Joe Klemmer

Jill Ellen Coddington

SG 150

July 13, 2009

Methodologies Used in Game and Application Software Development

As the demands for software development become more complex, there is a need to have methodologies in place in order to ensure timely delivery of quality products. In recent times, five development methodologies have risen to become prominent in the field: Object Orientation, Rapid Software Development, Design Patterns, the Unified Modeling Language (UML), and the Agile Software Development. This paper will explore each of these methodologies, their strengths and weaknesses, examples, resources and how some combinations of them might lead to a more productive development process.

The first methodology is the oldest on the list. Object Oriented Programming (OOP) and Object Oriented Design (OOD) are development models built around manipulating objects rather than focusing on the procedures used to manipulate data. “Object-oriented programming takes the view that what we really care about are the objects we want to manipulate rather than the logic required to manipulate them. Examples of objects range from human beings (described by name, address, and so forth) to buildings and floors (whose properties can be described and managed) down to the little widgets on your computer desktop (such as buttons and scroll bars).” Using object allows for better reuse of software and isolation of errors.

Next, there is Rapid Software Development, also known as Rapid Application Development (RAD). “[RAD] involves using development environments that include powerful tools to support system production. These include database programming languages, form and report generators and links to office applications.” The idea behind RAD is to use as much prebuilt functionality as possible to assemble a system. This can include, but is not limited to, Commercial Off-the-Shelf (COTS) software, prebuilt libraries, visual development systems and custom code.

Design Patterns first became popular with the book “Design Patterns Elements of Reusable Object-Oriented Software” by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides published October 1994 by Addison-Wesley. The definition of Patterns, as pertains to software development, is “a reusable approach to solving a commonly-occurring software design problem. A design pattern is a template for solving a problem that can be used in many different situations. Object-oriented design patterns illustrate relationships and interactions between classes or objects, without specifying the final application classes or objects that are involved.” (Bolton)

The Unified Modeling Language (UML) is “[a]n object-oriented analysis and design language from the Object Management Group (OMG).” This methodology uses visual description entities, known as diagrams, to show the organization of the system or program being modeled. With the UML, you can show the relationship of classes and objects from different perspectives. Since the UML models objects, not processes, it can be abstracted out to show how the development process itself can be organized.

Over the last decade, a new methodology has become popular for development. It is called Agile Software Development and approaches the development process from a different perspective. The Agile method focuses on rapid development of small, incremental changes and using a test first approach. The most popular Agile method is Extreme Programming (XP). XP has, at its core, some basic methods that are different from the design, build, test, deliver used in the past.

* Pairs of programmers program together
* Development is driven by test. You test first, then code. (If the tests don’t run, you aren’t done)
* Pairs do not just make test cases run. They also evolve the design of the system.
* Integration immediately follows development, including integration testing.

Each of these methodologies have their strengths and weaknesses. For OOP, its strength is that it uses a “real world” view of objects, which are built from classes. This allows for the representation of the software in a way that closely follows the real world. For example, a car is an object that is made up of other objects. A car has an engine (class). An engine has a distributor cap (class). The representation of the distributor cap knows all about itself so that the engine does not need to handle the internal workings of the cap. However, a disadvantage is that “the Class hierarchy is not properly designed, all the OO advantages disappear.”

RAD models have an advantage due to their rapid and incremental development cycles. Programs can be built fast with only minor updates. Delivery of some functionality to the customer happens regularly with the benefit that the customer can give immediate feedback on it. However, as Purcell writes, “The big disadvantage of the Rapid Prototyping Model is that the process can be to fast, and, therefore, proper testing (especially security testing) may not be done.” If the development of the system is distributed over many teams and/or distances, the coordination of each piece of the system can break down.

{Research into Design Patterns was sketchy. There are plenty of resources in both book and web formats. However, circumstances interfered with finding them for this essay.}

UML is a great way of showing object and system relationships in a visual style that is relatively easy to understand. Since it is a model that can fit every aspect of software development, from the coders in the trenches to project managers to analysts and designers. It can be used from the object/class level to the overall system and even business case designs. “The UML gives everyone from business analyst to designer to programmer a common vocabulary to talk about software design.” Its down side is that, for non-programmers like users and business management, it can be very confusing to understand. The complexity of large systems can also make UML unwieldy.

Agile methods, like Extreme Programming and SCRUM, are similar to RAD in that development is done in increments with fast turnaround times. Other advantages are that testing and code validation are tightly integrated into the development process with the developers and users working together. In addition, the iterative nature of the coding makes it more adaptable to changing market conditions or requirements due to business process adjustments. There are disadvantages to the Agile method, though. The idea of customer involvement often does not materialize or is inconsistent at best. Users still have their “regular” duties and can be to busy to participate in the development process. Prioritizing the changes and updates can be extremely difficult when there are different areas of the customer organization involved. Each organizational area will have its own idea of what is important.

These development methodologies all have one thing in common; they have been created to improve the quality and process of software development. In most cases, they do a good job of accomplishing this goal. In general, productivity is up and costs are down on projects using these methodologies. Recent times have seen hybrid methods coming to the fore with some of the different models used, in part or in whole, in combination with each other. This combining can help balance one method's strengths with another’s weaknesses. Thus giving a best of breed solution.

Works Cited

Beck, Kent. eXtreme Programming eXplained. Addison-Wesley, 2000.

Bolton, David. "Pattern - Definition." About.com. 10 July 2009 <http://cplus.about.com/od/glossar1/g/patterndefn.htm>.

Miller, Randy. "Practical UML: A Hands-On Introduction for Developers ." 1 December 2003. Embarcadero Developer Network (EDN). 13 July 2009 <http://edn.embarcadero.com/article/31863>.

Miller, Steve. "Exploring Agile Development." March 2007. Pragmatic Software Co., Inc. 13 July 2009 <http://www.pragmaticsw.com/Newsletters/newsletter\_2007\_03\_SP.htm>.

Purcell, James. "Comparison of Software Development Lifecycle Methodologies." 12 February 2007. Global Information Assurance Certification. 13 July 2009 <http://www.giac.org/resources/whitepaper/application/217.php>.

SearchSOA.com. "What is object-oriented programming?" 19 August 2008. SearchSOA.com. 7 July 2009 <http://searchsoa.techtarget.com/sDefinition/0,,sid26\_gci212681,00.html>.

Sommerville, Ian. Software Engineering 8. 8. Essex: Addison-Wesley, 2007.

The Computer Language Company Inc. "Definition of: UML." 1981- 2009. PC Magazine Encyclopedia. 11 July 2009 <http://www.pcmag.com/encyclopedia\_term/0,2542,t=UML&i=53392,00.asp>.