Snow & Bienkowski: Assessment Practices in Secondary Computer Science Education
Handout for CSTA Annual Conference, 2014

Why Evidence-Centered Design?

We tend to think of "assessments" as quizzes and exams that tell us something about what students know. But more generally, assessments give us evidence of learning whenever we give students situations, challenges, or tasks that will elicit the needed evidence. So what if we could design a *template*, *blueprint*, or *pattern* for the important ideas in computer science that would help us generate many different forms of assessments? What if one pattern could be used to generate paper and pencil or online tests or rubrics to score computational artifacts that students produce? What if it could help us reveal student learning in observations of strategies, tactics, and moves in game play, or student moves in simulation-based and tutored learning environments? Evidence-centered design—a principled method for systematically analyzing a domain of interest and building multiple assessment tasks—has a tool for just this purpose—a tool called a "design pattern.

In the slide deck accompanying this workshop, ECD is described as 5 layers of work with important questions and activities occurring at each layer. Design patterns are generated during one layer of evidence-centered design (ECD). Although these layers suggest steps in a sequential design process, cycles of iteration and refinement are intended, both within and across layers, and work in different layers can occur simultaneously. Table 1 below shows a slightly different view of these layers with examples from the computational thinking domain.

- Work at Layers 1-3 of ECD help us to unpack the idea of computational thinking and to create detailed design documents for assessments in different domains: computer science, science, math, programming, etc.
- Domain analysis (Layer 1) can include broad reviews of research literature from computer science, computer science education, and learning sciences; computer science curricula; sample assessment tasks; and relevant standards frameworks, as well as expert input from practitioners, researchers and educators.
- Domain modeling (Layer 2) and framework development (Layer 3) result in detailed design documents and assessment specifications.
- In Layers 4 and 5 of ECD, the job is to put assessment items into a form, implement (pilot and test) and deliver assessments, building off of the design documents.

* Produced by the Center for Technology in Learning, SRI International, with support from the National Science Foundation under contract numbers, CNS-1132232 and CNS-1240625. Any opinions, findings, conclusions, or recommendations expressed in herein are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Table 1. Roles and Key Entities in the Five Layers of Evidence-Centered Design

ECD Layer	Role	Key Entities & Examples
1. Domain Analysis	Gather substantive information about the computational thinking domain of interest that has implications for assessment; how knowledge is constructed, acquired, used, and communicated.	Computational thinking domain concepts (e.g., abstraction, automation); terminology (debugging); tools (programming languages); representations (storyboards); situations of use (modeling predator-prey).
2. Domain Modeling	Express assessment argument in narrative form based on information from Domain Analysis	Specification of knowledge, skills, and other attributes to be assessed (e.g., describe result of running a program on given data); features of situations that can evoke evidence (find errors in programs); kinds of performances that convey evidence (use of recursion).
3. Conceptual Assessment Framework	Express assessment argument in structures and specifications for tasks and tests, evaluation procedures, measurement models.	Student, evidence, and task models; student, observable, and task variables; rubrics; measurement models; test assembly specifications; task templates and task specifications.
4. Assessment Implementation	Implement assessment, including presentation-ready tasks and calibrated measurement models	Task materials (including all materials, tools, affordances); pilot test data to hone evaluation procedures and fit measurement models.
5. Assessment Delivery	Coordinate interactions of students and tasks: task-and test-level scoring; reporting.	Tasks as presented; work products as created; scores as evaluated.

What is in a Design Pattern?

ECD begins with domain analysis, and for our purposes here, assume that we have information from that layer on what is involved in the content area we are measuring. This could be content standards or learning goals for the curriculum. Design patterns arise from domain modeling in Layer 2 and begin with the statement of the core or key idea that the design pattern covers. Here are two examples: one is a general computational thinking practice, the other specific to an ECS Unit 3.

Computational Thinking Practice: Design and Implement Creative Solutions and Artifacts

This is called the "construct" in the noun sense of "something formed (image, idea, or theory) in people's minds."

Design Pattern Attribute	Attribute Content
Overview	This design pattern supports the development of tasks in which students translate novel ideas into tangible forms. It encompasses steps of the problem solving cycle, including understanding, decomposing, and exploring the problem, create work products that show one or more designed solutions and/or artifacts, and then test and improve the solution and or artifact.

Definition of the construct

Exploring Computer Science – Unit 3: Web-Design Design Patterns

ECS Unit Description
Unit 3 prepares students to take the role of a developer by expanding their knowledge of algorithms, abstraction, and web page design and applying it to the creation of web pages and documentation for users and equipment. Students will explore issues of social responsibility in web use. They will learn to plan and code their web pages using a variety of techniques and check their sites for usability. Students learn to create user-friendly websites. Students will apply fundamental notions of Human Computer Interaction (HCI) and ergonomics.

After we have a good definition or description of the overall construct or unit, the next step is to list all of the focal knowledge, skills and attributes (FKSAs). "Focal" here means central or core. The FKSAs are the knowledge, skills, and attributes related to the student that you want to assess. They should cover the main ideas within the construct of interest." There are 16 "abilities" we have defined for the construct "Design and Implement Creative Solutions and Artifacts" and we show 5 of them below, as well as 3 of the 8 we have defined for "ECS Unit 3: Web Design."

Example Focal Knowledge & Skills for "Design and Implement Creative Solutions and Artifacts"

1. Ability to state a problem in order to identify the inputs and outputs of the problem
2. Ability to decompose a problem into multiple sub-problems, including the specification of how solving the sub-problems will lead to a solution to the problem as a whole.
3. Ability to create a computational artifact given a purpose or intent
4. Ability to select appropriate techniques to develop computational artifacts
5. Ability to identify run-time errors

Example Focal Knowledge & Skills for "Unit 3: Web-Design"

1. Ability to create a set of specifications for a web page given the intent of the web page
2. Ability to express the design of a web page based on specified objectives
3. Ability to describe techniques used when designing and implementing a web page

After we have listed all of the FKSAs that make up each construct, the next step is to say what we could observe the student doing or producing that would provide evidence of that FKSA. These are shown below for one FKSA from each.

Example Potential Observations for One "Design and Implement Creative Solutions and Artifacts" FKSA

1. FKSA: Ability to state a problem in order to identify the inputs and outputs of the problem
    a. Accuracy of the statement of a problem
    b. Appropriateness of the inputs and outputs identified

Example Potential Work Products for One "Design and Implement Creative Solutions and Artifacts" FKSA

1. FKSA: Ability to state a problem in order to identify the inputs and outputs of the problem
    a. The statement of a problem
    b. The identification of inputs and outputs of a problem

Example Potential Observations for One "Unit 3: Web-Design" FKSA

1. FKSA: Ability to create a set of specifications for a web page given the intent of the web page
    a. Completeness of the description of the specifications (i.e., Did the student include a discussion of the relevant parts of a web page such as headings and menus?)
    b. Appropriateness of the set of specifications (i.e., How well do the specifications match the intent? Is the student providing space for all of the relevant content information?)

Example Potential Work Products for One "Unit 3: Web-Design" FKSA

1. FKSA: Ability to create a set of specifications for a web page given the intent of the web page
    a. The set of specifications for a web-page
    b. The identification of inputs and outputs of a problem

While we are thinking about what we can observe students doing and producing, it's natural to think about important features of the *tasks* that measure them. It's also natural to think about ways a task can be varied: to make it easier or harder, to remove barriers due to language or culture, or other issues. The parts of an ECD design pattern that capture these are characteristic and variable features. Characteristic features are features that any task developed should incorporate, while variable features (as the name implies) are features that can vary across the task. How features vary depends on the measurement goals for the task, and could involve changing the difficulty of an item or allowing for additional KSAs to be measured. Specifying the variable features ahead of time helps to highlight decisions that should be made when developing items. Characteristic features specify the required features of a task that will elicit evidence of the FKSAs.

Example Characteristic Features for a "Unit 3: Web-Design" Task

1. FKSA: Ability to create a set of specifications for a web-page given the intent of the web-page
    a. Each task must provide students with an overall intent for the web-page
    b. Each task must ask students to generate specifications

Example Variable Features for a "Unit 3: Web-Design" Task

1. FKSA: Ability to create a set of specifications for a web-page given the intent of the web-page
    a. The intent of the web page
    b. The level of detail required from the student in the set of specifications
    c. The format required of the specifications
    d. The amount of scaffolding provided for the development of the specifications

Once we have specified all of these parts, the design pattern is usable for creating specific items that can be given to a student, scored, and used to make claims about what a student knows or can do. This work involves determining which FKSAs to measure, developing items for each of the FKSAs, and making sure the considerations stated in the design pattern are met. After that, there's a lot of work in piloting and validating the items with students and going back to the design pattern to update it if needed.

Assessing Knowledge in Exploring Computer Science

One of the benefits of the design pattern is they can live on and be used for additional assessments in new contexts and new delivery forms. Lessons learned from classroom implementations—including teacher's opinions on how their students will perform on the assessment—can be used to update the elements in the design pattern – which will aid in the development of the next assessment.

Let's end by going back to the beginning: How does domain analysis impact assessment development? Consider that the computer science community has decided that the kind of understanding that students should have about these important computational practices goes beyond recalling facts, or giving inputs to a program and predicting its outputs. Drawing from the "5 E Model" of inquiry-based learning that ECS is built upon, students should be able to *explain*, *elaborate,* and *evaluate* what they are learning. Thus, the focal KSAs underlying the ECS assessment tasks emphasize these inquiry skills in the context of computer science learning. These types of skills are not as straight forward to measure as knowledge skills, and the design patterns we create can help organize information to aid in determining how to measure these concepts.

And how do these inquiry-oriented FKSAs then impact the form of a task? Building on current assessment research, which suggests that constructed-response tasks produce a more valid measure of the integration of inquiry skills and computer science conceptual knowledge, the assessments we developed for ECS used a small selection of mainly constructed response tasks. While we understand that short answer questions place a heavier burden on students' ability to read and understand the scenario for each assessment task, and to interpret the instructions, we believe that the ability to provide explanations (over picking reasonable explanations) is a critical skill. The design pattern work (characteristic and variable features) helps assessment designers aid students with scaffolds, such as diagrams and pictures, for each task.

Evidence-centered design gives us a sense of confidence about our argument from evidence when tasks are used in ECS classrooms. Design patterns help us both explain what is important to measure and give us guidelines for how to go about measuring it.