JOURNAL OF OBJECT TECHNOLOGY

Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2003**

Vol. 2, No. 1, January-February 2003

Hello World! Goodbye Skills!

Mahesh H. Dodani, IBM Global Services, U.S.A.

1 THE ART OF BUILDING SKILLS

"Apply yourself. Get all the education you can, but then...do something. Don't just stand there, make it happen." – Lee Iacocca

Building skills is difficult. Building "do level" skills that can be applied on complex application development projects are even more difficult. Building skills that establishes the foundation for continual change and enhancement is extremely difficult. Skill building is very dependent on hands-on exercises or labs that are part of the training program. Most training companies boast 50% hands-on exercises as part of their courses, suggesting that having that ratio of exercises to lectures is the only ingredient in determining the effectiveness of the skill-building program. In reality this simple characterization of exercises hides the complexity of designing and delivering effective skill building programs.

Since the inception of programming and application development, there has been an ongoing discussion of what is the best way to build skills. For example, consider the issue of how difficult should the exercises be? Exercises can range in difficulty from the obligatory "Hello World" introductions to programming and concepts (see http://www2.latech.edu/~acm/HelloWorld.shtml for descriptions of such an exercise for almost any computer language/environment) to immersion in a complex simulation or environment where the student can experience and learn through scenarios which mimic the actual projects that they will apply their skills to. Besides the obvious pros/cons of each extreme (e.g. "Hello World" exercises are real good to get a basic understanding while "immersion" exercises are real good to learn how to apply the skill in a particular context), the main problem with tackling this issue of complexity is based on considerations such as learner type, instruction style, pedagogical design and skill level. So, for example, the "Hello World" exercises might be suitable for beginners who like to be "hand held" through the exercises, in a "sage on the stage" type of instruction targeting "do with assistance" skills. This complexity is further exacerbated by the many dimensions of the design and delivery of exercises for skill building that must all be taken into consideration.

Lets start by establishing the ground rules for the discussion by enumerating the requirements for skill building labs/exercises:

- Labs/exercises for skill building have the following seemingly contradictory requirements:
    - Need for "hand holding" detailed instruction labs for the beginners vs. more "free form" instructions for DIY (Do It Yourself) developers
    - Need for "real world" application scenarios vs. "made up" simpler scenarios
    - Need for developing skills in the context of "best practice" vs. developing "hello world" get-your-feet-wet type skills.
- Part of the complexity of skill building as it applies to application development is the complexity of the technologies and products that are involved. So, for example, a typical e-business application will require the proper use of technologies (Java, XML, HTML, J2EE, etc.), products (web servers, application servers, messaging middleware, backend applications, commerce engines, portal servers, etc.), and frameworks (Model-View-Controller, struts, etc.) The kind of application being built has an impact on how the particular technologies, products and frameworks are used in conjunction with each other. So, for example, there is a big difference in how the technologies and products are used to build self-service business-to-consumer vs. application integration business-to-business applications.
- The labs/exercises must be flexible in order to be customized in different ways to address particular needs of a customer or student, including:
    - Skipping over some exercises that are not needed.
    - Changing some aspect of the exercise itself to reflect how the components are used in the context of the customers' architecture, methodology, tools, etc.
    - Adding a new lab/exercise.
    - Changing the case study.

Note that the above requirements are from both the perspective of the learner as well as the training organization. The exercises portion of a course are usually the most expensive to maintain as technologies, products, and frameworks evolve.

The requirements above set the stage for the rest of the discussion. In the next section we show the state of the art in designing and delivering exercises/labs for building skills. The final section shows the next steps that must be taken in order to support effective skill building.

2 THE STATE-OF-THE-ART IN SKILL BUILDING

The state of the art in skill building is to ensure a proactive learning environment, which has the following characteristics:

- Learner centric – the learner is in charge of the learning. The learning environment diagnoses problems that the learner is having, helps them to overcome these problems, and guides them to build their skills by doing.
- Exercise oriented – the exercises drive the learning. By doing the exercises the learner builds skills on the subject, gains a deep understanding of the subject, and most importantly understands how to apply the skills to solve problems.
- Facilitated learning – the learner is guided through their skill building journey. The learning environment facilitates an understanding of the learners' viewpoint, describes concepts/issues/solutions in terms that the learner can understand, and guides them effectively on building skills.
- Shared, collaborative experiences – skills are built in a context of a team. The learning environment takes into account the needs of understanding how to apply skills in a team-based project.
- Immersion in a simulated environment – that very closely mimics the environment in which the skills that are being built will be applied. The importance of the immersion in a realistic environment is to allow learners to build skills that that are immediately applicable. The learners understand how to apply their skills in their own workplace.

The "design" approach for labs/exercises takes into consideration the “learning object” philosophy (see http://www.jot.fm/issues/issue_2002_11/column3 for a discussion), and includes the following:

- A realistic case study against which labs/exercises are written. This case study is based on architectures and uses frameworks/patterns that reflect the current best practice in developing applications.
- A template for writing the labs/exercises that separates high-level instructions from the detailed instructions to perform these steps. This template facilitates flexibility through the ability to suppress the detailed sub steps for the more advanced DIY developer audience.
- Applying a "learning object" incremental and iterative design process as shown in Figure 1. This process builds skill objectives incrementally, and within each skill objective the following "lab" objects:
    - Introductory: these labs introduce the component, tool, or technique – usually as a facilitated walkthrough or demo.
    - Skill Building: these labs focus on building the skills defined in the objective at the intended "level".
    - Advanced: these labs allow more advanced students to explore areas not covered by the skill objective or allow more practice with the skill objective.

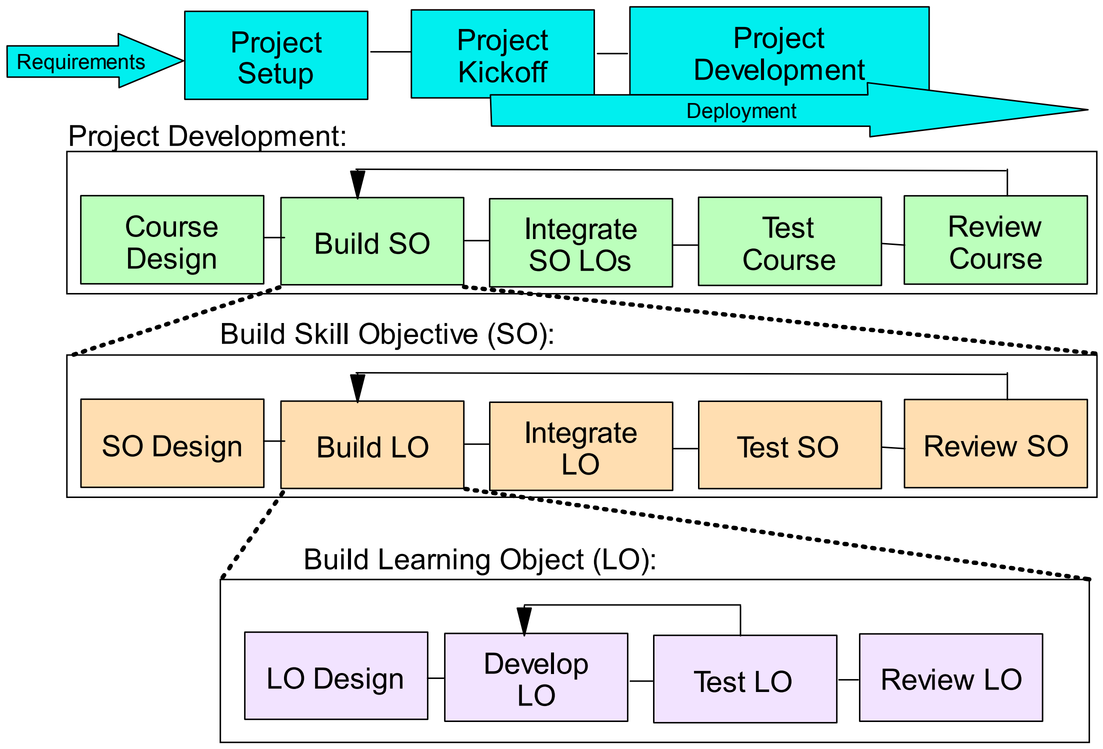


Figure 1: An Iterative, Incremental Development Process

The state of the practice is best summarized by the many patterns for education and training that target skill building labs/exercises (see http://st-www.cs.uiuc.edu/users/chai/research/Anthony.html and http://www-lifia.info.unlp.edu.ar/ppp/index.html for the many patterns already in use.) On the other hand, many anti-patterns for skill building labs/exercises have also been identified (see http://csis.pace.edu/~bergin/sol/oopsla98ed/AntiPatterns.html for examples.)

Even with these advances in designing and delivering labs/exercises for skill building the following issues still persist:

- The design approach does not result in effective skill building labs/exercises. They result in learning that either fall in the one extreme category of being too simple (so beginners like it and DIY developers don't) or too complex (so beginners don't like it and DIY developers do.) Furthermore, the design of an entire course is so "jam packed" that it leaves no "wiggle room" to allow the instructor and students to explore doing exercises in different ways or have discussions on it, or provide for practice sessions to "beef up the skills".
- Even though these approaches have tried to design and develop the labs/exercises with flexibility in mind, the reality is that it is difficult to change the exercises, and furthermore, changes are still costly. For example, it is still difficult to add an

exercise or change an exercise or provide different versions of the same exercise (e.g. on different tools or addressing different skill levels.)

- The design approach does not support the deployment of the labs/exercises in a way that facilitates changes to the case study, exercises, and product/tool environment to be made effectively and efficiently.

3 NEXT STEPS IN ENHANCING THE STATE-OF-THE-ART

So, what can we do to address these issues with the labs/exercises that are integral to skill building? Well, here are some recommended next steps in addressing issues with design, and flexibility:

Design considerations: The next iteration of "hardening" the design philosophy of building lab LOs at the introductory, skill building and advance levels requires the following (re)considerations:

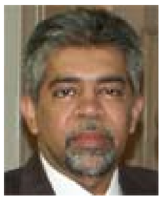
- Make sure there is an introductory lab defined for each skill objective. Even if it can not be used as part of the regular agenda (due to time constraints) having such an exercise allows the instructor the flexibility of using this exercise to "walk the class through" their first steps in building the skill – which they can do as part of the "lecture". The instructor can use this exercise as is if they feel they have a "weaker" class that needs a simpler exercise to get them started. The introductory labs will be of the "hello world" flavor, where the emphasis is on showing the mechanics of building the component using the tools rather than on the logic in the component.
- For the skill building exercises, divide it into two parts – the first should be on building skills that are focused on the component itself (so, without the complexity of the component living in an architecture, framework, or pattern) and the second should be on building skills that are focused on the component integrated with the other components (so, the component now lives in the appropriate architecture, framework, or pattern.) This is akin to having unit vs. integration testing – one tests the unit itself, while the other tests the unit in the context of the entire integrated environment.
- Make sure that there is at least one advanced exercise defined for each skill objective. Even if the advanced exercise can not be done in class, it provides the advanced students with the opportunity of doing this outside of class time (during the class) or after the class as take-home exercises. These exercise/labs would be more of the DIY flavor, and therefore do not need the "detailed" steps built in – however, there is a need to provide "one solution" that has been tested.

Flexibility considerations: The more difficult aspect is to make the labs/exercises more flexible. In essence, the most effective approach is to actually go through the process of making significant changes to the labs/exercises, understanding from these significant changes what causes the labs/exercises to be inflexible, "refactoring" the labs/exercises to address these inflexibilities, and applying the "refactored" approach to handling another change to see if it really helps. In essence, "flexibility" should be addressed at the following levels:

- Individual Exercise Level: how can we address different "usage scenarios", "components", and tools/products. The focus here is to identify a typical set of "change cases" that must be handled on an ongoing basis, and how to facilitate supporting these changes effectively in the labs/exercises.
- "Best Practice" Level: how can we address different frameworks/patterns, e.g. using Struts, J2EE Patterns, etc. The focus is here is how to "separate" the architecture/framework/patterns from the case study, which will allow us to support the two different types of skill building exercises mentioned above, and any new frameworks/patterns.
- "Case Study" Level: how can we address different case studies, especially to address particular industries (e.g. insurance), different business patterns (e.g. business-to-business and business-to-consumer applications) and the mentored workshops. The focus here is to define what it takes to build a case study at the curriculum level, ensure that the labs/exercises can effectively "separate" the skill being developed from the case study being used to develop the skills, and the steps in incorporating a new case study for the curriculum.

In summary, designing and delivering exercises for effective skill building is difficult. Advances in applying the learning object and other reusable OO framework approaches to building exercises will increase their flexibility and effectiveness. We have a lot of work to do to facilitate the "right skills" at the "right time" using the "right medium" to build the "right skills!"

About the author



Mahesh Dodani is an e-business architect with IBM Global Services. His primary interests are in helping individuals and enterprises transform themselves from their current environment into being an e-business. He can be reached at dodani@us.ibm.com.