FDD and Project Management

Martin Bauer
27/05/2004
Article as published in Cutter IT Journal

This article examines the style of management embodied in Feature Driven Development ("FDD").

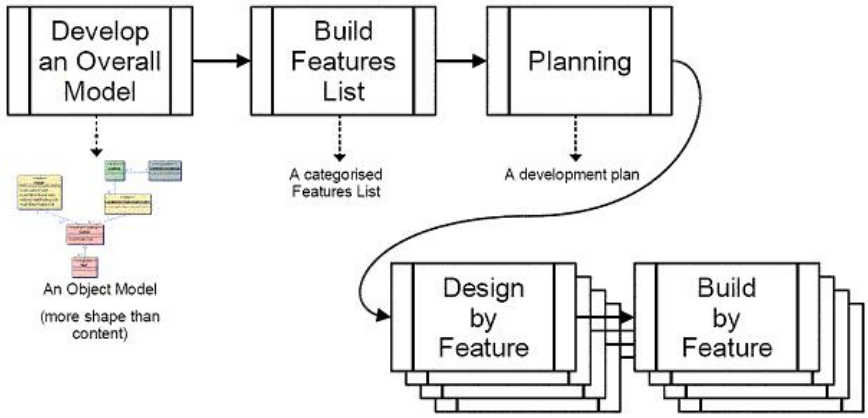
FDD is an agile development methodogolgy created by Jeff Deluca, the purpose of which he states as being

" *to enable and enforce the repeatable delivery of working software in a timely manner with highly accurate and meaningful information to all key roles inside and outside a project."*

FDD Overview Presentation http://www.nebulon.com/articles/fdd/download/fddoverview.pdf

FDD consists of 5 clearly defined processes that can be captured in 5 pages. The core of FDD is the concept of a feature which is a clearly defined Client valued piece of functionality. The processes that make up FDD are structured around defining every element of a project as a feature and then designing and building each feature in an iterative manner.

The high level structure of FDD is captured in the following diagram.



The project management approach built into FDD is based on commonsense and and the experience of Jeff Deluca in what works and what doesn't. The techniques contained in FDD are targeted at managing people because that's where the biggest problems arise and where the greatest gains can be made.

In this article, we look at each of the 5 processes and what techniques FDD uses during the project to manage people and the end result of delivering working software.

PROCESS 1 – DEVELOP AN OVERALL MODEL

A initial project-wide activity with domain and development members under the guidance of an experienced object modeller in the role of Chief Architect.

Process 1 involves the project team creating an object model of the business domain that is more shape than content. The model is not fully defined with all attributes and methods, it is more about getting the shape of the business domain captured correctly in an object model, not capturing every detail.

The process is highly collaborative process where everyone has to work together to create the overall model. It takes an iterative approach where the domain expert explains a part of the business domain. The project team is broken up into groups (preferably 3 per group) to model that part of the domain. The groups then present their models and consensus is reached on which to use. This is then repeated for each part of the domain until everything has been covered. The end result is an overall model of the entire domain.

There are a number of reasons that this approach is taken

1. it helps to ensure that there is consensus and understanding
2. to manage scope and expectations
3. to establish who is best placed to play specific roles within the project team

Both of these outcomes are fundamental and without getting them right, the risk of the project running into trouble is greatly increased.

Consensus

Getting consensus is extremely important but very difficult. In FDD, the definition of consensus can be described in this way.

" *I may not agree with the result but I agree to support it*".

The reason for this definition is that on a practical level, it is very difficult to get two people to agree let alone the entire project team. People that say that they agree but don't really are likely to cause problems later on in the project by placing obstacles in the way, not necessarily in an obvious or deliberate manner. It's more likely to be little things such that are neither constructive or helpful and simple slow the project down. Getting consensus according to this definition enables the project to move ahead with everyone onboard. It gets "buy-in".

If it is not possible to get consensus at this point, it will be very difficult to get decisions made and have people co-operate at a later stage when the pressure is on. In this process, the Project Manager's role is not to make the decisions but to ensure that decisions are made. The Client, Domain Expert and Chief Programmers are the people that need to agree for the project to work, the Project Manager is their to facilitate this outcome.

Understanding

The more the project team understands the business domain, the better equipped they are to come up with practical solutions. The reason the overall model is not detailed is that getting everyone to understand every single detail of the business domain is a time consuming and long process. And at this point, it's not a constructive use of time. The objective is to get an overall understanding. Later on, the detail required can be added and if necessary the overall model can be adjusted.

The other aspect of understanding is ensuring that it is across the entire team. Rather than have one person eg. the domain expert, explain things to the business analsyst who then documents the business domain for the architect to read, the entire project team is involved in process one. That way everyone has the same level of understanding.

Managing Scope & Expectations

One of the many challenges that a Project Manager faces is the expectations of the Client. It is not uncommon for a Client to have an idea of the scope of a project, ie. how long it should take and how much it should cost. The reality can be quite different, what seems like a simple project can be very complex, what seems like a simple feature to add can be time consuming. It might seem obvious to the Project Manager but communicating that to the Client and more importantly getting them to understand and accept the situation is very different. The group approach in Process 1 deals with managing expectations in the creation of the overall model.

The size and complexity of the model accurately reflects the size of the project. Jeff's rule of thumb is the time taken in the modelling process is 10% of the overall project duration. For example, if it takes 1 week to create the overall model, the project will take 10 weeks to complete. If after 1 week, only half of the business domain has been modelled, the Project Manager can confidently say the project will take 20 weeks. As the Client is a part of the modelling process they know how much time it has taken, how much has been covered, the Project Manager doesn't have to try to convince them that the project is much larger than expected, the Client will already know this.

The other aspect dealt with is scope. Often at the start of a project, there are lots of well intentioned suggestions and requests for additional feature, eg. a wish list of things it would be great to have. Although many of these requests might be nice to have, each will take time and effort to implement and of course cost. What might seem like a simple addition can take quite a bit of time. The way to manage this is to model that additional functionality, the Client will then see how much work is required to add that functionality. If they still want the functionality, then it can be factored into the cost and timeline of the project with the Client fully understanding what they have asked for an how much it will cost them. It creates a true sense of value.

The time to manage scope and expectations is before the project starts not during development when the cost, time and resources have already been set. The best way to manage change in a project is to have a clear idea of the scope of the project so both the client and Project Manager know what is in and out of scope. Should changes be necessary later in the project, they can be accommodated but it will be clear that the budget and timeline must also be adjusted in accordance with the change in scope. Accommodating change is not the issue but rather having to deal with more work without adjusting the timeline and budget. Process 1 provides an excellent basis for the Project Manager to deal with the issue of scope creep by clearly identifying the scope upfront.

Creating the Project Team

It is good project management to make sure you have the right people playing each role on a project. The difficulty for a Project Manager is in knowing who are the right people at the start of the project when you haven't worked with them before. The approach taken in Process 1 exposes individual strengths and weaknesses. When the domain expert explains the business domain the Project Manager has an opportunity to review how well the domain expert communicates and if they are able to succinctly define the domain. If they can't, it will be very difficult to create an accurate model. If the model is not an accurate reflection of the business domain, the end result will be flawed. If the domain expert cannot communicate well or the domain is unclear, the Project Manager would be unwise to continue as there's a high risk that the project will fail will need to seriously consider if the project should go ahead.

Another aspect of the group approach to modelling allows the Project Manager to see how the team works together as well as looking at each member of the team. The Project Manager will be able to discover personality traits, eg. people that dominate, people that are argumentative, people that don't contribute. The only true way to tell if people can work together as a team is to try it. Finding out that there are personality clashes at this point means the Project Manager can adjust the team to allow for this or remove problem people from the project team.

The tangible result that Process 1 delivers is an overall model. From a project management perspective, it provides techniques that are focused on getting, consensus, a common understand of the business domain, managing expectations & scope and identifying the right people for each role. If it is not possible to achieve one or all of these outcomes, the project is at great risk before development has even started. Process 1 ensures that these fundamental issues are identified and addressed upfront.

PROCESS 2 – BUILD A FEATURES LIST

An initial project-wide activity to identify all the features to support the requirements.

Creating the feature list is the task of the Chief Programmers involved in the modelling process. The Client and Stakeholders don't need to be a part of this process as they have made their contribution in Process 1 and now it is a matter of capturing the project in a list of features. This does not require collaboration, getting a group of people involved at this stage would not be productive or constructive. The key to this process is defining the project using the language of the business domain. This means that the Client will be able to understand and value each feature. It also enforces a common language across the project team and reduces the risk of miscommunication or assumptions.

Poor communication is the basis of most problems in software development. The language we choose has a significant impact on how effectively we communicate. There are many techniques in FDD that help to provide meaningful communication, the most powerful one is in Process 2, defining the entire project in features using the language of the domain, i.e. using the Client's language. This might sound simple and obvious but should not be underestimated. The focus that this brings is incredible and affects the project in many ways.

In FDD, everything is described as a feature which is defined as

the by/for/to an

For example

calculate the total for a sale calculate the total sales for a cashier

A feature is also defined in terms of size, ie. more than 2 hours work but less than 2 weeks work. If it more than 2 weeks work, then it should be broken down in separate features.

Having everything defined as a feature avoids the problems that occur all the time when the Client refers to something in one way and the programmer refers to it in another and the Project Manager has to continually interpret. If the Project Manager doesn't get it exactly right, mistakes happen, the programmer thinks they are building one thing, the Client thinks another. Using the same language doesn't mean the problem disappears, but it reduces the risk considerably.

Client Value Using the language of the Client means there's a focus on Client value, eg. if the feature can't be described in Client terms and isn't delivering value, then why build it? This is an excellent litmus test. Programmers often want to build things because they think it's important, Project Managers without a technical background can't always tell whether what the programmer wants to do is useful or something they want to do because they have a personal interest. Forcing everything to be defined in Client valued terms gives the Project Manager a technique to focus programmers on only doing things that are of value to the Client.

Project Tracking

A feature list provides a way of tracking projects in a meaningful manner. The Project Manager can tell the Client what has been delivered and the Client will not only understand what was delivered but the value. Explaining to a Client that they've just completed normalising the database and adding indexes for all primary and foreign keys is far less meaningful to a Client than saying we've just delivered the feature that allows you to calculate the total of a sale.

Setting Priorities

Defining the project by features in Client terms makes it much easier to prioritise what really matters. The approach taken in FDD is starts with getting the Client to order the features from most important to least important. Then, the Client is asked to say which features couldn't the project be launched without. The first ordering is then ignored and the Project Manager knows what's absolutely necessary as apposed to what's nice to have. It's human nature to want everything but when you are forced to decide what's an absolute necessity, it changes the decision making process. The Project Manager can use this understanding in a number of ways, to help define upfront what to put in Stage 1 and what can wait until later stages or later in the project what can be cut if the scope needs to be reduced should problems arise.

The tangible result that Process 2 delivers is a list of features. From a project management perspective, it delivers a common language and a real understanding of the value of each feature.

PROCESS 3 – PLANNING

An initial project-wide activity to produce the development plan.

This process extends the benefits provided by Process 2. It provides the Project Manager with a way of planning the development phase in a meaningful way for both the client and the programmers. It is done in conjunction with the Development Manager and Chief Programmers.

Ordering

The first level is basic practicalities; some features will depend on others and therefore must be built first. This sounds like commonsense but it's surprising how often on projects a programmer can reach a point where they can't continue because they are waiting for another part of the project to be completed. If one feature is dependant on another, the Project Manager plans for them to be built in the right order. On the other hand, if a set of features are dependant on each other, the can be assigned to a single individual to avoid the risk of one programmer waiting on another to continue working.

The value of the collaborative approach in Process 1 is seen again in Process 3. Having the Development Manager and Chief Programmers involved in this planning process reduces the risk of the Project Manager missing dependencies or issues that they may not be aware of. As the Chief Programmers will ultimately have to deliver the working code, having them as a part of the planning process means that the plan will be better informed and there is less chance of problems arising during the development phase. No-one likes having a schedule dictated to them, programmers are no different. Process 3 allows the people that can add value and who will be most affected by the plan to be involved and have a say. The end result will be a better informed plan, supported by the Chief Programmers as they helped to create it.

Balancing Load

Not all features will be of equal size and complexity. As a part of the development plan, features will be assigned to feature teams or individual programmers. It is important that the plan considers the load on each programmers in terms of ability and appropriateness for the task. Once again, the collaborative approach comes into play. The Project Manager will not always know the full complexity of a task or the abilities of the programmers in particular areas. This is where collaboration with the Development Manager and Chief Programmers will help to create a plan that reflects the abilities of the project team.

Delivering Early Results

Another benefit to the planning phase is the ability for the Project Manager to manage staged releases. Rather than the Client having to wait until the end of the project to see anything, the Project Manager can decide to release a part of the project early on. Understanding the values and priorities of each feature means the Project Manager can decide which features would be best to include in an early release. For example, there might be a number of features than can be completed quickly and easily and in the process, delivering a beta release to the Client early on in the project. This will build confidence and trust with the Client and stand the Project Manager in good stead should problems arise down the track.

PROCESS 4 – DESIGN BY FEATURE

A per-feature activity to produce the feature design package.

Process 4 is broken down into three steps, walkthrough, design and inspection.

In the walkthrough, programmer familiarises themselves with what they are about to build before starting on a detailed design which is then inspected before they can start building. The inspection of the design means defects can be found and removed before a single line of code is written for that feature.

It might seem like commonsense to design and inspect before building but it is often ignored. In many other industries, the idea of building something before it has been fully defined, designed and planned would be considered negligent, yet it happens all the time in software development. The first reaction of many programmers, especially in web development is to open their favourite editor and start coding. The amount of risk that this approach exposes to any project to is enormous. However the converse can also happen where there is an attempt to design everything upfront and can often result in “analysis paralysis”.

Exactly how much design should occur upfront is a hotly debated topic amongst advocates of agile methodologies. The approach taken in FDD is seen in the difference between Process 1 and Process 4. As we have seen, Process 1 includes design early in the lifecycle, but it is not a detailed design. The detail is left to Process 4. Putting the detailed design in Process 4 ensures that it is considered at the right time – before the code is written. It also breaks the design down into meaningful chunks, feature by feature. This means programmers don’t feel like they are spending all of their time designing and no time coding; immediately after the design has been completed and inspected, the programmer can start coding.

The final step of Process 4 is the design inspection. It is another example of how FDD uses collaboration, in this case between the Chief Programmer and programmer, to produce a better result. The discipline of inspection has been proven to produce outstanding results. From a programmer’s perspective, if you know someone is going to look at your design, you’re going to take more care. It also helps to promote a consistent approach rather than each programmer doing things their own way. It also has been proven to be an excellent method for defect removal. (Testing is the Least Effective Defect Removal Techniques, De Luca on FDD, Newsletter #2003 – 5) The idea of this step being to remove as many defects prior to coding which in turn reduces the overall coding and testing effort required in the project.

The tangible result delivered by Process 4 is a design for each feature. From a project management perspective, it uses collaboration between the Chief Programmer and Programmer to ensure detailed design occurs at the right time and enforces the discipline of inspections.

PROCESS 5 - BUILD BY FEATURE

A per-feature activity to produce a completed client-valued function (feature).

Process 5 is also broken down into three steps: code, code inspection, and promote to build. As with Process 4, the idea of collaboration and benefits inspections is enforced. What makes Process 5 unique is the final step “promote to build”.

For code to be “promoted to build” it must be finished. The key to this is the definition of “finished”. A feature is not finished until there is nothing else to be done. The way a Project Manager can test if a feature is truly finished is simply by asking, if the programmer answers “yes”, the Project Manager then asks “so there’s nothing else to be done?”. This often gets a different answer. It’s not that the programmer is being difficult or misleading it’s just that given the chance, many programmers will continue to work on code, tweaking, optimising or trying to improve it. If there’s time to do this then it’s not a problem, but if there’s a tight deadline, the Project Manager needs to focus programmers on getting the project completed and this is a great way to ensure that focus.

The other benefit of this Process is that helps the Project Manager see clearly how much of the project has been completed and how productive each programmer is. According to Gerald Weinberg (Quality Software Management Vol.1), the productivity difference between programmers can as much as 20 to 1. The issue is how to assess the productivity. Even though features might range in size from 2 hours to 2 weeks of work, it doesn’t take long for the Project Manager to assess how much work each programmer is actually producing once the variances in feature size are taken into account.

The tangible result delivered by Process 5 is working code. From the project management perspective it delivers techniques for assessing how much of the project is being completed and the productivity of each programmer. Given this knowledge the Project Manager can then adjust the workload for each programmer to optimise the work rate overall.

CONCLUSION

FDD is lightweight, well defined and targeted at producing results. From a project management perspective, it provides a framework that for managing the big issues that arise in most projects at a

time and in a manner that will have the greatest impact. The approach FDD takes is focused on people and people management as that's the key to successful project management

The key to FDD is in Process 1 and the group approach to modelling which includes the Client, Stakeholders, Domain Experts and Chief Programmers. The Project Manager facilitates rather than dictates the process, the outcome of which is consensus and understanding across the project team. This understanding allows the Project Manager to with scope and Client expectations. The group approach also allows the Project Manager to see how people work and their abilities before creating the project team. The key people issues are dealt with upfront before they become problems.

In Process 2, collaboration is refined to the Chief Programmers. This creates a consistent language and basis for communication throughout the rest of the project.

Process 3 brings together the key people that will deliver the project and gets them all involved in creating a development plan that considers scheduling, complexity, individual abilities and the client's priorities. Once again, the Project Manager facilitates rather than dictates. A plan is of little value if people won't agree to follow it, the collaborative approach helps to ensure the plan is meaningful and the people that fulfill the plan are onboard.

In Process 4, the Chief Programmers and Programmers collaborate on the details of the project, the design of each feature. This discipline produces excellent results in a manageable way.

Process 5 follows on from Process 4 with the collaboration between Chief Programmers and Programmers. What it adds on top is the ability for the Project Manager to see what each Programmer is actually producing and manage the build phase appropriately.

What FDD does is manage people effectively through a well defined framework that gives context to the process of software development. It realises that it is not an exact science and depends on the abilities of the people involved. It harnesses collaboration and clear communication each step of the way to ensure the project has the best chance of success.

Contact Built by designIT | Powered by eZ Publish