Problem**, that *requires* (1) a **Solution**, since problem solving is a commonplace for CT, as discussed in Section III. The Solution, in turn, may *be reached through* (2) the Subject/Skill, which may *be approached by* (4) a **Resource**. This relation has a reverse, i.e. Resource *approaches* (3) the Subject/Skill. Resource is a class to generalize any given content that could be of interest to share in a CT-focused platform. This class has a group of children classes (presented in Fig 7) to classify these contents. For each child there is a definition to serve as criterion to classify the resource, listed as follows:

- **Teaching/Learning Resource**: A resource directed towards the teaching/learning processes.
  - **Lesson Plan**: A guide for conducting educational interventions in formal environments, with the necessary information for applying an activity in classroom, including what to teach and how to assess it (like sheets with information about subject, course, topic, duration, objectives, tasks and references).
  - **Activity Guide**: A detailed description of an activity, with necessary info for replicating it, including required materials and steps to set it up.
  - **Documentation**: A series of support text about an artifact, describing what each part does and how to use it (like an user guide or instruction manual).
  - **Tutorial**: A step-by-step guide, demonstrating a process for achieving a specific goal or reaching a specific output (like a "how-to").
  - **Others**: All other assessment resources that does not fall into the other categories.
- **Assessment Resource**: A resource directed towards the assessments.
  - **Questionnaire**: A data gathering resource without evaluating concerns upon the interrogated knowledge or skill, but centered towards socioeconomic aspects, opinions or other qualitative concerns (like a survey).
  - **Exam**: A test for evaluating the interrogated knowledge or skill (like a course's final exam).
  - **Interview**: A scripted series of questions for gathering non-edited declarations and information of the interrogated towards a given topic/subject.
  - **Rubric**: Guidelines to assist a qualitative assessment. "a scoring guide used to evaluate the quality of students' constructed responses—for example, their written compositions, oral presentations, or science projects. A rubric has three essential features: evaluative criteria, quality definitions, and a scoring strategy." [32]
  - **Score System**: Guidelines to assist a quantitative assessment, locating the person being assessed at a given scale (like the set of rules and parameters that define the final score of a game).
  - **Observation**: Any resource to assist observational assessment (like checklists and forms with items of interest).
  - **Others**: All other assessment resources that does not fall into the other categories.

Questionnaires, Exams and Interviews (eventually Others), may be composed by the following kind of questions:

- **Closed-ended**, like true or false and multiple choice questions, assessed through rigid answers.
- **Open-ended**, like an essay or a short-answer question), assessed through flexible answers.
- **Developing**, i.e. activities or assignments where there is a development to reach a final answer, like math problems and logical challenges, assessed through performance.

- **Study**: Detailed investigations and analysis of a subject or situation (like scientific papers and informal empirical reports).
- **Tool**: Specific digital artifacts (like software and video games) or concrete artifacts (like hardware, robots and board games).
- **Kit**: Generic, common objects (like paper and scissor) required by an activity or used together with a mean.

In order to explore the Resources with more precision, specially upon which Subjects/Skills they approach, Resources are *composed by* (5) other Resources or **Fragments**. Another relation with a reverse, Fragments *may compose* (6) a Resource, as shown in Table I. Fragments *materialize* (7) a **Task/Description**, which may *be materialized by* (8) multiple Fragments, as seen in the case study presented in the next section. At last, Fragments *require* (9) an **Aptitude**, that is a definition or estimation of its difficulty.

TABLE I
ONTOPACT RESOURCE-FRAGMENT CORRESPONDENCE.

Resource	**Fragment**
Teaching/Learning Resource	Teaching/Learning Fragment
Lesson Plan Activity Guide Tutorial	Step
Documentation	Instruction
Assessment Resource	Assessment Fragment
Questionnaire Exam Interview	Closed-Ended Question Developing Question Open-Ended Question
Rubric	Criterion/Standard
Score System	Rule/Parameter
Observation	Note
Study	Section
Tool	Feature/Asset
Kit	Material

For the application, the LLO imports and reuses the vocabulary from the HLO. Fig. 7 presents the classes and subclasses of the resulting application ontology created after importing the three aforementioned HLO. Note that, as the HLO of Brennan and Silva Junior were imported, the classes **Concept**, **Perspective**, **Practice** (from Brennan) and **Line** (from Silva Junior) are present in the full application ontology (in Fig. 7 their subclasses are minimized). In the full model, they have the same hierarchical level of the class **Subject** and were defined as *Equivalent* to Subject. Thus, the individuals of Subject have the same object properties of the individuals of the subclasses of Concept, Perspective, Practice and Line. Because Wing's HLO does not provide an hierarchy organizing CT Subjects, no additional class need to appear in the full model, the Subject class is enough to cover her HLO.

V. CASE STUDY

Three resources of different nature (but with some intersections) were chosen to exemplify how the PACT model could not just model them, but highlight their similarities and differences. The chosen examples are: the **Computational Thinking Test (CTt)**[7], an instrument for assessing CT skills that has provided evidences of reliability and validity [14]; the **Computer Science Unplugged (CSU)**[2], a project that provides "a collection of free teaching material that teaches CS through engaging games and puzzles that use cards, string, crayons and lots of running around" [33]; and **Scratch**[1], a drag-and-drop programming language coupled with a website to sustain a community for sharing and remixing projects [26].

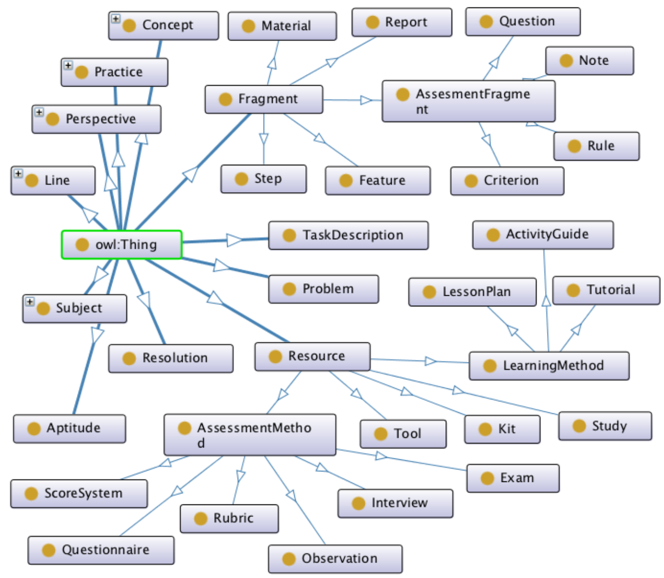

Fig. 7. Classes and subclasses of OntoPACT ontology

For each of these, there are a number of examples of Fragments: 6 questions for CTt; 4 steps of the KidBots[8] Activity Guide, for CSU; and 4 steps of the CATS![9] project, for Scratch. KidBots[8] is an activity where the student is led to "control" another by writing him a sequence of directions. While CATS![9] is a project where the student is led to create an application that endlessly creates walking cats in the screen.

These examples are presented as OntoPACT instances in the following subsections, graphically displaying it in Fig. 8 and reusing the notation from the previous section: classes are in **bold**; relations in *italic*; and instances are inside **[brackets]**.

A. Computational Thinking Test

The **[CTt]** (orange in Fig. 8) is an instance of **Resource** (an **Assessment Resource**, more specifically, an **Exam**) that *approaches* the **Subject/Skill**s: **Algorithm**, **Sequence**, **Conditional**, **Loop**, **Function**, **Remixing** and **Debugging**. It is **Composed by** a **Computer** (Individual of **Material**) - because it is a virtual test, requires a computer to access - and 24 questions, which are instances of **Fragment** (**Assessment Fragment**s,

[7]The original Computational Thinking Test (in Spanish) can be found at https://docs.google.com/forms/d/e/1FAIpQLSdPdSj_ZVUhIhG4S3bCH6zXSHZoHHbv6OsmCF9drmbDpfBy_Q/viewform?fbzx=1710660382635002705

[8]https://csunplugged.org/en/at-home/kidbots/

[9]https://projects.raspberrypi.org/en/projects/cats

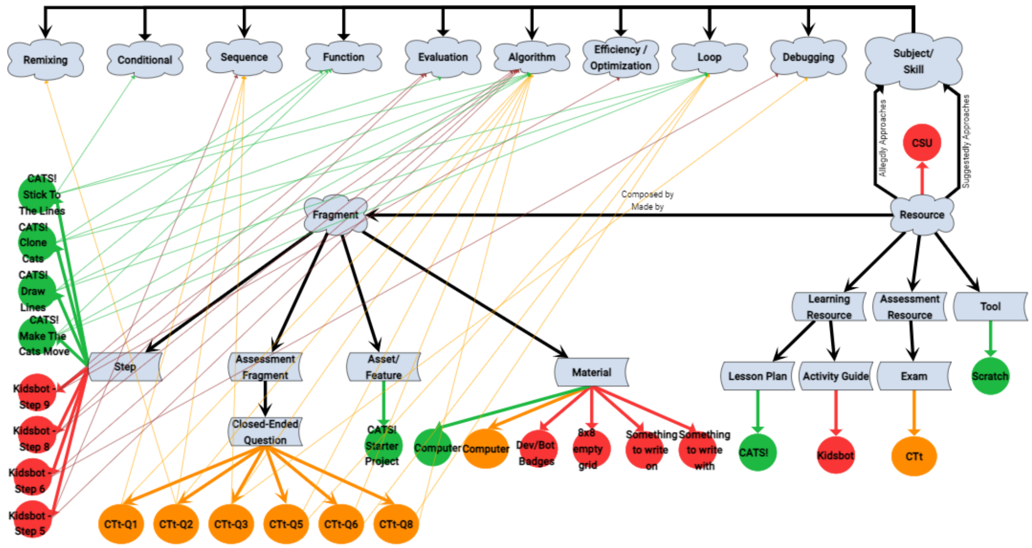

Fig. 8. Instantiated example resources and fragments of CTt, CSU and Scratch in OntoPACT.

more specifically, **Closed-Ended Question**s). Each question **Requires** an **Aptitude** and **Materializes** a **Task**.

This model was created according to information extracted from: "Instrument Type: multiple choice test with 4 answer options (only one correct). (...) Basic directions and sequences (4 items); Loops-repeat times (4 items); Loops-repeat until (4 items); If-simple conditional (4 items); If/else-complex conditional (4 items); While conditional (4 items); Simple functions (4 items). (...) Sequencing: the student must sequence, stating in an orderly manner, a set of commands (14 items); Completion: the student must complete an incomplete given set of commands (9 items); Debugging: the student must debug an incorrect given set of commands (5 items)." [14]

[CTt] is an **Resource (Exam)** *composed by* the following **Fragment (Question)**:

- **[CTt-Q1]**, that *materializes* the **Task [To Make a Path by Sequencing Directions]**, *approaching* **Sequence** and **Algorithm**.
- **[CTt-Q2]**, that *materializes* the **Task [To Complete a Path by Sequencing Directions]**, *approaching* **Sequence**, **Remixing** and **Algorithm**.
- **[CTt-Q3]**, that *materializes* the **Task [To Correct Paths]**, *approaching* **Sequence**, **Debugging** and **Algorithm**.
- **[CTt-Q5]**, that *materializes* the **Task [To Make a Path by Repeating Directions]**, *approaching* **Loop** and **Algorithm**.
- **[CTt-Q6]**, that *materializes* the **Task [To Make a Path by Repeating Directions]**, *approaching* **Loop** and **Algorithm**.
- **[CTt-Q8]**, that *materializes* the **Task [To Make a Path by Repeating Directions]**, *approaching* **Loop** and **Algorithm**.

B. Computer Science Unplugged

[Computer Science Unplugged] (red in Fig. 8) is a **Resource (Teaching Resource)**, *composed by* many activities, which are **Resources** themselves. For instance, **[KidBots]** is a **Teaching Resource (Activity Guide)**, that requires the following **Fragment (Material)**: **[Dev/Bot Badges]**; **[8x8 Empty Grid]**; **[Something to Write On]**; and **[Something to Write With]**. **[KidBots]** is *composed by* the following **Fragment (Step)**s:

- **[Kidsbot-Step 5]**, that *materializes* the **Task [To Make a Path by Sequencing Directions]**, *approaching* **Sequence** and **Algorithm**. From "The developer writes the instructions (program) for how to get from the start position to the end position i.e move the bot to the ball."[8].
- **[Kidsbot-Step 6]**, that *materializes* the **Task [To Correct Paths]**, *approaching* **Debugging** and **Algorithm**. From "(If you don't get to the correct square) Nice try, but you need to debug your program!"[8].
- **[Kidsbot-Step 8]**, that *materializes* the **Task [To Find Alternative Paths]**, *approaching* **Evaluation** and **Algorithm**. From "Is there another program you could have given the bot to get to the ball? Write the instructions/program for that route."[8].
- **[Kidsbot-Step 9]**, that *materializes* the **Task [To Recognize Inefficient Paths]**, *approaching* **Efficiency/Optimization**, **Evaluation** and **Algorithm**. From "What's a long set of instructions to get the bot to the ball?"[8].

C. Scratch

[Scratch] is a **Resource (Tool)**, (green in Fig. 8) composed by several **Features** and related to several other **Resources**. For instance, **[CATS!]** is a **Teaching Resource (Lesson Plan)** that requires a **[Computer]**, which is a **Fragment (Material)**, the **[Scratch]**, which is a **Resource (Tool)**, and **[CATS!**

Starter Project], a **Fragment (Asset)**. **[CATS!]** is *composed by* the following **Fragment (Step)**s:

- **[CATS!-Draw Lines]**, that *materializes* the **Task [To Program an Event to be Triggered by an Input]**, *approaching* **Loop**, **Function** and **Algorithm**. From "Add some code to tell the sprite to draw a line on the Stage if the mouse button is pressed down."[9]
- **CATS!-Clone Cats**, that *materializes* the **Task [To Program an Event to be Periodically Triggered]**, *approaching* **Loop**, **Function** and **Algorithm**. From "Click on the sprite called 'Cat', and add some code to hide the sprite, and also to clone it every three seconds."[9]
- **CATS!-Make the cats move**, that *materializes* the **Task [To Program an Event to be Periodically Triggered]** and **[To Make a Path by Repeating Directions]**, *approaching* **Loop**, **Function** and **Algorithm**. From "Add code to the when I start as a clone section to make the cat sprite move ten steps, and switch between the sprite's two costumes every 0.1 seconds to make the cat look like it's walking."[9]
- **CATS!-Stick to the lines**, that *materializes* the **Task [To Program an Event to be Conditionally Triggered]** and **[To Correct Paths]**, *approaching* **Conditional**, **Evaluation** and **Algorithm**. From "This time, the loop should tell the cat to move upwards by 2 until it is not touching blue."[9]

D. Discussion

Following the scheme presented in Fig. 2, the given examples are instantiated upon Resources and Fragments of the LLO, by which they relate with various Subjects, as seen in Fig. 8. These Subjects, in turn, may be organized according each author, as Fig. 9 shows for the authors presented throughout this paper (Wing, Brennan and Silva Junior) and the Subjects approached by the example Resources (Remixing, Conditional, Sequence, Function, Evaluation, Algorithm, Optimization, Loop and Debugging).

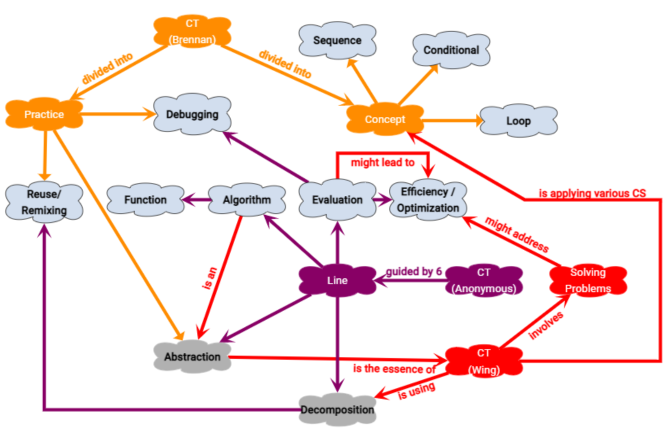

Fig. 9. Views of Wing, Brennan and Silva Junior upon the same CT subjects.

Abstraction is called as an auxiliary Subject for Wing's view to reach Algorithm, as Decomposition, for Silva Junior's view to reach Reuse/Remixing. Although, Abstraction is an intersection between the three of them, while Decomposition is intersection between Wing and Silva Junior. It is interesting to note that those intersections are not necessarily disagreements, Abstraction being the essence of CT (for Wing) does not deny it being a Practice (for Brennan) nor a Line (for Silva Junior).

Using Fragments, OntoPACT precisely maps the points of interest, which enables to track and retrieve desired resources by subject even when it is just a part (fragment) of the whole (resource) that is effectively dealing with this subject. Making use of that, a teacher that wants to asses debugging, for instance, could instantly avoid all other CTt questions not targeting it. Using Tasks/Descriptions, OntoPACT offer the possibility of multiple materializations, approaches of what is essentially the same thing. For instance, the task [To Make a Path by Repeating Directions] may pop up in a multiple-choice question, an outside activity or a programming project. As more data populates the model, these tasks may show exciting alternatives for activities that, for any reason, couldn't be completed. As well as easily linking forms of developing student performances to forms of assessing them.

VI. Conclusion

This paper addressed to the problem of a missing center of CT resources, which is suggested as a future direction for CT research in recent literature [18]. It goes further, recognizing that the multiple views and approaches of CT could be a possible strength of the area, if the platform centering the resources did not impose a view, but rather captured, mapped and explored as many as it could.

The presented model, PACT, is designed for beyond the scientific environment, bridging two communities that often operate far for each other. It paves the way of a mutually beneficial relationship, a collaboration community that could help both sides advance and share their knowledge and resources. It is fit for nourishing Citizen Science in education, potentially feeding the CT research. Enabling Citizen Science projects is one of the main goals of the PACT, guiding the future work once the platform is online.

The implementation of this model, the OntoPACT, showed that ontologies are able to construct vocabularies even under such a diversity of perspectives upon the subjects, like the one found in the CT literature. That is a remarkable contribution for the ontology research, since they are known to rely on well-established vocabularies and consensus between the specialists of the domain of knowledge.

The case study illustrates that the model does not need to be massively populated to prove useful. With three example resources and the perspectives of three authors, notwithstanding their different natures and purposes, it was already possible to visualize points in common and draw conclusions.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001

REFERENCES

[1] V. J. Shute, C. Sun, and J. Asbell-Clarke, "Demystifying computational thinking," *Educational Research Review*, vol. 22, pp. 142–158, 2017.

[2] S. Haines, M. Krach, A. Pustaka, Q. Li, and L. Richman, "The effects of computational thinking professional development on stem teachers' perceptions and pedagogical practices," *ATHENS JOURNAL OF SCIENCES*, vol. 6, pp. 97–122, 2019.

[3] F. F. da Silva, L. B. R. Aylon, and D. E. Flôr, "Teaching computational thinking to a student with attention deficit through programming," in *2020 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2020, pp. 1–9.

[4] B. Fanny, H. Julie, and C. Anne-Sophie, "Developing a critical robot literacy for young people from conceptual metaphors analysis," in *2020 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2020, pp. 1–7.

[5] J. J. Lu and G. H. Fletcher, "Thinking about computational thinking," in *Proceedings of ACM SIGCSE Bulletin*, vol. 41. ACM, 2009, pp. 260–264.

[6] D. Yang, Y. Baek, and S. Swanson, "Developing computational thinking through project-based airplane design activities," in *2020 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2020, pp. 1–4.

[7] M. Román-González, J. Moreno-León, and G. Robles, "Combining assessment tools for a comprehensive evaluation of computational thinking interventions," in *Computational Thinking Education*. Springer, 2019, pp. 79–98.

[8] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[9] P. J. Denning and M. Tedre, *Computational Thinking*. Mit Press, 2019.

[10] International Society for Technology in Education, Computer Science Teachers Association, and National Science Foundation, *Computational Thinking leadership toolkit*, 1st ed. International Society for Technology in Education, 2011. [Online]. Available: https://id.iste.org/docs/ct-documents/ct-leadershipt-toolkit.pdf?sfvrsn=4

[11] B. A. d. Silva Junior, "GGasCT: bringing formal methods to the computational thinking," Master Thesis (Master in Computer Science), Universidade Federal de Pelotas, Pelotas, Brazil, 2020.

[12] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," in *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*, vol. 1, 2012, p. 25.

[13] D. Hooshyar, H. Lim, M. Pedaste, K. Yang, M. Fathi, and Y. Yang, "Autothinking: An adaptive computational thinking game," in *International conference on innovative technologies and learning*. Springer, 2019, pp. 381–391.

[14] M. Román-González, J.-C. Pérez-González, and C. Jiménez-Fernández, "Which cognitive abilities underlie computational thinking? criterion validity of the computational thinking test," *Computers in human behavior*, vol. 72, pp. 678–691, 2017.

[15] J. A. Rodríguez-Martínez, J. A. González-Calero, and J. M. Sáez-López, "Computational thinking and mathematics using scratch: an experiment with sixth-grade students," *Interactive Learning Environments*, vol. 28, no. 3, pp. 316–327, 2020.

[16] A. Fidai, M. M. Capraro, and R. M. Capraro, """scratch"-ing computational thinking with arduino: A meta-analysis," *Thinking Skills and Creativity*, vol. 38, p. 100726, 2020.

[17] A. Threekunprapa and P. Yasri, "Unplugged coding using flowblocks for promoting computational thinking and programming among secondary school students." *International Journal of Instruction*, vol. 13, no. 3, pp. 207–222, 2020.

[18] X. Tang, Y. Yin, Q. Lin, R. Hadad, and X. Zhai, "Assessing computational thinking: A systematic review of empirical studies," *Computers & Education*, vol. 148, p. 103798, 2020.

[19] R. Bonney, C. B. Cooper, J. Dickinson, S. Kelling, T. Phillips, K. V. Rosenberg, and J. Shirk, "Citizen science: A developing tool for expanding science knowledge and scientific literacy," *BioScience*, vol. 59, no. 11, pp. 977–984, 2009. [Online]. Available: https://doi.org/10.1525/bio.2009.59.11.9

[20] N. Guarino, D. Oberle, and S. Staab, "What is an ontology?" in *Handbook on ontologies*. Springer, 2009, pp. 1–17.

[21] J. M. Wing, "Computational thinking and thinking about computing," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 366, no. 1881, pp. 3717–3725, 2008.

[22] T. Bell, J. Alexander, I. Freeman, and M. Grimley, "Computer science unplugged: School students doing real computing without computers," *The New Zealand Journal of Applied Computing and Information Technology*, vol. 13, no. 1, pp. 20–29, 2009.

[23] N. Araujo, T. Primo, and A. Pernas, "Ontoscratch: ontologias para a avaliação do ensino de pensamento computacional através do scratch," in *Simpósio Brasileiro de Informática na Educação (SBIE)*. CBIE, 2020, pp. 1823–1832.

[24] A. Azevedo, C. Araújo, and P. Henriques, "Micas, a web platform to support teachers of computing at school," *Challenges 2019: Desafios da Inteligência Artificial, Artificial Intelligence Challenges*, pp. 625–641, 2019.

[25] S. Grover and R. Pea, "Computational thinking in k-12: A review of the state of the field," *Educational researcher*, vol. 42, no. 1, pp. 38–43, 2013.

[26] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. S. Silver, B. Silverman *et al.*, "Scratch: Programming for all." *Commun. Acm*, vol. 52, no. 11, pp. 60–67, 2009.

[27] G. Guizzardi, "Ontological foundations for structural conceptual models," PhD Thesis (Doctor of Philosophy), University of Twente, Enschede, The Netherlands, 2005.

[28] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[29] D. Fensel, *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Verlag, Berlin: Springer, 2000.

[30] L. Pouchard, L. Cinquini, B. Drach, D. Middleton, D. Bernholdt, K. Chanchio, I. Foster, V. Nefedova, D. Brown, P. Fox *et al.*, "An ontology for scientific information in a grid environment: the earth system grid," in *CCGrid 2003. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003. Proceedings.* IEEE, 2003, pp. 626–632.

[31] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific american*, vol. 284, no. 5, pp. 34–43, 2001.

[32] W. J. Popham, "What's wrong–and what's right–with rubrics." *Educational leadership*, vol. 55, no. 2, pp. 72–75, 1997.

[33] C. S. Unplugged, "Computer science without a computer," Available at: https://csunplugged.org/en/. Accessed: 2021-05-17, 2021.