
Critical Thinking Sheets: Encouraging critical thought and sketched implementation design

Jonathan C. Roberts

Bangor University

Bangor, Gwynedd, UK

j.c.roberts@bangor.ac.uk

Panagiotis D. Ritsos

Bangor University

Bangor, Gwynedd, UK

p.ritsos@bangor.ac.uk

ABSTRACT

Learners are often asked to create an interface as part of their course. For example, they could be asked to “create a calculator”, “develop a stopwatch” or “develop an image processing app”. But students often struggle to know how to start. At the same time, teachers want their students to think critically about their assignments and plan how they will build an interface. We have developed, and used for two academic years, a structured “critical thinking sheet (CTS)”. It is a method to help students consider a problem from different views, and help them critically consider different aspects of the task. The sheet gets the learners to (1) sketch the solution, (2) explain the challenge, (3) detail system components, (4) list algorithmic steps, and (5) explain next steps and issues of implementation. In this paper we introduce the sheet, explain how we have used it, and discuss learner experience.

CCS CONCEPTS

- **Human-centered computing → Interaction design process and methods; Interface design prototyping; User interface design;**

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

EduCHI2019, 2019, Glasgow, UK

© 2019 Copyright held by the owner/author(s).

KEYWORDS

Sketching, low-fidelity design, critical thinking, education, five design-sheets, interaction

ACM Reference Format:

Jonathan C. Roberts and Panagiotis D. Ritsos. 2019. Critical Thinking Sheets: Encouraging critical thought and sketched implementation design. In *EduCHI 2019 Symposium: Global Perspectives on HCI Education*. ACM, New York, NY, USA, 8 pages.

INTRODUCTION

Learners, when faced with a programming assignment, such as to ‘develop a calculator program’, often struggle to think about the issues and the structures necessary to build such a computing interface. Many students jump into coding, without spending time to analyse the challenge and devise the best way to tackle it. Often, because of their haste they get stuck on unsuitable solutions, and at some point inclined to throw away their code and start again. But on many taught courses, the students do not have enough time to code several versions and therefore they resort to submitting sub-optimal work. However, if they are actively encouraged to make plans and think about the assessment before commencing any implementation, they will have a better opportunity to judge which are viable, understand a good potential solution, and will end up having a better grasp of the problem.

One way to remedy this superficial engagement with their challenge would be to get learners to perform an in-depth design-study. Yet, these often take much effort and time (time that students may not have), may require students to learn another methodology, and importantly it is actually a different task to what they are set out to do - a simple yet non-trivial implementation. By getting the students to perform a design-study the teacher is facilitating the development of creative skills of their students, rather than focusing on application and analysis skill development. While creativity is an important aim, it is not always suitable or befitting of the learning objectives of a module, or a course. Especially for novice computing students, creative expression can divert, distract, or even demoralise students, such that they do not achieve the goals of the assignment. We are aiming for a middle ground: methods that are sensible, pragmatic and pedagogic; assessments that are moderated to the level of the students and focused on the learning outcomes of the course. What is needed is a method that is quick to use, simple to administer, and encourages students to think about the problem through different perspectives; but is not a full design-study.

Consequently, the focus of this paper is to engender critical thinking over the posed problem. We direct our method to computing students who are half-way through a degree course, i.e., we assume they can code, understand algorithms, data-structures and database manipulation. At this level, students already know a lot about developing interactive programs, and can draw upon previous knowledge. But the teacher needs to get them to apply their knowledge to more challenging tasks.

PROBLEM:	NAME:	DATE:
Patterns & Abstraction (sketch the problem)		
Sketch pad		Concept/challenge
		What is the challenge?
Parts (e.g., vars)		
What are the parts or components?		
Steps & Rules (e.g., pseudo code)		
What are the algorithmic steps?		Think start, next, stop
Caveats / conditions		
What to think about when implementing?		
Critical Thinking Sheet v1.2		

Figure 1: The critical thinking sheet (CTS) showing the five categories.

Our intention is to encourage students to analyse their problem, break it into sub-components, and devise a strategy for what they will build.

Our solution is to focus on sketching and writing simple lists of components, and to consider the task at hand by thinking about it from different viewpoints [5]. As teachers we often sketch drawings on paper to quickly demonstrate the layout of an interface, underpinning structure of the code, or the algorithm itself. Likewise students become more confident coders when they start thinking about the problem using pencil and paper. Like other educators, we have found that sketching and low-fidelity prototyping techniques help most learners to carefully consider their tasks and judge the efficacy of different solutions [19]. Sketches have the potential to help students envision what their tool will look like, contemplate and assess different solutions, make plans of how to develop their tools, and reflect on their outcomes [15, 16].

We have started to formalise this approach by developing a sheet that encourages the user to consider consider the problem through different viewpoints. We name this the Critical Thinking Sheet (CTS), which is divided into five parts. Using a formal process, rather than using ad hoc sketching, allows students to submit their sheets, and reflect on their process in a report. We (1) introduce the CTS in this paper placing it into context with related work, (2) present how we have used the sheet in our teaching, and discuss students' feedback and reflection on the process.

BACKGROUND & RELATED WORK

Question phraseology changes the task that the students need to perform. In fact, by using different phrasing the student may need to perform the task differently. Open-ended questions require students to contemplate alternative design solutions, whereas highly prescriptive questions remove design creativity from the students. Let us first focus on open-ended problems, also known as ill-defined questions [21] in comparison to well-defined problems.

There are many different strategies to solve open-ended questions, such as to “build an interface to visualise data of your choice” or “develop a tool to create different patterns”, but each strategy requires that the student considers alternative solutions. As the person tries to tackle them, they are learning about the domain, contemplating limitations of different existing solutions and considering their potential significance, relevance and efficacy. Many models could be used to answer such questions, including models by the Design Council [4], Munzner’s nested model for visualisation design [13], the understand, ideate, make, deploy process of McKenna et al. [11] and the nine-stage design-study model by Sedlmair et al. [20]. Approaches such as Making [10], token and construction [8], or brainstorming tools such as VisitCards [7] can be used to explore the design space. Likewise, sketching can help students consider different solutions. Methods such as ad hoc sketching [3] or structured sketching, such as with the Five Design Sheet (FdS) method [15, 16] could be used. Still, even when the student has created their final sketched design (e.g., sheet 5 of the FdS) they need to interpret its concepts

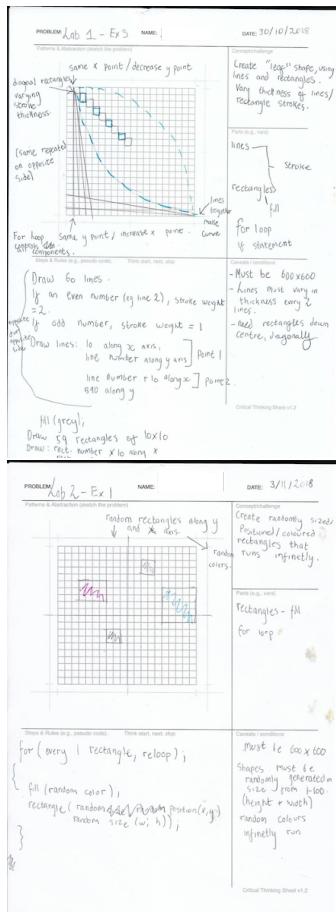


Figure 2: Example of two sheets from one student's classwork submission.

and write the necessary code to implement it. Some students know exactly how to develop their application, while others struggle to transform their design into a code implementation. For this goal we have used the Explanatory Visualisation Framework (EVF) in our teaching, as a way to structure the students' design process [19]. The design-study approaches work well with longer assessments and with more experienced programmers. But what is needed is a practical approach to lead the (less experienced) learners from the conceptual stage into the implementation stage. To get the students to contemplate different aspects of the interface design.

For our purpose in teaching mid-level learners of a second year computing degree course, less open-ended tasks and questions that are *well-defined* [17, 21] are more suitable. Such questions do not need the student to (necessarily) perform an in-depth design-study, but still require them to think carefully about the necessary sub-components, and plan their synthesis before they code their implementation. Students need to make their own decisions about how to structure the code, how they will implement the required features, or what the interface will appear like etc. There are many computing strategies and techniques that a student may use to help them think how to implement their solution. For instance, they could use Unified Modelling Language (UML) [1] to describe the structure, or use wireframe mockups to outline the interface. But unfortunately, there is still a conceptual leap from the design sketch (or the task description set by the teacher) into individual components that then can be coded.

As aforementioned, we want the students to conceptually break down a problem into individual parts, and think about how to implement the underpinning algorithms, functionality and interface. In other words, we wish to separate concerns. We achieve this goal by getting the students to focus their attention on one aspect at a time [5]. There are many models that the teacher could use to achieve this end, and we acknowledge a huge body of work, not only from the computing domain, that is relevant to our goal. This includes considerations of *design competence* [14], lessons from global companies and lessons from architectural design (e.g., Suwa and Tversky [22] and Frederick [6]), the waterfall model (of requirements, design, implementation, verification, maintenance), and instructional models (e.g., ADDIE [2], Jonassen [9]). Each of these strategies have merits, however when integrated into a course assessment it can be difficult for a student to know how to analyse the problem breaking it into parts, and then synthesise those parts together. We need something that was quick to perform, a focused and quantifiable activity, that then students can submit, in order to demonstrate that they had performed the necessary steps.

THE CRITICAL THINKING SHEET

We demonstrate the CTS through its use with second year BSc Computing students studying Computer Graphics and Algorithms. The aims of this course are to develop skills in critical thinking and analysis, and to get students to apply their programming knowledge to larger challenges around the

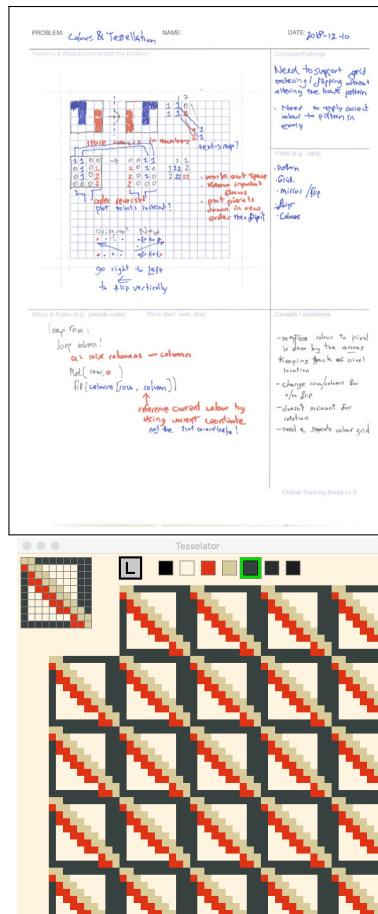


Figure 3: Student's submission to the pattern tesselator interface, showing their Critical Thinking Sheet (CTS) and final submission (different reflections or rotations can also be chosen by clicking on the L icon).

development of computer interfaces. The students have three requisite modules: two programming modules of 5 ECTS credits each, and a 5 ECTS credit module on Human Computer Interaction.

Our original approach was to give students a blank sheet of paper and ask them to write or sketch a plan. We started with this approach, because we (as teachers) often found ourselves reaching for sheets of blank paper to sketch ideas of how to structure the code. However, this strategy requires that the students imagine what a potential result would look like and be confident to break the problem into implementation steps. We found that while some students were comfortable to sketch and label their ideas (of how they would structure and implement the tool), the majority of them just kept asking what they should do on the sheets. Reflecting on ways to offer more prescriptive guidance to those students, we realised that we were using the blank sheets in multiple ways and for different purposes. Sometimes we would sketch an interface, other times we drew a schematic diagram, or described an algorithm with pseudo-code. We also often wrote some notes, along with these depictions, about how they could be implemented in practice.

We therefore decided to divide the sheet into separate sections (see Figure 1), each focusing on different concerns.

Sketch pad. Sketch the concepts. Depending on the task this could be the user interface, or a conceptual diagram to explain some part of the algorithmic solution.

What is the challenge? Explain or sketch the challenge. Try to articulate the goals that the tool will achieve, and any assumptions that need to be made.

What are the parts/components? How would you implement it? What parts can you divide the work into? Can you explain the work in terms of a class, method or a design pattern?

What are the algorithmic steps? What is the core algorithm? What are the main steps that the algorithm needs to take?

What to thinking about when implementing? Do you need to do more research? How can you take your ideas and implement it? What assumptions have you made, or how have you simplified your solution?

Using the Critical Thinking Sheet (CTS)

In our Computer Graphics module we give five lectures, covering basic graphics drawing using Processing, and how to develop interactive systems. We use live-coding sessions in the lectures to support the material. We give the students five classroom sessions, where they follow increasing difficulty tasks. The students can use up to two hours in these classes, and ask for help from lab assistants to complete the exercises. Three tasks are given in each class, and students are asked to complete a CTS for each task before implementing their solution. Figure 2 shows two CTSs of the same student, demonstrating how they have completed the sheets. The students then write and

submit a report on their activities in these classes, along with copies of their CTSs. In their report they reflect on their tasks, and attach a screen shot of their solution and code snippets demonstrating their completion of each task. A simple marking strategy is used to grade these classes, biased on them completing the tasks and demonstrating successful use of the CTS. The idea is that students become familiar with the sheets during these sessions, such that they develop good practices; which they can then apply to the large self-study assignment. The class tasks range from drawing simple coloured grids, to triangular patterns, and simple interactive interfaces. The students then work on a larger assessment. We have used different assessments including creating a volume data viewer (a slice/dice program that views MRI data in 2D) and a pixel pattern tesselator. The CTS and a screenshot of an interactive program from one student for the latter task is shown in Figure 3.

DISCUSSION

Reflecting on the use of the CTS, which we have successfully integrated the CTS into the Computer Graphics module, we draw on a broad range of experiences with applying different methodologies that integrate sketching and design thinking into the computing curriculum; notably the Five Design Sheets method [15, 16] and the Explanatory Visualisation Framework [19]. For this module, we ask the students to write a reflective report at the end of their assessment, within which they have the opportunity to express their opinions about the CTS. This feedback is provided on their own volition, without guidance or influence from ourselves. To reflect on the process we consider the reactions of several students; changing their names for these examples.

Frank wrote comprehensively about his experience with the pattern-tesselator assessment. “*The first critical sheet I created was when I just started working on the project. I had an idea of how to create the tool, but actually making a quick design and noting down some initial ideas helped me. By completing this sheet, I was able to see how the pattern could be repeated. ... If I started by going straight to writing the code, it's likely that I would have forgotten about the push/pop matrix, meaning I would be wasting time trying to fix errors.*” This is exactly the reaction we were hoping for. Frank had thought carefully through the problem and was drawing on prior knowledge. Frank also used a second sheet, explaining how he moved his design on from his initial thoughts into something that he then implemented. He refined his thoughts through using the sheets. Gillian made a similar response, saying “*The main benefit of using a critical thinking sheet was in solving this problem by allowing me to visually demonstrate where the tiles and squares needed to be rather than making guesses about the offset formula which should be used*”. Indeed, one of our goals is to help students visualise the target outcome. Although Lisa did not complete a complex interactive system, she used the CTS to explain what additional functionality she could have done. “*More advanced patterns, ... could have been added to make design more creative, as explained in the Critical Thinking Sheet.*” Again this is a positive outcome, because Lisa has realised that there are multiple ways to contemplate the solution.

On the other hand, a few students did use a more negative language. Jack wrote “*I didn’t use any critical thinking sheets for this assignment. While I am sure they are useful for some, I found them more hindrance than help in the previous labs, and so in the interest of time ignored them. People learn and work in many ways; for me personally, the critical thinking sheets were not helpful. I would work through the critical thinking sheet but by the end of the assignment I had changed so much, often due to an initial oversight or new idea, that it became barely relevant.*” Jack’s reservations are indeed valid. The CTS is a tool to help the student progress to the next stage. Sometimes the information on the sheet becomes superseded by new information (they need to be thrown away, and a new idea created), exactly as Frank did, above. Another student, Francis said “*I struggle to gain anything much of value before sitting down and trying to bash some code out ... and if I get stuck ... I do scrawl some working out in my notebook*”. He submitted his pencil scribbles on paper, which were UI designs and some plans demonstrating thought over the underpinning algorithm. While Francis did not use the CTS structure, he demonstrated elements of critical thinking of the assignment, which is an important motivation for using the CTS in the first place.

As authors we have a goal to integrate more design thinking into our computing curriculum, to encourage students to think before acting and coding, and to encourage other teachers to add more design-thinking into their teaching methods. Yet, integrating design and sketching into the curriculum, and encouraging students to sketch is not necessarily easy. In past years we have had many outspoken students throwing their arms in anguish over being asked to use a pen and pencil [16, 18]. But, anecdotal evidence suggests to us that these cries of frustration are becoming less frequent. This maybe because students are now being exposed to different teaching methodologies, especially those that work away from a computer and are more inclusive of different techniques. And while today’s generation Z students are digital natives, they revisit and enjoy retro concepts (as suggested by the rise of vinyl record sales), and equally comfortable using pen and paper. Still, many students want to be ‘taught to the test’ and be hand-held through the exercises [12]. For instance, in one class Lisa asked “what exactly do I have to do to pass”. It is therefore not surprising that today’s students have become less skilled at critical thinking. We conjecture that the CTS works well because it helps to focus their thoughts on individual tasks, and in fact creates a structure that organises the critical thought process into a directive, goal-oriented set of sub-tasks. In addition, the accompanying reflective report works as a companion to the CTS. By writing their report and reflecting on their critical thinking process they are deepening their critical thinking skills. The tasks in the classes also help the students to apply their knowledge. The process of quickly thinking about a problem, articulating it on paper (on the CTS) and immediately coding it, helps them to directly apply their knowledge. Consequently, there is a rational behind using these sheets that the students appreciate.

REFERENCES

- [1] G. Booch, J. Rumbaugh, and I. Jacobson. 1998. *The Unified Modeling Language User Guide*. Addison Wesley, MA, USA.
- [2] Robert K Branson, Gail T Rayner, J Lamarr Cox, John P Furman, and FJ King. 1975. *Interservice procedures for instructional systems development executive summary and model*. Technical Report. DTIC Document.
- [3] Bill Buxton. 2010. *Sketching user experiences: getting the design right and the right design*. Morgan Kaufmann.
- [4] Design Council. 2007. Eleven lessons: managing design in eleven global companies Desk research report. www.designcouncil.org.uk.
- [5] Edsger Dijkstra. 1982. On the role of scientific thought. In *Selected Writings on Computing: A Personal Perspective*. 60–66.
- [6] Matthew Frederick. 2007. *101 Things I Learned in Architecture School*. MIT Press.
- [7] S. He and E. Adar. 2017. VizItCards: A Card-Based Toolkit for Infovis Design Education. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (Jan 2017), 561–570. <https://doi.org/10.1109/TVCG.2016.2599338>
- [8] Samuel Huron, Sheelagh Carpendale, Alice Thudt, Anthony Tang, and Michael Mauerer. 2014. Constructive Visualization. In *DIS 2014: Proc Designing Interactive Systems in 2014*. ACM, 433–442. <https://doi.org/10.1145/2598784.2598806>
- [9] David H Jonassen. 1997. Instructional design models for well-structured and ill-structured problem-solving learning outcomes. *Educational Technology Research and Development* 45, 1 (1997), 65–94.
- [10] Jonas Löwgren. 2016. On the Significance of Making in Interaction Design Research. *Interactions* 23, 3 (April 2016), 26–33. <https://doi.org/10.1145/2904376>
- [11] S. McKenna, D. Mazur, J. Agutter, and M. Meyer. 2014. Design Activity Framework for Visualization Design. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2191–2200. <https://doi.org/10.1109/TVCG.2014.2346331>
- [12] Michele Monaco and Malissa Martin. 2007. The millennial student: A new generation of learners. *Athletic Training Education Journal* 2, 2 (2007), 42–46.
- [13] Tamara Munzner. 2009. A Nested Process Model for Visualization Design and Validation. *IEEE Transactions on Visualization and Computer Graphics* 15 (Nov 2009), 921–928. Issue 6. <https://doi.org/10.1109/TVCG.2009.111>
- [14] Harold G. Nelson and Erik Stolterman. 2012. *The Design Way: Intentional Change in an Unpredictable World*. MIT Press.
- [15] J. C. Roberts, C. Headleand, and P. D. Ritsos. 2016. Sketching Designs Using the Five Design-Sheet Methodology. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (Jan 2016), 419–428. <https://doi.org/10.1109/TVCG.2015.2467271>
- [16] Jonathan C. Roberts, Christopher J. Headleand, and Panagiotis D. Ritsos. 2017. *Five Design-Sheets – Creative design and sketching in Computing and Visualization*. Springer. <https://doi.org/10.1007/978-3-319-55627-7>
- [17] Jonathan C. Roberts, Daniel Keim, Timothy Hanratty, Robert Rowlingson, Rick Walker, Mark Hall, Zack Jackobson, Valerie Lavigne, Chris Rooney, and Margaret Varga. 2014. From Ill-defined Problems to Informed Decisions. In *EuroVis Wkshp Visual Analytics*, M. Pohl and J. Roberts (Eds.). Eurographics, 7–11. <https://doi.org/10.2312/eurova.20141138>
- [18] Jonathan C. Roberts, Panagiotis D. Ritsos, and Chris Headleand. 2017. Experience and Guidance for the use of Sketching and low-fidelity Visualisation-design in teaching. In *Pedagogy of Data Visualization Workshop, IEEE Visualization (VIS), October 1-6, Phoenix, Arizona, USA, 2017*, A. Joshi, E. Adar, E. Bertini, S. Engle, M. Hearst, and D. Keefe (Eds.).
- [19] J. C. Roberts, P. D. Ritsos, J. R. Jackson, and C. Headleand. 2018. The Explanatory Visualization Framework: An Active Learning Framework for Teaching Creative Computing Using Explanatory Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 791–801. <https://doi.org/10.1109/TVCG.2017.2745878>
- [20] M. Sedlmair, M.D. Meyer, and T. Munzner. 2012. Design Study Methodology: Reflections from the Trenches and the Stacks. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2431–2440.
- [21] Herbert A Simon. 1973. The structure of ill structured problems. *Artificial intelligence* 4, 3-4 (1973), 181–201.
- [22] Masaki Suwa and Barbara Tversky. 1997. What do architects and students perceive in their design sketches? A protocol analysis. *Design studies* 18, 4 (1997), 385–403.