Decompressing and Aligning the Structures of CBI Design

Joel W. Duffin
Utah State University
jwduffin@yahoo.com

Andrew S. Gibbons
Utah State University
gibbons@cc.usu.edu

Abstract

CBI researchers have long pursued goals of adaptivity, generativity, and scalability. The architecture of a CBI product has a large influence on how well it achieves these goals. This paper introduces principles called “decompression” and “alignment” that can be used to guide the design of a CBI product’s architecture.

1. CBI designers often compress their designs

CBI designers tend to think of their designs in monolithic terms because of the tools they use. However, thinking in terms of structures allows “decompression” of the designs into structural layers. When this happens it is clear that structures at one layer place requirements on structures at other layers. This suggests that there should not only be alignment of components within the structures of a given layer, but also alignment of components and structures between layers. This principle guides the appropriate modularization of program modules.

2. Decompressing and aligning the structures of a CBI product can help make the product adaptive, generative, and scalable

SRI researchers observe “individual projects tend to consider one structure dominant and cast other structures, when possible in terms of the dominant structure. Often, but not always, the dominant structure is the module structure.” One reason for this is that the module structure tends to mirror the project (work assignment) structure. In CBI this is not the case, division of labor typically falls along areas of expertise such as subject matter expert, instructional designer, computer programmer, and graphic designer. Nevertheless, discussions about CBI architectures are often carried out only in terms of the structures used by the computer programmer. We believe that in order to accomplish CBI goals of adaptivity, generativity, and scalability, architecture discussions should include representations of those structures important to stakeholders other than the computer programmer. In addition, the connections between non-programming structures and programming structures should be defined in ways that are traceable through their alignments.

2.1. Decompression involves considering the structures of a CBI product independently before collectively

When developing stand-alone CBI products, all structures used to represent the design are eventually combined and embodied in constructs offered by the media-logic tools used to create the CBI product. The process of combining the structures of a design is referred to as “compression”.

In a decompressed design process, decisions at each layer are considered first independently and then with respect to their influence on each other in light of priorities of the project. This allows designers to give attention to the details at each layer of a design.

2.2. Alignment involves creating modularizations at each layer that do not conflict with modularizations at interdependent layers

The alignment of CBI design layers refers to how well the components at each layer of design match up with and support the requirements established by the constructs used at the other layers of design. Design layers are aligned through linkages or articulation points analogous to where one structure of a building joins with another. Most designers compress their consideration of content, strategy, and representation by giving first priority to decisions about media-logic.

3. The Mystery Box simulation: A case study demonstrating decompression and alignment

---

1 For the 4 page paper submitted (has references), see http://cc.usu.edu/~sllcd/Papers/ICALT2001/ICALT2001.doc
For the purpose of research on instructional feedback we created a “Mystery Box” simulation environment in which learners solve complex problems situated in the context of an electric laboratory. To accompany the simulation environment we developed an intelligent agent that delivered detailed feedback and commented demonstrations of expert performance. The program logic for the intelligent agent is separate and independent of the program logic for the simulation environment. This is an example of program modularization along the lines of layers of a design. It also demonstrates aligning the modularizations at non-program logic layers with program logic modules. Figure 1 shows the Mystery Box user interface:

![Figure 1: Mystery box user interface](image)

The Mystery Box simulation presents an electrical laboratory in which learners are given the task of discovering what is contained in six mystery boxes.

### 3.1. The Mystery Box aligns modularization across several design layers

Figure 2 shows the Mystery Box program modules. The simulation environment program module embodies structures from the model and representation layers. The pedagogical planner (strategy layer) uses the problem generator (problem layer) to create a problem to pose to the learner in the simulation environment. As problem solvers work in the simulation environment, their actions produce event descriptions that are passed to the event interpreter (expert performance model layer). The event interpreter translates event descriptions into data that the pedagogical planner uses to decide and carry out responses to problem solving steps. The pedagogical planner determines the type of instructional message to deliver and activates the logic needed to deliver it.

![Figure 2: Mystery box program modules](image)

For some program modules there is a one-to-one mapping between them and a component at single other design layer. For others, decisions at multiple layers are embodied in a single program module. Modularization at the program logic layer comes from modularizations at the other design layers. This case study demonstrates how identifying appropriate design layers, modularizations, and alignments, facilitates creating product architectures that are adaptive, generative, and scalable.

![Figure 3: Compression of the mystery box structures](image)