Chapter 1
Author: Rick Lukowski and Others
Compiler and Editor: Thomas W. de Boer

1. Freedom Concept Models

Some Approaches to Methodology Construction

As with software, there are different ways to develop a methodology.

The most straightforward way is the Edisonian approach -- try a bunch of different approaches until one is found that works noticeably better than the others, or maybe just works at all. This is where I, Grady Booch, and most others were back in the '80s. Success with the Edisonian approach was very hit and miss -- mostly miss.

A second approach is to find some notations that are useful in software development, then document a process or methodology around developing these notations. This is where Grady Booch is now with UML and RUP. The success of this approach depends on the quality of the underlying notations (or, possibly, how intensely they are marketed!)

A third approach is to find a development solution or 'design pattern' that seems to be sufficiently generic, then specify a process or methodology for building software according to that pattern. This is where both Sun and the Struts community are with processes for building enterprise systems based on the Model-View- Controller (MVC) pattern. Another example are object- oriented methodologies since OO is fundamentally a design pattern. Success of this approach depends on the genericity and suitability of the underlying pattern.

A fourth approach is to define values and principles which arguably produce better software when applied, then build a methodology around those values and principles. This is where the agile community is with methodologies such as XP and SCRUM.

A fifth way is to select a proven engineering model as a basis, then define a software engineering methodology that conforms to that model. This is what Freedom does.

A sixth way is to postulate mathematics-based formal methods for recording requirements, design, and implementation, then build a process and accompanying tool set around those formal methods. The success of this approach depends largely on the understandability of the formal methods to developers. Formal methods for implementation met with good success with the advent of assembly language, followed by spectacular success with Fortran and COBOL. The software research community has been riding this success wave ever since. However, in the areas of requirements and design, the practical success of formal methods has been nil.

Freedom Concept Model

Most methodologies contain elements of several of these approaches. However, one of the approaches will nearly always be dominant in shaping the methodology, as noted in the examples above. Since our goal is to understand Freedom, we will focus on Freedom's

dominant underlying approach -- its concept model and definitions. We will cover its concept model here, and its definitions in the next Post.

The underlying concept model used in the creation of Freedom is the engineering concept of the black box. How to apply black box-based thinking to software engineering was suggested by Harlan Mills in http://csdl.computer.org/comp/mags/co/1988/06/r6023abs.htm (Unfortunately, one has to pay IEEE dues to view this paper online.)

In this paper, Mills proposes extending the black box model into a triple box-structured model consisting of black box, state box, and clear box. Freedom adopts this triple model, with some changes in semantics for the state box. Also, Freedom uses slightly different terminology of black box, gray box, and white box for the three facets of the model.

In a manner fundamentally, if not literally, consistent with Mills, Freedom uses the black box as the requirements model, the gray box as the design model, and the white box as the implementation model.

For more information on Freedom's triple models, including a graphical depiction of each, see:

black box -- http://www.jreality.com/freedom/bb_model.html
gray box -- http://www.jreality.com/freedom/gb_model.html
white box -- http://www.jreality.com/freedom/wb_model.html

The black box model is a familiar and proven concept in many engineering fields including electrical engineering, computer hardware engineering, aerospace engineering, and even behavioral psychology. Mills believed that application of the proven box-based model concept held great promise for software engineering as well. The developers of Freedom concurred when they selected Mills' box-based concept model as the foundation for Freedom.

History Of the Name ‘Freedom’

Freedom was developed from 1988 to 1990 as one of the deliverables to NASA on the Space Station Freedom Project (SSFP). At the time, the methodology was called simply the DRLI 42 (rhymes with ‘girlie 42') methodology, after the number of NASA document in which it was recorded and delivered. When the SSFP was canceled by Congress in favor of the smaller and less expensive International Space Station, much SSFP technology died. That would have included the DRLI 42 methodology except for one of its unique features -- its ability to encapsulate requirements in code objects for ease of change of requirements. Having been one of the authors of the methodology, I recognized this attribute held much promise. Hence, I continued to use and refine the methodology over the next 12 years, mostly on personal projects.

In 1992, the opportunity arose to apply the methodology to software development for the Freedom Ship project (http://www.freedomship.com). A team of five software developers (myself included) was formed, and I commenced to teach them the NASA methodology. After a series of 10 meetings and much email over the better part of a year, they finally agreed the methodology was suitable for use on Freedom Ship. (Note 1)

However, the team did propose the methodology be given a proper name. Due to its inception on Space Station Freedom, and its adoption for use on Freedom Ship, the name ‘Freedom’ was a natural choice.

1. Note: Since the team understood they would be stuck using the methodology if they agreed to it, they subjected it to a VERY intense peer review. The methodology survived the peer review with very little modification even though the team was free to suggest any modifications they wished. They were also free to propose any alternate methodology. No one did.

-- Rick Lutowski

From: PeteCRuth@aol.com
rick@jreality.com writes:
1. Freedom Concept Models

Rick:

So far, so good, and veddy veddy intedesting!

It's also starting to look a little familiar, but that might be due to having had some exposure to Mills a while ago in a research project regarding Chief Programmer Teams. I also attended a couple of forums regarding software development in the 70s and 80s (I think) at which he was the featured speaker.

Thanks.

Regards,

Pete

From Brad Appleton
Rick Lutowski wrote:
> A third approach is to find a development solution or 'design pattern' that seems to be sufficiently generic, then specify a process or methodology for building software according to that pattern. This is where both Sun and the Struts community are with processes for building enterprise systems based on the Model-View- Controller (MVC) pattern. Another example are object- oriented methodologies since OO is fundamentally a design pattern. Success of this approach depends on the genericity and suitability of the underlying pattern.

Actually, that would be more of a ‘pattern language’ rather than a lone/generic pattern. The overarching architectural (generic) pattern that emerges from the whole would be what is more commonly called an architectural ‘style’ (see the work of Garlan&Shaw, Bosch, and Alexander Wolf). MVC and 3-tier architecture are examples of such styles, so is pipes and filters, hub&spoke, etc.

Still absorbing the rest of the post ... --
Brad Appleton

From: Steven Gordon
Rick wrote:
> 1. Freedom Concept Models
Some Approaches to Methodology Construction
As with software, there are different ways to develop a methodology The most straightforward way is the Edisonian approach – try a bunch of different approaches until one is found that works noticeably better than the others, or maybe just works at all. This is where I, Grady Booch, and most others were back in the '80s. Success with the Edisonian approach.....

I object that, with the exception of XP and RUP, the 'approaches' discussed the first section are merely techniques. Some of these techniques address specific programming problems, some address specific design problems, some address specific architectural problems, ... My initial understanding of Freedom is that it addresses the problem of representing requirements (perhaps, I will discover this initial impression is incorrect).

Techniques just form a tool box that allows the master software developer to use the right tool for each job. These techniques generally do not preclude each other. I bristle at the suggestion that any one technique will dominate how we develop software. Even RUP utilizes a lot of other techniques besides UML. It should even be possible to do RUP using a different modeling language than UML.

Nevertheless, I do welcome the chance to add Freedom to my tool box, and look forward to your next installment.

Steven Gordon

From: Rick Lutowski
Alex Jouravlev wrote:
> Sorry, I missed the beginning of the conversation about Freedom Can somebody point me to their website?

Alex,

The site is at http://www.jreality.com/freedom/

> I have been using tables instead of UML with more conservative clients for more then two years by now. Glad to find out I am not alone in that...

Nice to hear!

Thanks for the feedback.

Rick Lutowski

From: Rick Lutowski <rick@jreality.com>
PeteCRuth wrote:
> It's also starting to look a little familiar, but that might be due to having had some exposure to Mills a while ago in a research project regarding Chief Programmer Teams. I also attended a couple of forums regarding software development in the 70s and 80s (I think) at which he was the featured speaker.

Pete,

Lucky you! I never had the opportunity to see Mills in person. Anything about him that sticks in your mind?

Rick Lutowski

From: "Scott W. Ambler"

A few thoughts from the point of view of reviewing this post as if it was an article or chapter.

Rick wrote:
> 1. Freedom Concept Models

Some Approaches to Methodology Construction

As with software, there are different ways to develop amethodology.
The most straightforward way is the Edisonian approach – try a bunch of different approaches until one is found that works noticeably better than the others, or maybe just works at all. This is where I, Grady Booch, and most others were back in the '80s. Success with the Edisonian approach was very hit and miss -- mostly miss.

A second approach is to find some notations that are useful in software development, then document a process or methodology around developing these notations. This is where Grady Booch is now with UML and RUP. The success of this approach depends on the quality of the underlying notations (or, possibly, how intensely they are marketed!)

Why are you dropping Booch's name?

UML and RUP are different things, might want to point that out.

The RUP is about far more than the UML.

> A third approach is to find a development solution or 'design pattern' that seems to be sufficiently generic, then specify a process or methodology for building software according to that pattern. This is where both Sun and the Struts community are with processes for building enterprise systems based on the Model-View- Controller (MVC) pattern. Another example are object- oriented methodologies since OO is fundamentally a design pattern. Success of this approach depends on the genericity and suitability of the underlying pattern.

A fourth approach is to define values and principles which arguably produce better software when applied, then build a methodology around those values and principles. This is where the agile community is with methodologies such as XP and SCRUM.

And Agile Modeling!

> A fifth way is to select a proven engineering model as a basis, then define a software engineering methodology that conforms to that model. This is what Freedom does.

What proven engineering model? Where's the proof?

> A sixth way is to postulate mathematics-based formal methods for recording requirements, design, and implementation, then build a process and accompanying tool set around those formal methods. The success of this approach depends largely on the understandability of the formal methods to developers. Formal methods for implementation met with good success with the advent of assembly language, followed by spectacular success with Fortran and COBOL. The software research community has been riding this success wave ever since. However, in the areas of requirements and design, the practical success of formal methods has been nil.

Freedom Concept Model

It would be good to have an example of what you're talking about. After reading the following text, it's still not clear what you're discussing.

> black box -- http://www.jreality.com/freedom/bb_model.html
gray box -- http://www.jreality.com/freedom/gb_model.html
white box -- http://www.jreality.com/freedom/wb_model.html

Some key points from these URLs would help in this section. Right now the material that you do have here doesn't seem to say much, IMHO.

Scott W. Ambler

From: "Kari Hoijarvi"
PeteCRuth wrote:
> It's also starting to look a little familiar, but that might be due to having had some exposure to Mills a while ago in a research project regarding Chief Programmer Teams. I also attended a couple of forums regarding softwre development in the 70s and 80s (I think) at which he was the featured speaker.

I have a question that you might be able to answer.

What did Mills say about CPT in his late years????

I do not know anybody using chief programmer teams. Simonyi tried it at MS, but it failed.

Mills later created Cleanroom, which is quite different from CPT, equal peers reviewing each other work.

I find it hard to believe in CPT if one of it's most respected proponents dumped it.

Thanks for any comments,
Kari

From: PeteCRuth
rick wrote:
> I never had the opportunity to see Mills in person. Anything about him that sticks in your mind?

Rick,

Excellent presentations. Always wanted to be sure he was being understood. Very personable, and took a lot of time to make sure we were "getting it". Dedicated and intense. Talked about "the software of the future". (That's where I picked up the phrase).

All in all, a pretty down-to-earth guy. He was an outstanding thinker and contributor. Our business is the poorer for his passing.

Regards,

Pete

From: PeteCRuth
hoijarvi wrote:
> What did Mills say about CPT in his late years????

I don't know what he said about it in his later years. In one presentation in Palo Alto, he described it as an evolving concept, but he didn't go into how it might evolve. He semed certain of its immediate benefits, though. Several companies in the Silicon Valley/San Francisco area tried it and were reportedly very successful with it. As I recall, there was a local CPT discusion group that met in Santa Clara on a regular basis; I remember seeing the meeting notices in the Mercury-News (I think).

I was not directly involved with CPT, but I did keep informed, although I don't remember many of the details about how it was implemented. I believe it was an appropriate strategy for its time. There have been several other strategies that were reminiscent of CPT since then, or at least seemed to have borrowed the basic concept and extended it.

I apologize for not being able to recall many of the details, but those were pretty hectic times, full of ferment in software development.

Regards,

Pete

From: PeteCRuth
hoijarvi wrote:
> I find it hard to believe in CPT if one of it's most respected proponents dumped it.

One more thing:
If I didn't mention that Mr. Mills considered CPT one particular step in an evolutionary process, I should have. In a presentation in Palo Alto (I think), he described it that way to a group of about 150 software development "heavy hitters" from quite a few local companies and universities. It was a great presentation, and the "bull session" afterward lasted for several hours.

Those were the days. (But "the good old days" for you young whippersnappers are right now!).

Regards,

Pete

From: Rick Lutowski
Pete wrote:
> All in all, a pretty down-to-earth guy. He was an outstanding thinker and contributor. Our business is the poorer for his passing.

Amen.

Rick Lutowski

From:jasongorman
> Mills later created Cleanroom, which is quite different from CPT, equal peers reviewing each other work.

And I think that's where I recognise most of these concepts from :-) I would suggest folk avail themselves of the oodles of free learning resources out there on Cleanroom, because it explains these core concepts very clearly (and without the big build up!)

Why not try http://www.cleansoft.com/cleansoft_library.html for starters.

Jason Gorman

From: Kari Hoijarvi
PeteCRuth wrote:

> As I recall, there was a local CPT discusion group that met in Santa Clara on a regular basis; I remember seeing the meeting notices in the Mercury-News (I think).

Thats' good to know. CPT has a logic behind it. The problem that Simonyi faced at MS was, that everyone wanted to be the chief programmer.

> I was not directly involved with CPT, but I did keep informed, although I don't remember many of the details about how it was implemented. I believe it was an appropriate strategy for its time.

Probably, during early seventies almost everything was batch so structured decomposition was logical: feed some tapes in, run your algorithm, get some tapes out.

Thanks, Kari

From: Kari Hoijarvi
Oodles of resources?

Cleansoft library is clearly the best free resource that I have found. But he hasn't added anything to it recently. One article 2001, previous from 1997.

the mailing list is very quiet.

Google returns links to pages that are never updated.

I know one user with good results at http://www.sitoni.fi/zero-defect/

if anyone is using Cleanroom or CPT, I wish they would be more vocal about it.

Kari

From: PeteCRuth
Kari Hoijarvi writes:
> CPT has a logic behind it. The problem that Simonyi faced at MS was, that everyone wanted to be the chief programmer.

I am reminded of what someone contributed to this thread on one occasion when I brought up the "80-20" rule regarding software developers: their comment was that everyone believes they're in the top 20%.

Regarding Simonyi at Microsoft, as I recall, he is now off doing his own thing, with Big Bill's blessing, supposedly looking for a new way to build the "software of the future". Heck, welcome to the club, as I'm sure we've all been doing that since came into this exciting and dynamic and innovative and ceative, and sometimes fustrating field. It will be interesting to see what he comes up with.

I believe someone recently brought up the Cleanroom development process in this thread, suggesting that it was similar to the process Rick brought in with the Freedom posts. This was another gem from Harlan Mills' mind. I was involved in a somewhat similar endeavor at Memorex in the 1970s that predated his exposition of the Cleanroom process, but which shared some of the salient features. I'm looking forward to hearing more about Freedom, as the Cleanroom stuff was very interesting.

Of course, my memory isn't quite what it once was, so some of the preceding comments may be juxtaposed. If I've mixed things up, or attributed anything inappropriately, chalk it up to too many "all nighters" in my prime: no offense.

Regards,

Pete

From: PaulOldfield
Pete C Ruth wrote:
> Regarding Simonyi at Microsoft, as I recall, he is now off doing his own thing, with Big Bill's blessing, supposedly looking for a new way to build the 'software of the future'.

Somebody told me Microsoft was 'going Agile', and quoted their inclusions of books by Ron Jeffries, Will Stott and Ken Schwaber in their book series as evidence.

(BTW, I might be quiet for a while; just starting some new work on one of those sites that doesn't seem to like giving folk Internet and e-mail access... Ho hum. It pays.)

Paul Oldfield.

From: PeteCRuth
PaulOldfield wrote:
> Somebody told me Microsoft was "going Agile", and quoted their inclusions of books by Ron Jeffries, Will Stott and Ken Schwaber in their book series as evidence.

I had heard rumors to that effect recently as well.

As a curious coincidence, a friend of mine called a few minutes ago to ask me if I knew anything about this "agile stuff". Seems he has friends at the Microsoft campus in Mountain View (Silicon Valley area of northern California) who are looking for anything they can get regarding agile methodologies. I gave him a capsule sketch of what agility means, vis-s-vis software development, and he said he's pass it along, as well as my e-mails, phone numbers, and the agilemodeling sites.

I wonder what's up with Gates and company. We'll have to wait and see, I guess.

Good luck with the new work: Kick some butt, Big Guy!

Regards,

Pete

From: J. B. Rainsberger
PeteCRuth@aol.com wrote:
> In a message dated 4/3/2004 8:37:35 AM Pacific Standard Time,
PaulOldfield1@compuserve.com writes:

>> Somebody told me Microsoft was 'going Agile', and quoted their inclusions of books by Ron Jeffries, Will Stott and Ken Schwaber in their book series as evidence.

> I had heard rumors to that effect recently as well.

If their presentation at XP/Agile Universe 2003 is any indication, they have a long way to go to be Agile. As you might expect, they did a few Agile things, twisted some other Agile things to fit their environment, and tried to call it Agile. I'll give them credit for trying, but many of us thought they were far too far off to start calling themselves such.

Of course, we don't know how much of our opinion was based on collective self-defence against the Evil Empire. Some, I'm sure. --

J. B. Rainsberger,

From: Ian Chamberlain
PeteCRuth wrote:
> I wonder what's up with Gates and company. We'll have to wait and see, I guess.

I have just received David West's Object Thinking and James Newkirk's Test Driven Development in Microsoft.Net. They are both part of a new "Microsoft Professional" series. The others are Extreme Programming Adventure's in C# by Ron Jeffries, Agile Project Management with SCRUM by Ken Schwaber, Threat Modelling by Frank Swiderski and last, but by no means least a new second edition of Steve McConnell's Code Complete. I almost bought the lot but decided to get them two by two!!

According to Microsoft the series "will represent a unique collection of topics centered on Microsoft .NET architecture, agile programming methods, and design concepts and patterns." If you are interested the link is
http://www.microsoft.com/mspress/findabook/list/series_PB.asp

Whether this means Microsoft will actually go down the agile route is a different question. You still have to do MSF to get the MCSD certification, and I wouldn't call that agile in the least. As far as I know MSF is still the internal standard, although it is not used on all MS internal projects.

Regards

Ian Chamberlain

From: PeteCRuth
J. B. Rainsberger writes:
> I'll give them credit for trying, but many of us thought they were far too far off to start calling themselves such. Trying to turn a behemoth like Microsoft "on a dime" is pretty difficult. If they are even talking about agile, though, it could portend a move in that direction, or it could simply be their attempt to put their particular "spin" on everything in the industry that might really turn into "the next big thing". If it works out, they can take some credit, and it it doesn't, they have an easy out. It does seem to me that the agile movement is beginning to stir some interest at highr levels than ever before, so the rumor mill might not be too far wrong this time. The question is, in he event that Microsoft is, in fact, "going agile", what form will it take? How can they make it appear as though they have been embracing agility, or worse, invented it, and how will that affect the work that has been done so far?

Whenever Microsoft-watchers report that something is afoot inRedmond, I am reminded of the "old days" when IBM ruled the computing world. Whenever some other company brought out a product that might pose a challenge to one of their products, IBM would make some announcement regarding some new feature in "the next release" of their product. This had the effect of "freezing" the market as both existing and potential customers waited for "the other show to drop". Two of the most common attributes of those times that have stood the test of time are:

FUD Factor: IBMs strategy of introducing Fear, Uncertainty and Doubt into a particular market through "strategic product announcements".

"No one ever got fired for buying Big Blue", a reference to IBMs dominant position in all things "computorial".

Mirosoft had a good role model. Looks like things are "deja vu, all over again".

Regards,

Pete

From: Rick Lutowski
PeteCRuth wrote:
> I believe someone recently brought up the Cleanroom development process in this thread, suggesting that it was similar to the process Rick brought in with the Freedom posts. This was another gem from Harlan Mills' mind.

I suspect Cleanroom was brought up as additional info relating to Harlan Mills, since Freedom draws heavily on Mills. Since the current Freedom discussion is focused on requirements, until now I have not made any statements regarding Freedom's approach to testing. If one looks at the Implem & Test part of the Freedom website, one can find some evidence that indicates Freedom may be using Cleanroom -- for example, separation of

test and development teams. However, the fact is that Freedom does not explicitly adopt Cleanroom as its testing approach, although it does not preclude it either.

The premise of Cleanroom -- that code should be written to be defect free, and tests should merely be used to measure quality -- is laudable. The problem is that this is much easier said than done. I only know of two approaches for producing defect-free code: intense manual review/inspection of code, and/or use of formal methods for requirements and design. Cleanroom utilizes both of these, but both have their problems.

Manual reviews, while certainly useful, are themselves human process and hence subject to error. Manual reviews do not eliminate the need for testing.

Formal methods for requirements and design can eliminate many errors associated with writing, inspection, and testing of code. However, formal methods have never 'caught on' with the development community due to their rather esoteric nature. Even if one or more did eventually come into widespread use, they would only eliminate 'accidental' errors but not errors of the 'essence,' to use Brook's terminology.

In short, Cleanroom and its techniques are no panacea, as admitted by Cleanroom advocates. This is not to say its techniques do not have value; they do and, especially for projects dealing with life-critical code, are arguably the best approach to software quality we know of today, The cost-benefit of using it should be seriously investigated by those types of projects. (Note 1)

Rather than advocating a specific testing approach, Freedom supports a number of Implementation & Test principles including Separation of Test and Implementation roles (also a Cleanroom principle), Full Testing (not Cleanroom, but not a priori inconsistent), and conformance of both tests and implementation to Design specs. These can be used by themselves as a minimal test methodology, or as a foundation for more sophisticated approaches such as Cleanroom.

Both the overriding strength and achilles heel of Freedom's native test approach is not mentioned on the website, namely, the existence of a toolset that enables automatic generation of application code declarations AND matching test code from design specs. Only via automatic generation of code, both tester and testee code, from specs can the conformance of code to specs be ensured **at practicable cost**. Indeed, the economics of writing tests has always been the problem with the Full Testing principle. Freedom solves this problem via automation of code generation from design specs, an approach which was proven effective in practice back when I was using Freedom with Fortran. Unfortunately, the generators have yet to be converted to support more modern implementation languages such as Java. In the area of specialized tool support, Freedom is 100% incomplete. (Note: entrepaneural minds will see this as an opportunity rather than a defect!)

Note 1: While I have no hard evidence to support it, I suspect that the Star Wars effort of the 80's was one of the driving forces behind Cleanroom and its approach of zero-defect software without reliance on testing. It is a fact that Parnas gained notoriety in the DoD community for a well-thought out paper concluding that the concept was infeasible due to testing considerations -- not a message his DoD employers wanted to hear! It is also a fact that Parnas and Mills at one point worked together. The applicability of the Cleanroom approach to solve

the inherent testing problems Parnas identified with Star Wars would hardly have escaped notice when these two minds came together. The main point of speculation is the exact causality relationship between Star Wars and Cleanroom.

Rick Lutowski

From: Kari Hoijarvi
PeteCRuth wrote:
> Regarding Simonyi at Microsoft, as I recall, he is now off doing his own thing, with Big Bill's blessing, supposedly looking for a new way to build the "software of the future". Heck, welcome to the club, as I'm sure we've all been doing that since came into this exciting and dynamic and innovative and ceative, and sometimes fustrating field. It will be interesting to see what he comes up with.

http://intentsoft.com

Intentional programming. I saw their demo in 1997 and I was very impressed. Probably they ran into some dead end, since Czarnecki told me that by 1999 their reduction engine worked completely different way that what he described in his thesis.

Now they have Kiczales at intentsoft, that gives me hope, since AspectJ is very impressive. But their website seems to be hibernating, which makes me worried.

Kari

From: Kari Hoijarvi
Rick Lutowski wrote:
>About formal methods:
Even if one or more did eventually come into widespread use, they would only eliminate 'accidental' errors but not errors of the 'essence,' to use Brook's terminology.

While this is true, formal methods don't guarantee a working product, it is often used as a bad excuse. I have heard many times that formal methods are useless because they don't prove that requirements are right, but nobody ever said that requirements gathering is useless because they don't prove the code correct. What's the difference?

I have to take a look at this freedom thing, although at first sight the ‘white box’ term annoys me. Mills used ‘clear box’ since white as opaque as black.

Kari

From: Kari Hoijarvi
PaulOldfield wrote:
> Somebody told me Microsoft was ‘going Agile’, and quoted their inclusions of books by Ron Jeffries, Will Stott and Ken Schwaber in their book series as evidence.

I was in the Outlook/Exchange group 1995-1998.

In my opinion, MS was already then very agile, comparing to its size. The process we used, internal version of MSF, is pretty well described in Steve Maguire's 'Debugging Development Process'.

Kari

From: Rick Lutowski
Kari Hoijarvi wrote:
> I have to take a look at this freedom thing, although at first sight the 'white box' term annoys me. Mills used 'clear box' since white as opaque as black.

Call it clear box if you wish -- the web site uses both, and I prefer clear box myself. However, white box predates clear box, and is more consistent with a 'color' naming convention for the box models. In reality, there is no semantic difference between 'white' and 'clear' in this context.

Rick Lutowski

From: Steven Gordon
Kari Hoijarvi wrote:
> While this is true, formal methods don't guarantee a working product, it is often used as a bad excuse. I have heard many times that formal methods are useless because they don't prove that requirements are right, but nobody ever said that requirements gathering is useless because they don't prove the code correct.
What's the difference?

I do not see why formal methods would eliminate accidental errors. After all, the formalisms are being composed by people, and thus can have accidental mistakes just like code.

How does any formal method prevent people from either representing an incorrect requirement or representing a correct requirement incorrectly? Feedback from the people who will ultimately use the software is ultimately the only way to discover these errors. One of the major underpinnings of agility is that increasing the amount and frequency of feedback is the surest way to increase quality and decrease rework.

Steven Gordon

From: Rick Lutowski
Steven Gordon wrote:
> I do not see why formal methods would eliminate accidental errors. After all, the formalisms are being composed by people, and thus can have accidental mistakes just like code.

They do not eliminate them, but they do reduce them system-wide by an amount roughly equal to the G level times ten. This is my own SWAG based on two assumptions:

1. Each G level represents a factor of 10 LOC leverage
2. Programmer will make roughly the same number of mistakes per LOC at each G level.

The G levels are:

1G -- machine code
2G -- assembly language
3G -- Fortran, COBOL, ... Java
4G -- formal methods, domain-specific languages (e.g., SQL)

In a nutshell, my thinking is that formal methods reduce the total LOC count to record a given amount of info, leading to a reduction in the number of 'accident' errors.

Everyone is invited to disagree with my assumptions in favor of their own assumptions, and come up with their own estimate for 'accident' reduction due to formal methods.

Turns out, this is not just an academic issue to Freedom. While premature in the sequence of Freedom posts, I will say now that the notation used for recording required response behavior is one aspect of Freedom that can likely be improved significantly. Formal methods may be one way to do this.

One BIG problem is that use of any requirements formal method would result in a severe negative impact on the ability of the 'average' developer to use Freedom. There are other problems with formal methods as well, including major economic problems. Still, they are always there on the radar horizon.

> How does any formal method prevent people from either representing an incorrect requirement

Not 'accident' errors. Defined as 'essence' errors by Brooks (who stole the terms from Aristotle, with appropriate credit.)

> or representing a correct requirement incorrectly?

If a good requirement goes bad, it is a bad requirement.
Essence error.

> Feedback from the people who will ultimately use the software is ultimately the only way to discover these errors.

Completely agree. Moreover, Freedom completely agrees as well. Hence, Freedom provides mechanisms for obtaining this feedback as early as possible in the development process. Much sooner than any other methodology of which I am aware.

Thanks for the questions!

Rick Lutowski

From: Kari Hoijarvi
Steven Gordon wrote:
> I do not see why formal methods would eliminate accidental errors. After all, the formalisms are being composed by people, and thus can have accidental mistakes just like code.

As Dijkstra put it: it's a separate concern.

Kari