Agile Learning Object Development

Ruben R. Puentedura, Ph.D.

Definitions - I

- A Learning Object:
  - Must be modular;
  - Must be repurposable, recontextualizable;
  - Must be metadata capable.

Definitions - II

- An agile process:
  - Should be “lightweight”;
  - Should be “fast”;
  - Should be “responsive”.
- These are all very nice things, but a little too general to accurately define an agile process - we’ll fix this in a moment...

Why is this needed?

- Issues of personnel
- Issues of cost
- Issues of appropriateness
- Issues of timeliness
- Issues of sustainability

Origins of the Agile Approach (Part I)

- Brooks: *The Mythical Man-Month* (1975)
  - "The man-month is a fallacious and dangerous myth, for it implies that men and months are interchangeable."
  - "A small sharp team is best – as few minds as possible."
  - "Plan to throw one away; you will, anyhow."
- Alexander et al.: *A Pattern Language* (1977)
  - "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice."

Origins of the Agile Approach (Part II)

- Gamma, Helm, Johnson, Vlissides: *Design Patterns* (1995)
    - Translates Alexander's approach to design from architecture to programming.
- Beck: *Extreme Programming Explained* (1999)
    - Defines a programming methodology stressing:
        - small (3-10 people) teams, working (preferably) in a single room in pairs, with expert customer representatives on-site;
        - short (3-week) programming cycles, with emphasis on constant refactoring and usable code produced at the end of each iteration;
        - "stories" jointly developed and negotiated by the customer and the team as the basis for development, and providing the bulk of documentation.
- *The Agile Software Development Manifesto* (2001)
    - Written (via an informal Delphi-type process) to determine what the different lightweight development processes have in common.
    - Comprises four values and twelve principles.

Definitions - II (Revisited)

- Agile Development Processes value:
  - **Individuals and interactions** over processes and tools;
  - **Working software** over comprehensive documentation;
  - **Customer collaboration** over contract negotiation;
  - **Responding to change** over following a plan.

Source: Agile Software Development Manifesto, 2001

Alexander's Patterns

- The definition of a pattern comprises:
  - Its name;
  - An illustrative picture;
  - An introductory contextual paragraph, explaining how it helps complete other patterns;
  - A summary of the problem;
  - The development of the problem;
  - The solution to the problem, describing the field of physical and social relationships needed to solve it;
  - A diagram to aid in visualizing the solution;
  - A final paragraph linking the pattern to other patterns in the language.
- Alexander's approach is fundamentally oriented towards people taking control of architecture by providing them with a process for developing an awareness of their own pattern languages.

Patterns - An Example (I)

102 FAMILY OF ENTRANCES*

. . . this pattern is an embellishment of CIRCULATION REALMS (98). CIRCULATION REALMS portrayed a series of realms, in a large building or a building complex, with a major entrance or gateway into each realm and a collection of minor doorways, gates, and openings off each realm. This pattern applies to the relationship between these "minor" entrances.

✣ ✣ ✣

When a person arrives in a complex of offices or services or workshops, or in a group of related houses, there is a good chance he will experience confusion unless the whole collection is laid out before him, so that he can see the entrance of the place where he is going.

Source: Alexander et al., A Pattern Language *(1977)*

Patterns - An Example (II)

In our work at the Center we have encountered and defined several versions of this pattern. To make the general problem clear, we shall go through these cases and then draw out the general rule.

1. In our multi-service center project we called this pattern Overview of Services. We found that people could find their way around and see exactly what the building had to offer, if the various services were laid out in a horseshoe, directly visible from the threshold of the building. See *A Pattern Language Which Generates Multi-Service Centers*, pp. 123–26.

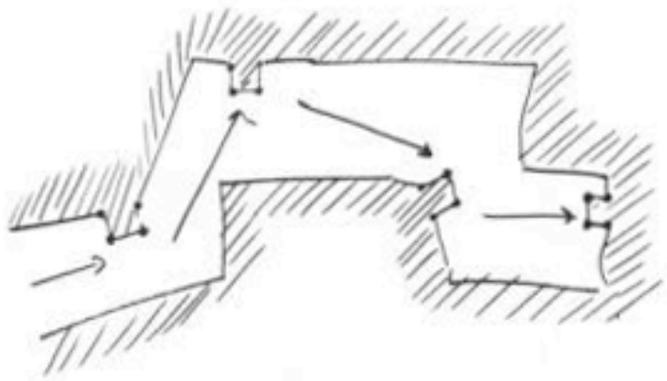
Overview of services.

Therefore:

Lay out the entrances to form a family. This means:

1. **They form a group, are visible together, and each is visible from all the others.**
2. **They are all broadly similar, for instance all porches, or all gates in a wall, or all marked by a similar kind of doorway.**

family of entrances



✣ ✣ ✣

In detail, make the entrances bold and easy to see—MAIN ENTRANCE (110); when they lead into private domains, houses and the like, make a transition in between the public street and the inside—ENTRANCE TRANSITION (112); and shape the entrance itself as a room, which straddles the wall, and is thus both inside and outside as a projecting volume, covered and protected from the rain and sun—ENTRANCE ROOM (130). If it is an entrance from an indoor street into a public office, make reception part of the entrance room—RECEPTION WELCOMES YOU (149). . . .

Source: Alexander et al., A Pattern Language *(1977)*

What Should an Agile Learning Object Creation Process Look Like?

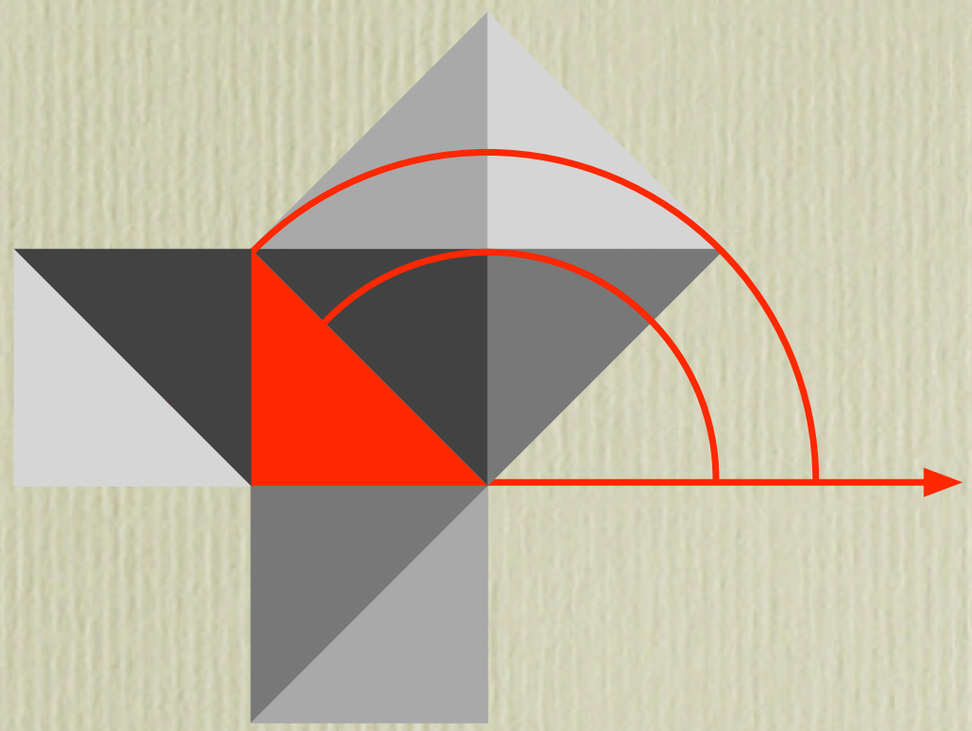
- It should:
    - Be focused on faculty, students as *reciprocal* customers and producers;
    - Reflect dialogue as a *production process*;
    - Be keyed to incorporating the design process *into* the regular course planning process.

What Should an Agile Learning Object Creation Toolkit Look Like?

- It should favor small, compact, focused tools - for example:
  - Photoshop Elements over Photoshop
  - Mellel over Word
  - iMovie over Final Cut Pro
- Integration tools are crucial:
  - Pachyderm - ideal integration tool
  - NoteTaker - excellent structuring tool
  - Multiple choice, match, etc. templates are **not** integration tools!
- Other key components:
  - Screen capture tools (e.g., Snapz Pro X)
  - Screen annotation tools (e.g., Ultimate Pen)
  - Remote screen sharing tools (e.g., SnapperHead)
  - Remote collaboration tools (e.g., iChat, SubEthaEdit)
  - Media databases (e.g., iPhoto, iTunes)

Let's Create a Learning Object...

Hippasus



- http://www.hippasus.com
- rubenrp@hippasus.com