Toward Infusing Modular and Reflective Design Learning throughout the Curriculum

John C. Georgas
Department of Electrical Engineering and Computer Science
Northern Arizona University, Flagstaff, AZ, USA
john.georgas@nau.edu

Abstract

Design is a core activity in computer science and software engineering, but conventional curricula do not adequately support design learning since they isolate the explicit study of design to only a few select courses. This paper reports on our initial efforts toward addressing this challenge through an intervention focusing on the modular deployment of design challenges and design stories throughout the curriculum: Design challenges allow learners to work through open-ended problems, which is followed by the creation of a design story narrative through the use of a structured reflection framework. Our intervention strategy is intended to be easily adoptable within existing curricular structures and leverages reflective-learning theories.

1. Introduction

Design is a pervasive element of software engineering and computer science: for software systems across diverse domains, practitioners make design decisions that determine code-level control flows, data structures that encapsulate interaction paradigms, algorithms with a variety of performance characteristics, object-oriented hierarchies, complex software architectures, test plans and unit tests, user interfaces, and most other facets of software. Given this centrality of the design process and its impact on the eventual quality of software systems, supporting design learning is critical in preparing students to be professional software developers.

Despite the centrality of design, conventionally structured undergraduate computer science curricula explicitly address the development of design skills in only a few isolated curricular points—most commonly in an introductory course on software engineering, which often only considers architectural and interface design. Furthermore, the instructional methods adopted often lack the inclusion of a strong reflective component, which is critically important in the context of design learning. As a result, it is difficult to impart learners with a set of cohesive design skills that appropriately address both the wide spectrum of design activities in software development as well as the fundamental similarities of the design process across this spectrum.

Aiming to address these challenges, we are developing an educational intervention that centers on the widespread infusion of design learning throughout the curriculum using methods that are well-suited to the design activity. The core of the approach are *design challenges* and *design stories*: course-specific design challenges provide learners with problems that give them the opportunity to navigate an open-ended solution space, and the use of a structured reflection framework helps guide students in the creation of design stories that capture insights about the challenge.

While it is not uncommon to use open-ended problems during instruction, our approach is distinguished by adopting a cohesive perspective on design throughout the curriculum while being particularly focused on reflection and the student-driven creation of design story narratives. In sum, this approach exhibits the following key features:

- *Modular:* An emphasis on broadly infusing design learning through the curriculum using modular design challenges and stories, which supports ease of adoption and integration into existing curricular structures and alleviates the isolation of design learning without the burden of specialized courses—particularly for externally accredited programs;
- *Reflective:* Strong adoption of insights from reflection-based learning through the use of structured reflection and the creation of reflective narratives, therefore leveraging learning modes that are particularly well-suited to design; and,
- *Cohesive:* Use of a cohesive approach to general design learning throughout the curriculum in a way that is independent of specific design notations and emphasizes the essential nature of design as a decision-making process critical to the entire software development lifecycle.

This report outlines our initial efforts toward implementing this approach in our curriculum: Thus far, we have focused on planning the intervention strategy and deployment modes for leveraging design challenges and design stories into select curricular points, and on developing course-specific structured reflection frameworks that support the creation of design stories.

2. Background and Related Work

Our pedagogical approach is fundamentally grounded in constructivism [9]: a theory of cognition focused on learner interactions with the environment that has given rise to various conceptualizations of learning. Particularly important in the context of our work is experiential learning [3], which focuses on application-oriented problems during learning. The core activity supported by the development of design stories is learner *reflection*, emphasized by Schön [7] as an essential process for generating greater learning from an experience. Structured reflection [6] focuses on providing guidance to learners for supporting reflection: a key insight of structured reflection is to provide a set of open-ended questions that help learners explore their perceptions and various facets of a learning activity, an insight leveraged in our development of such a framework for reflecting over design decisions over various computer science topics.

Our approach is also informed by design theory: Simon [8] presents a conceptualization of design as a hierarchical decision-making process informed by the constraints of the environment. Other perspectives focus on dynamic aspects of design, such as the evolution of solutions over time and the preservation of design knowledge through patterns [1]. Finally, Jones [2] offers a view of design as a three-stage process of a divergence of possibilities bounded by constraints, transformation of design specifications, and convergence toward solutions. While these different conceptualizations of design may result in divergent approaches to the activity itself, they share a common perspective with our work: Design is situated within a space composed of alternatives, with each combination of design elements imparting different characteristics to the final system.

Finally, the creation of design stories—narratives created by reflecting over a design solution—is related to design rationale [5], which focuses on the explicit capture of the reasons behind decisions and rejected alternatives. While the study of design rationale has been identified as important in improving the design of software systems, capturing rationale entails significant practical challenges, such as the additional effort demanded of designers and the obtrusiveness of capture techniques and tools [4]. The impact of these difficulties is minimized in our adoption of design stories—

fundamentally sharing the goal of capturing a measure of rationale—by the small-scale context established by design challenges and the explicit guidance provided through structured reflection.

3. Approach

Our approach centers on the use of course-specific knowledge as the foundation for *design challenge* (DC) modules that capture open-ended problems. Learners then leverage a structured reflection framework to create a *design story* (DS) narrative, which captures their experiences in navigating the design space of the challenge and making design decision trade-offs.

3.1. Design Challenges

The learning objectives of a specific course provide the foundation for creating DCs, which leverage discovery-based learning by allowing students to explore a design space independently, without an instructor providing an answer presumed to be correct. For example, in a CSII-level course where learners have mastery of basic programming constructs and are learning about algorithms and data structures, a simple DC may involve a problem about calculating individual and course grade averages. For such a challenge, a student may develop the following solution:

> *Sam decides that the most straightforward approach is to build a Student class to store student names, scores, and individual averages, and a main AverageCalculator class. Fields are made private, so appropriate getter and setter methods are also needed. Sam also decides that all Student objects will be stored in an array and sets the size of this array to 60 elements; objects will populate the array as they are created by reading the data file. Finally, Sam thinks that the most straightforward way of completing the requirements of this problem is to use a loop to first calculate each students average score and then another loop to calculate the course-wide average.*

While not overly complex, this challenge still entails a variety of design choices about class decomposition and structure, data structures, and algorithms. The key insight behind the development of DCs is this use of course-specific content to provide the foundation for explicitly discussing design decisions at an appropriate level of abstraction, which helps build general design expertise alongside course-specific technical skills beginning in the earliest parts of the curriculum.

3.2. Design Stories

Once learners solve a challenge, they engage in the creation of a DS narrative that explores the design space of the challenge, considering aspects of design such as functional and non-functional requirements, alternatives to the solution, principles and notations that were used, and how knowledge gained can be preserved for the future. For the example DC shown in the preceding section, a student-created design narrative may be:

> *Sam identifies the main functions as reading data from a file, creating objects to store student data, and calculating averages. He thinks the most critical function is getting the average score right, as that affects student grades. Sam thinks alternative object-oriented designs, for example including a higher-level Person class, would add future expandability but impose more complexity. One important alternative would be using a more dynamic data structure, such as a list: this would generalize the solution to more*

Table 1. A general-purpose structured reflection framework for software design.

DESIGN ASPECT	STRUCTURED REFLECTION QUESTIONS
Functional Requirements	- What were the main functions that the challenge explicitly entailed? - Of these main functions, which one was the most critical and why? - Did you discover any unstated but needed functions while designing?
Non-Functional Requirements	- What non-functional properties did the challenge explicitly entail? - Of these properties, which one was the most critical and why? - What were the trade-offs between non-functional properties?
Decision Points	- What was the design decision made for each important function? - What alternatives were considered for each of these decisions? - How did the decisions made limit the choices of future decisions?
Design Principles	- What design principles did you use in your solution to this challenge? - How did each of these principles influence your design? - Of these principles, which one did you find most helpful and why?
Design Notations	- What kinds of notations did you use when capturing your design? - What design needs guided your choice of notation? - Which of these notations was the most and which the least helpful?
Knowledge Preservation	- Did your solution embody any known design patterns or styles? - What useful patterns can you extract from your solution? - What else did you learn during the challenge that would help others?

students, but impose additional overhead. He thinks the toughest decision is using a two-pass solution for calculating averages: one pass for each students average, and one for the course-wide average. Sam realizes that this is an important trade-off between simplicity and performance. Sam didnt formally use any notations, but he reports that he sketched out a simple UML-style diagram in exploring his design and used some pseudocode to better understand the algorithm. Sam doesnt think he used any existing patterns, but feels that his understanding of the one- versus two-pass solution could be important for future re-factorings.

A key insight of our approach is that this exploration is guided through a structured reflection framework that provides learners with guidance for the creation of a DS narrative and ensures that they consider critical questions. While we are developing a number of course-specific reflection frameworks, a general example (heavily redacted for brevity) appears in Table 1.

3.3. Curricular Integration Concerns

Our approach provides for a wide range of deployment methods and commitments of time on the part of adopting educators: The general process is to provide a short introduction to a design challenge, allow students the time to create a solution to this challenge, prompt the creation of design story narratives using a reflection framework, and conclude by fostering discussion over these design stories while helping guide learners toward any overlooked alternatives. In our experience, the time required for this process ranges between 10 and 15 minutes.

We also consider a range of possible deployment modes for this process: (a) *individual* deployments allow students to work through the process themselves but isolate learners from their peers, (b) *group* deployments are interactive and engaging but suffer from common issues relating to group dynamics, and (c) *mixed* deployments—combining individual with group activities—provide a balanced approach at the expense of a larger time commitment.

Our initial focus on institutionalizing this approach has centered on our software engineering and software architecture courses, with initial deployments in early parts of the curriculum where design is not conventionally addressed. In the context of CSI, we are focusing on the design decisions associated with basic programming artifacts—particularly those that accomplish similar tasks such as `for` and `while` loops. For CSII, we are targeting more advanced programming skills, basic algorithms, and simple data structures such as arrays and lists. For our course on data structures, we are focusing on more complex programming techniques while integrating advanced data structures such as trees, heaps, and graphs. Our software engineering and architecture challenges focus on advanced object-oriented design, design patterns, architectural styles, and a consideration of various application domains, such as service-oriented and distributed systems.

4. Conclusion and Ongoing Work

In conventional undergraduate degree programs, design learning is isolated in only a few discrete curricular points, which contrasts with the pervasive nature of design evident in all levels of abstraction involved in software construction during all phases of the development lifecycle. Our ongoing work in the development of an educational approach focusing on the study of design throughout the curriculum addresses this fundamental shortcoming of current practice, while also leveraging techniques focused on reflective learning that are particularly well-suited to design.

One of the most interesting research directions relates to the analysis of collected student-developed design stories. By collecting, categorizing, evaluating, and analyzing these narratives, we see promising avenues for an evidence-based inquiry regarding the effect of design challenges on the quality and richness of their associated design stories. For example, we are interested in identifying the characteristics of design challenges that are more effective in fostering rich design stories, which may help identify the most effective types of problems for fostering design learning.

Acknowledgments

This research is supported in part by the National Science Foundation under Grant numbers CCF-1016134 and CCF-1017408.

References

[1] C. Alexander. *The Timeless Way of Building*. Oxford University Press, New York, NY, USA, 1979.

[2] J. Jones. *Design Methods*. John Wiley & Sons, Inc., New York, NY, USA, 1992.

[3] D. Kolb, R. Boyatzis, and C. Mainemelis. Experiential learning theory: Previous research and new directions. In R. Sternberg and L. Zhang, editors, *Perspectives on Thinking, Learning, and Cognitive Styles*, pages 227–247. Lawrence Erlbaum, Hillsdale, NJ, USA, 2001.

[4] J. Lee. Design rationale systems: Understanding the issues. *IEEE Expert: Intelligent Systems and Their Applications*, 12(3):78–85, 1997.

[5] T. Moran and J. Carroll, editors. *Design Rationale: Concepts, Techniques, and Use*. Psychology Press, 1996.

[6] I. Reymen. *Improving Design Processes through Structured Reflection: A Domain-independent Approach*. PhD thesis, Technische Universiteit Eindhoven, 2001.

[7] D. Schön. *Educating the Reflective Practitioner*. Josey-Bass, San Francisco, CA, USA, 1987.

[8] H. Simon. The structure of ill structured problems. *Artificial Intelligence*, 4(3):181–201, 1973.

[9] L. Vygotsky. *Mind In Society: The Development of Higher Psychological Processes*. Harvard University Press, Cambridge, MA, USA, 1978.