Rob's Data Dump

Exploring Old and New Digital Worlds

Feature-Driven Development (FDD)

Introduction

Jeff De Luca, the creator of Feature Driven Development (FDD), once stated that certain methodologies resonate with developers (De Luca, 2008), and I must admit that at first, FDD did not resonate with me. My first impression of FDD was that it was a method to extract user stories, in the form of “features”, from use case diagrams. Unfortunately, further research, specifically targeting FDD, did not help; everything I read seemed to skip the critical step of how to derive the features in the first place. Luckily for me, the phrase “model-driven engineering” continued to appear. Pulling that thread, I discovered Peter Coad’s work on object-oriented analysis and design (OOAD). Now, I could trace an intuitive path from object-oriented patterns to FDD, and I must say, that FDD appears to be a very effective methodology. Here, I will discuss what is FDD, its advantages and disadvantages, and in what environments it would be effective or appropriate.

Background

In 1995, the United Overseas Bank (UOB), based in Singapore, hired IBM to automate the UOB’s process for making loans. After two years of development and little progress, though, IBM decided it had underestimated the complexity of the task, and informed the bank that IBM could not complete the project, and that IBM doubted the project itself was feasible (De Luca, 2014). However, Jeff De Luca, a former IBM systems engineer who had been sub-contracted by IBM to assist in the project, disagreed. He believed that he and his team at Nebulon could complete the project, using object-oriented design (De Luca, 2014). De Luca advocated using the class-based, and object-oriented Java programming language, as well as the Yourdon/Coad approach to OOAD. OOAD is partially based on Coad’s paper, “Object-Oriented Patterns”, where he postulated that objects have “patterns” or characteristics that cause them to fall within seven different categories, and each of

these objects had “attributes” (i.e., what the object knows) and “services” (i.e., what the object does) (Coad, 1992).

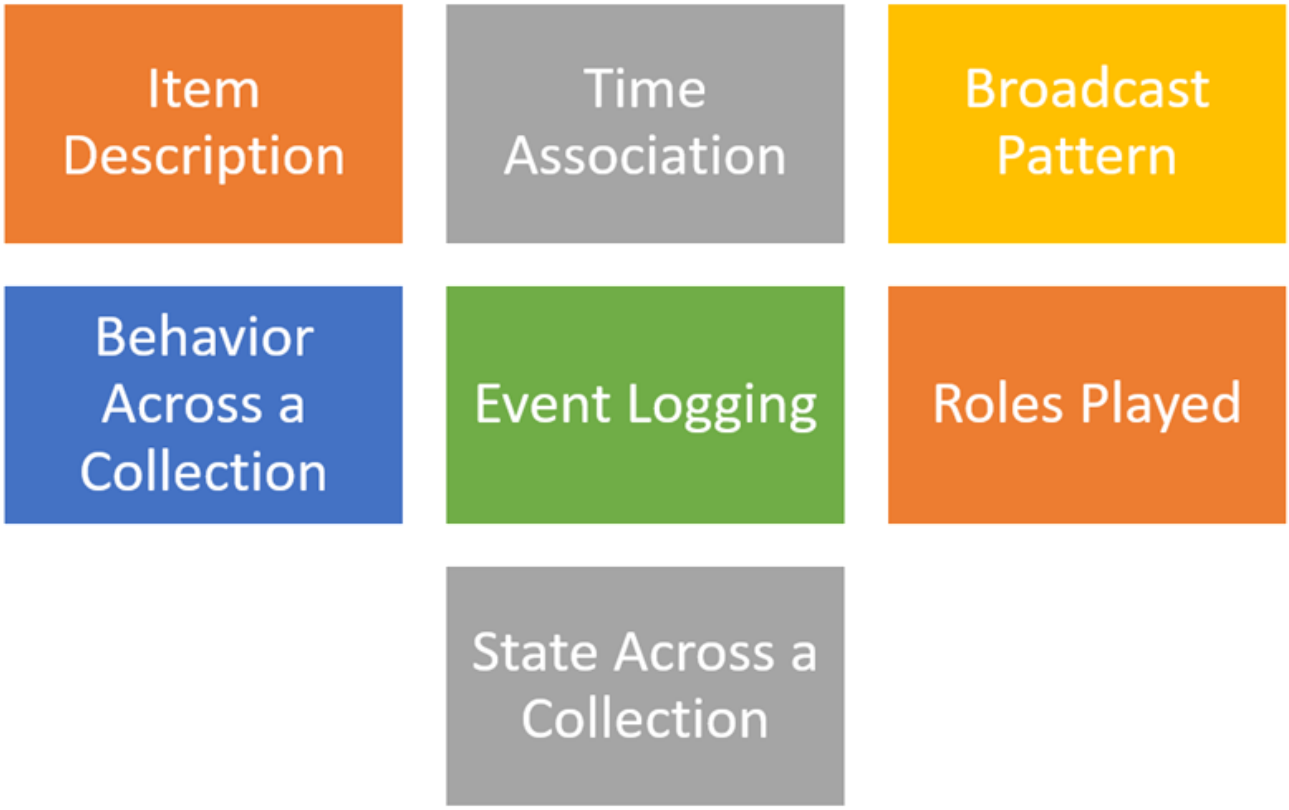

Figure 1. Original seven archetypes

De Luca stated that by identifying these objects and their interactions, as well as determining their attributes and services, he could create an appropriate domain model that could translate the byzantine paper process that UOB used to make loans into a smooth, digital system. De Luca actually had Peter Coad fly into Singapore and work with the Nebulon team for several weeks on the project, during which time he and Coad worked on integrating Coad’s concepts with De Luca’s methods (Gliedt, 1999). De Luca and Coad reduced the seven patterns to four archetypes, colored to make identification on domain model diagrams easy:

- **Moment-Interval [Pink]** – This archetype references an action that may occur instantaneously (moment) or over a period of time (interval). For example, checking a balance is an example of a moment, while driving is an example of an interval.
- **Role [Yellow]** – This archetype represents who or what is participating in the moment. For example, a customer, bank officer, or supervisor object may be an instantiation of the user object, and a specific account may be the instantiation of the account object.

- **Party/Place/Thing [Green]** – An abstract object, akin to an abstract or superclass in Java. For example, a user object contains all the patterns and behaviors common to customer, bank officer, or supervisor objects.

- **Description [Blue]** – A catalog-type description of objects or elements an object can use. For example, a junction table is a catalogue-type description that associates user roles with permissions, and it may be used when the customer role is instantiated.

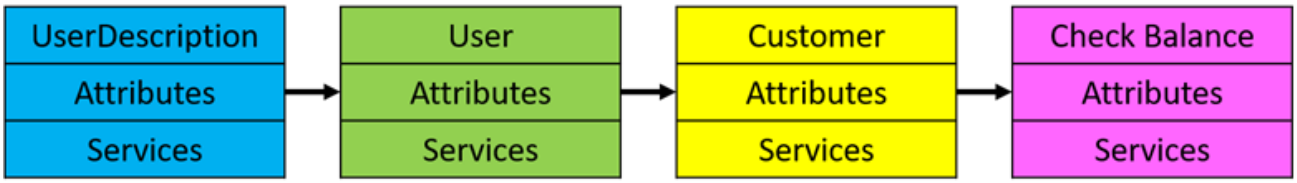

Figure 2. FDD Domain Model

Using the diagram, a development team could determine the attributes and services of each object necessary to allow the object to interact with other objects. They could then turn these attributes and services into class attributes and methods in Java. De Luca and Coad also decided to focus on the “features” of an object, which used a template to explained how the attributes and services of an object worked together to accomplish a task.

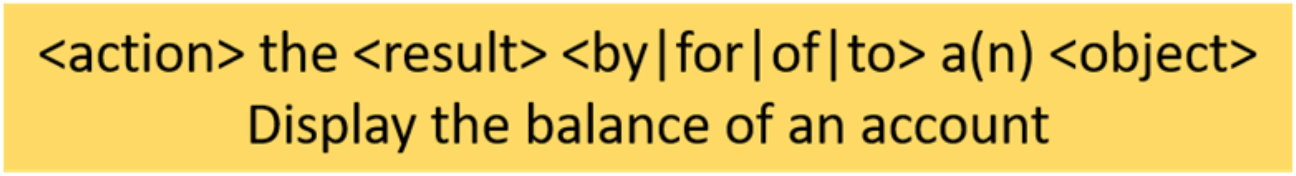

Figure 3. Feature Template

The end result was not only a successful product, but a new, incremental and iterative development process using the following activities (Nebulon, 2005):

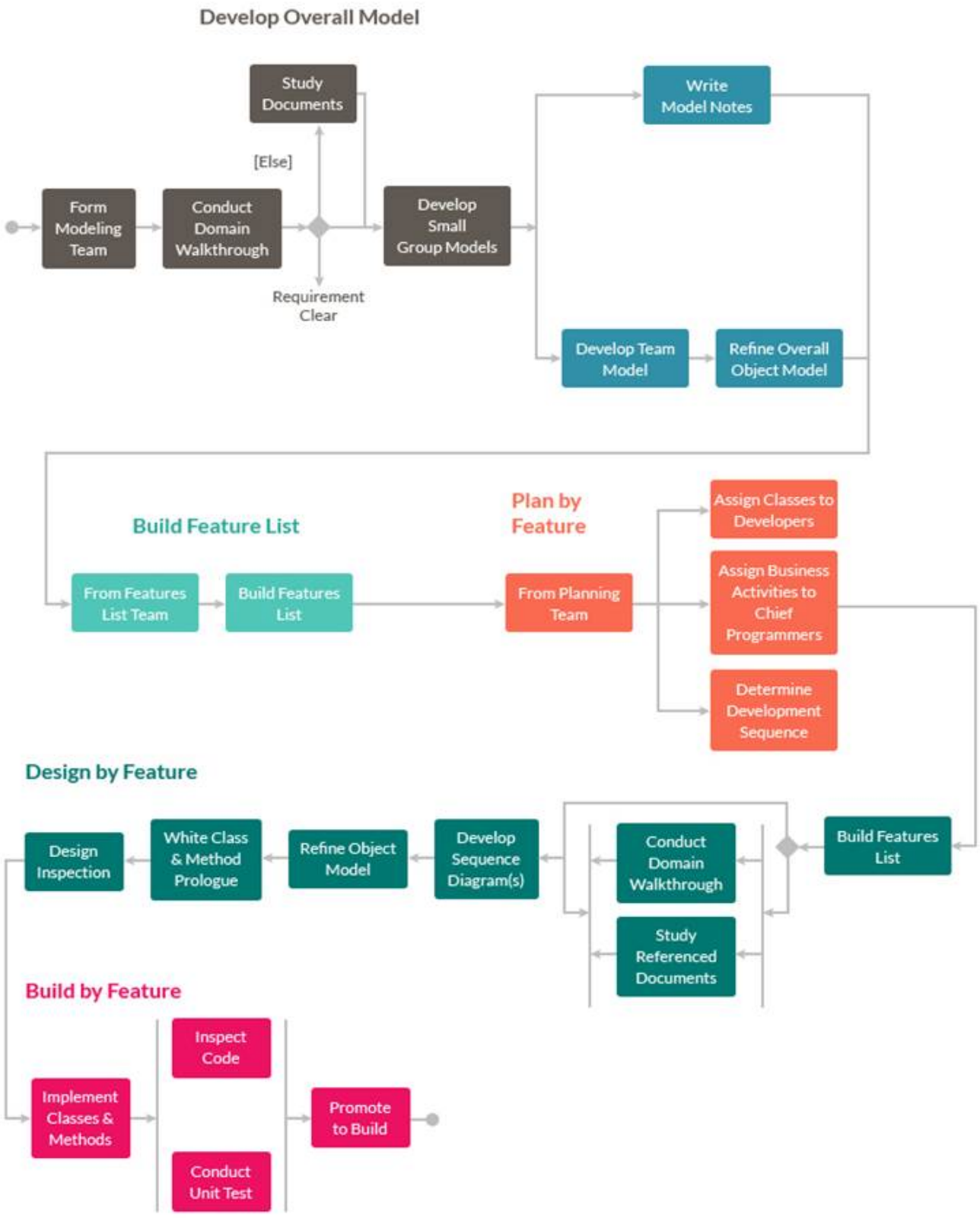

Figure 4. The Feature-Driven Development Process

Advantages

While methodologies, such as XP, focus on Role and Moment/Interval objects, such as user stories, FDD identifies other objects, such as events and data. This is a small, but

significant difference that mitigates technical debt by discovering relevant classes early on. In addition, the deep collaboration between developers and the client during development of the model mitigates misunderstandings and gives the team a better picture of what the client desires. FDD also translates very well into classes and methods used by object-oriented languages (e.g., “Create [Role] that [implements or extends Party/Place/Thing] using the [Description] model, which will perform [Moment/Interval]”). Finally, other methodologies, such as SCRUM, integrate well with FDD, especially during construction.

Disadvantages

Even though De Luca (2008) states that FDD is not a Big-Design-Up-Front (BDUF) method, a significant amount of planning and design takes place before code construction. In addition, FDD does not require a working product after each iteration. A perfect example of this is during the development of Powerlender. Midway through development, UOB officials asked for a working demonstration of the application. The team had to pause normal development and focus on creating the demo, consuming “at least a person-year of time getting ready for it” (Gliedt, 1999). However, De Luca also states that if necessary, FDD can be modified to accommodate such a requirement (De Luca, 2008).

Where Appropriate and Conclusion

Some developers claim that FDD cannot be used for small projects. I personally think that even though it may not be the best choice (i.e., XP or RAD may be better for experimental applications), FDD can be used with any project. I like the fact that I can integrate FDD’s archetypes with methods such as UP, and I can use XP or Test-Driven Development (TDD) during construction. One other bonus of FDD has nothing to do with how the method is executed, but in its history. Terry Gliedt’s tale of how Powerlender was developed is an excellent case study of software development. It covers how the team got through “death marches,” “seagull management,” and other anti-patterns; how they created FDD; how they implemented good practices, such as limiting span of control; as well as other challenges faced by project managers at all levels (Gliebt, 1999). I recommend it as a “must-read” for all budding software engineers.

References

- Coad, P. (1992). Object-Oriented Patterns. *Communications Of The ACM*, 35(9), 152-159. doi:10.1145/130994.131006
- Coad, P., Lefebvre, E., & De Luca, J. (1999). *Java modeling in color with UML*. Retrieved from https://edn.embarcadero.com/article/images/29871/colormodeling1.pdf and http://csis.pace.edu/~marchese/SE616/L2New/jmcuch06.pdf
- Cohn, M. (2010, January 11). User stories and user story examples. Retrieved from https://www.mountaingoatsoftware.com/agile/user-stories
- De Luca, J. (2014, October 31). Jeff De Luca biography. Retrieved from http://www.nebulon.com/pr/bio.html
- De Luca, J. (2008, January 12). Episode 83: Jeff DeLuca on Feature Driven Development. Interview by M. Lippert. [Audio Podcast]. *Software Engineering Radio* [Audio Podcast]. Retrieved from http://www.se-radio.net/2008/01/episode-83-jeff-deluca-on-feature-driven-development/
- Gliedt, T. (1999). Special report from tomorrow. Retrieved from http://www.hps.com/~tpg/singapore/index.php?file=powerlender98
- Nebulon Pty Ltd. (2005). FDD overview. Retrieved from http://www.nebulon.com/articles/fdd/download/fddoverview.pdf

---

Author: Rob Garcia

Ever since I spent my cousin's birthday party on a TRS-80 PC2 instead of hanging out, I've been hooked on programming. Now, after many years, I am turning my passion into a profession, and I would like to share anything I come across that may be useful, impressive, or just plain cool!
View all posts by Rob Garcia

Rob Garcia / November 12, 2017 / SWEN 603 Modern Software Methodologies