The First Agile Slugout:
eXtreme Programming versus Freedom

by
Rick Lutowski
rick@jreality.com

Version 2003−0929

Contents

Acknowledgments

Thanks are extended to the following individuals for their assistance in reviewing this material:

Cristof Falk
Derek Lane

Preface

The Alliance and the Werewolf

One one hand, the recent Agile software movement is good. The Agile Alliance offers a central forum for discussion of software methodology, a subject which for a time fell into near−disrepute. Thanks to the Alliance, software methodology is now reacquiring a measure of respect.

On the other hand, the framers of the Agile movement have taken an almost revolutionary stance with respect to methodology. The Agile Manifesto and Principles appear to have been drafted with the most eXtreme (*sic*) of methodologies in mind. At the same time, some Principles are so narrowly defined as to be inapplicable to larger software projects, which are most in need of methodological solutions. Given that the methodologies of the 1980's failed to slay Brooks' werewolf, the hope appears to be that within the newly−stated principles may lurk a silver bullet, or at least a solid leg trap. Regardless of their shortcomings, the Principles do offer a fresh perspective on the problem. Certainly, the very act of an industry group formally proposing methodological principles is without precedent, and is a step forward on the track of the werewolf.

Three decades of in−the−trenches software development has caused a rather profound change in my own perspective as well. I now believe that the problems to which the Agile Alliance seek solutions, in fact, have solutions. I no longer believe those who thoughtlessly quote Brooks by wailing "there is no silver bullet." Software developers around the world now slay the old werewolf daily using Internet−based open source software development, a silver bullet unimagined by Brooks[1] in the mid−1980's. Ironically, precious few are tanning the hide, choosing instead to dance with a live werewolf and inevitably be bitten yet again and again. It appears a phenomenon that Brooks could not have anticipated has occurred: a silver bullet has been found but, due to its being so different from traditional practice, is being ignored by the mainstream. Do we really expect silver bullets to fit the mental mold of leaden ones? Illogically, many apparently do. In any case, the question is not "is there a silver bullet," but "where is number two?" After all, if there is one silver bullet there may well be two.

[1] Those who doubt that open source development is a silver bullet are encouraged to re−read Brooks to ascertain his definition of "silver bullet." (http://www−inst.eecs.berkeley.edu/~maratb/readings/NoSilverBullet.html)

Where Is Number Two?

Although there is not yet a second silver bullet, its mold may exist. This mold, like many of its kind, has two halves. These two halves were forged in the research crucibles of Dr. David Parnas and Dr. Harlan Mills over 20 years ago. It remained only for someone to fit the halves together. The crucial integration was accomplished over 10 years ago on the NASA Space Station Freedom Project, and the Parnas−Mills integration technique has been undergoing slow but steady refinement ever since. In the last two years its evolved form was documented in the context of the Freedom Ship project, and the more streamlined shape granted a name: Freedom.

That Freedom may be the mold for a second silver bullet was hinted at by the newly−stated Agile Principles. To explore this in more depth, a comparison was carried out between Freedom and the most pervasive current Agile methodology, XP, using the Principles themselves as the comparison criteria. This work documents that comparison. In the process, it also suggests improvements to those Principles which appear to have been too narrowly formulated.

The results of the comparison are, to me, surprising. They cast Freedom in a new light relative to the expanding stable of Agile methodologies. If Freedom is the mold for a second silver bullet, one or more of the new Agile methodologies may be the mother lode of silver for the bullet itself. But even that combination will not be enough. Contrary to standard Agile dogma which discounts tools in favor of people, high−powered, methodology−specific automation will be essential to fire the resulting bullets at sufficiently agile speeds.

With two bullets in the magazine instead of one, enough projects may be able to slay the werewolf to finally secure the software development landscape. As taught in basic training, it is more effective to double tap the target. That is never more true than when the target is a werewolf.

Why the Metaphor?

Rigorous comparisons of agile methodologies are, so far, rare. Although one might expect such comparisons to be a major formal activity of the Agile Alliance, only one comparison paper appears to exist in the Agile Library. That paper, "Agile software development methods, Review and analysis" by Abrahamsson, Salo, Ronkainea, and Warsta, is a comprehensive review and comparison of 10 agile methodologies. Interestingly, their comparison does not make direct use of the 12 Principles that officially define "Agile." However, the review and comparison is quite systematic, which inevitably leads to rather dry and tedious reading. The considerable value of the content is thus diluted by the mind numbing process of encountering a seemingly endless stream of facts. One is inclined to jump to the Conclusion and skip the details.

Unfortunately, in comparisons especially, the meat is in the details.

This work attempts to avoid these two shortcomings of the "Review and analysis" paper. First, the 12 Agile principles are directly used as the methodology comparison criteria. Second, a rather zany metaphor is used as a narrative backdrop for recording the comparison in an effort to encourage actually reading the details. That this is an experiment in technical topic treatment is undeniable. Its effectiveness, even its suitability, is uncertain. However, the worst one can do is simply stop reading it, which is no ground lost compared to the dry traditional approach.

A shortcoming of this work compared to "Review and analysis" is scope. Whereas the latter compares 10 methodologies, this work compares only two, one of which is not even recognized by the industry. However, the comparison conclusions are quantitative with respect to agility, offer insight into the relative strengths of the compared methodologies and, perhaps most importantly, indicate that maximum agility may be best achieved via interplay among two or more methodologies in addition to interplay among people.

Pre−Game

Agile

Reviewing the Rules

Howard: Welcome sports fans! This is Peter, Howard and Don, the PHD team, bringing you the first methodology slugout under the new Agile rules. Don, tell us about the new rules. What is Agile?

Don: The new rules come from the Agile Alliance, an organization of individuals united by a common goal of "uncovering better ways of developing software." Their definition of "better" is summarized in a Manifesto of four items, supported by 12 specific Principles. The four items of the Manifesto for Agile Software Development are (http://www.agilemanifesto.org/):

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck
Mike Beadle
Air van Biennium
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Greening
Jim Highsmith
Andrew Hunt
Ron Jeffry's
Jon Kern
Brian Marci
Robert C. Martin
Steve Mellow
Ken Stewier
Jeff Sutherland
Dave Thomas

Howard: That sounds like a pretty big change from all those thick documents that used to govern software development, at least on paper. Remember how everyone always used to ignore them? Maybe by replacing thick document−based methodologies with simpler rules and principles, folks will be more inclined to pay attention to them. Don, tell us more about the 12 Agile Principles.

Don: The 12 Principles (http://www.agilemanifesto.org/ principles.html) can be organized by the Manifesto items they most directly support. When organized by the Manifesto groups which they most closely represent, the Principles are:

Individuals and interactions over processes and tools

> Business people and developers must work
> together daily throughout the project.
>
> Build projects around motivated individuals.
> Give them the environment and support they need,
> and trust them to get the job done.
>
> Agile processes promote sustainable development.
> The sponsors, developers, and users should be able
> to maintain a constant pace indefinitely.

The best architectures, requirements, and designs
emerge from self−organizing teams.

At regular intervals, the team reflects on how
to become more effective, then tunes and adjusts
its behavior accordingly.

Working software over comprehensive documentation

Deliver working software frequently, from a
couple of weeks to a couple of months, with a
preference to the shorter time scale.

Our highest priority is to satisfy the customer
through early and continuous delivery
of valuable software.

Working software is the primary measure of progress.

Continuous attention to technical excellence
and good design enhances agility.

Simplicity−−the art of maximizing the amount
of work not done−−is essential.

Customer collaboration over contract negotiation

The most efficient and effective method of
conveying information to and within a development
team is face−to−face conversation.

Responding to change over following a plan

Welcome changing requirements, even late in
development. Agile processes harness change for
the customer’s competitive advantage.

Howard: Those Principles seem to make a lot of sense, Don. OK, can you tell us a little about the contenders?

Don: Sure, Howard. In the near corner we have eXtreme Programming, popularly known as XP, the current lightweight champion. Facing him today is Freedom, a 12 year veteran of the space program, trimmed down to compete in the lightweight category.

Howard: Under the new rules, victory will go to the most Agile. Which will that be −− the current industry champ XP, or the veteran challenger Freedom?

XP

The Current Champ

Don: Maybe we can tell more about each contender's chances in this slugout by taking a look at their bios. Here's what the official eXtreme Programming web site says about the current champ (http://www.extremeprogramming.org/what.html):

> (Permission has not yet been granted to reproduce the "What Is XP" web page content. here. Please see the above web address for an overview of XP.)

Don: What the official bio doesn't say is that eXtreme Programming has been criticized as being incapable of scaling to really big jobs in government and large corporations. What do you think, Howard?

Howard: Well, it is a lightweight. Some of its moves are pretty nimble, though, and should be effective on any size job. Certainly communication and courage are important qualities as the job gets bigger. Of course, bigger jobs are inherently more complex, so simplicity may be hard to achieve on really large and involved projects. What do we have on the challenger, Don?

Freedom

The Challenger

Don: We really don't know much about Freedom, Howard. Heard he's spent the past 12 years on private projects. Rumor is, he has a 100% success record. Not sure it I believe that or not! Here's what the Freedom bio on the web says (http://www.jreality.com/freedom/index.html):

> Freedom is a software development methodology originally developed for the Space Station Freedom Program, hence the name "Freedom". Since its inception in 1990, Freedom has been proven effective in use on internal projects for organizations such as NASA, Freedom Ship International, Object Access, and JReality. Authorization for public release of the methodology was granted by NASA in April of 2003.
>
> The goal of Freedom is to reduce the life cycle cost of software by leveraging information−hiding as defined by Dr. David Parnas and the Software Cost Reduction Project. Most notably, Freedom achieves the previously unrealized goal of encapsulation of software requirements in objects to enhance ease of change of requirements.
>
> In achieving this goal, Freedom draws heavily upon the work of the late Dr. Harlan Mills to precisely define

requirements as an aid to their encapsulation. The precise definition of requirements has many benefits beyond requirements encapsulation such as design neutrality of requirements, elimination of requirements traceability as an issue, creation of reusable requirements components at the code level, and Quality Requirements that guide infusion of quality into software in a systematic and consistent manner.

Freedom achieves its software cost reduction and quality enhancement goals via technical solutions, particularly an emphasis on software interfaces. Freedom is an interface−centric technical development methodology, not a software management methodology. Because Freedom is management−neutral, it can be used with many different risk management life cycle models including spiral, incremental, evolutionary, early release, frequent release, prototyping, and other management models to reduce risk in a manner appropriate to the project and customer.

Being interface−centric leads naturally to Freedom being customer−centric as well. The Freedom process specifies that the external user interface code be written first, and a User Interface (UI) Prototype be delivered to the customer extremely early in the development process. The UI Prototype helps the customer verify the correctness and completeness of the requirements, which are themselves defined in an interface−centric manner consistent with the work of Mills and Parnas. Freedom’s Quality Requirements further enhance its customer−centric nature by servicing the customer’s needs for software quality (i.e., "service−oriented") at each step in the development process.

In summary, Freedom produces lower cost, higher quality software by being interface−centric −− applies information−hiding to requirements,

customer−centric −− involves the customer in specifying and verifying requirements,

service−oriented −− continuously services customer needs for quality at each development step,

management−neutral −− pairs with most any software management methodology or process.

The net result of using Freedom is higher quality software delivered to the customer sooner, and software that is less costly to enhance when adding or modifying required capabilities.

Howard: Parnas and Mills −− sounds like Freedom has been honed by some of the best. I suppose adaptability to diverse management styles is what you'd expect, coming out of the space program. But encapsulation of requirements in objects? Elimination of traceability? Systematic infusion of quality? Come on! Not even a champion heavyweight can do all that. Sounds like typical contender bravado to me, Don.

Don: I'll have to agree with you there, Howard. This Freedom challenger sounds more like smoke than sinew. If so, XP is going to lay him flat in short order. Well, the ref is motioning the contenders into the ring. The slugout is about to begin!

Agile Slugout

Round 1

Businessman−developer cooperation

Howard: We're ready for round 1. The winner of round 1 will be decided by best conformance to the first of the 12 Agile Principles, Businessman−developer cooperation:

> Business people and developers must work
> together daily throughout the project.

Don: Here they come, Howard, and XP is coming out swinging! XP is throwing its "Daily Stand Up Meeting" punch. The stand up meeting is when the entire team gets together briefly every morning to discuss the latest issues. The meeting includes the business people assigned to the project.

Howard: Look at Freedom, he's backing off! Freedom's coach is going to the ref. What's he telling the ref?   Maybe Peter, our ringside reporter, can tell us what's going on.

Peter: This is Peter at ringside. Hang on, Howard, while I get closer to the conversation between the ref and Freedom's coach... Wow! Get this. The coach is petitioning the ref to nullify this round because Freedom is strictly a technical methodology, and getting different types of people working together is a management−related principle. The coach is claiming this principle is addressed by whatever management methodology is selected for use in concert with Freedom, but Freedom has not been assigned a management methodology partner here today. So the round should not count.

Howard: Unbelievable! Thanks, Peter.

Don: I knew it, Howard. Freedom is all smoke. XP is beckoning Freedom to get in the middle and fight.

Howard: The ref is agreeing with XP, and is telling Freedom to get in there. There he goes. Ouch!

Don: XP is really beating up Freedom on this one, Howard. Look, Freedom isn't even trying to throw a punch. I guess he really doesn't have much in the way of management moves. Freedom is deferring the whole issue to XP. Oww!

Howard: There's the bell. End of round 1. Freedom is going to the ref again. He's claiming that if this were a fair fight, he'd be paired with a specialized management methodology which would have knocked XP down on this principle, so the decision should go to Freedom. Sounds like there could be some logic to that, Don.

Don: Maybe so, Howard, but the ref isn't buying it. The ruling is that this match is XP against Freedom straight up. Here's the signal −− round 1 to XP!

Howard: I'm wondering what the outcome would have been had this been a tag team match. If Freedom is really trained to operate as the technical member of a technical−management methodology pair, the combination of two specialized methodologies could be really formidable. How would XP fare in a methodology tag team match?

Don: Well, Howard, XP is a stand−alone methodology. It has both technical and management moves. It may resent having to share its corner with another methodology. To do so would be to admit either its technical or management jabs could be inferior to something else. Most of these methodologies are pretty pugnacious and don't like to share turf, Howard.

Howard: But apparently not Freedom. He looked absolutely wimpy in there against XP’s Stand Up Meeting punches. I’ll bet Freedom would have welcomed the help of a partner methodology in fostering technical and business team communication. One thing’s for certain; these two contenders are certainly different!

Don: Right! But in this straight up bout, the differences have so far been to XP’s advantage. Ok, I think we are about ready for round 2.

Round 2

Developer trust

Howard: Start of round 2. Round 2 will be decided by best support for Agile Principle 2, Developer trust:

> Build projects around motivated individuals.
> Give them the environment and support they need,
> and trust them to get the job done.

Don: They're back at it again! XP is throwing lots of quick jabbing management punches, but this time Freedom is fighting back. Looks like Freedom is rearing back for a big technical punch. Wham!..square in the jaw with a powerful chop I've never seen before. XP is reeling from that one! XP has recovered a bit and is starting to throw those quick management punches again. Howard, tell us about those XP moves while I investigate that thunderbolt Freedom just threw.

Howard: Ok, Don. It seems XP is relying on combinations of Moving People Around, Release and Iteration Planning, and even a few Fix XP When It Breaks jabs to show that it gives developers the flexibility to get the job done without rigidity of management or planning getting in the way. But Freedom is just rolling with those punches, basically accepting them, while sneering back at XP. Peter is down at ringside. Peter, can you hear what Freedom is saying?

Peter: Yes, Howard. Freedom is taunting XP, saying that XP isn't even hitting the target. He’s chiding that these XP moves are all management control related and just show that XP doesn’t really trust developers to get the job done without lots of management controls. Also, the fact that the XP Map, which is XP’s process chart, has to be followed rigorously is more proof of management control that gets in the developer’s way. Even Fix XP When It Breaks is a controlling process that has to be followed to change the process. Freedom is really waving the red flags at XP, Howard, like a matador baiting a bull.

Howard: Oops! XP responded to all that with a Pair Programming punch. Freedom just rolled with that, too. All these management jabs don’t seem to be fazing Freedom on this principle. Don, what did you find out about that big Freedom punch... better tell us now because I think he’s getting ready to throw it again!

Don: Here’s the scoop, Howard. Freedom is relying on a nearly−forgotten jab exhibited by Parnas in a 1986 paper called "A Rational Design Process: How and Why To Fake It". (http://www.cs.umd.edu/class/spring2003/cmsc838p/Design/FAKE−IT.doc) Parnas claimed that rational processes are impossible to follow rigorously, so do not try. Rather, developers should be free to do whatever they need in order to produce the desired work products. The only caveat is the products they produce must be the same as would have been produced had it been possible to follow the rational process rigorously. The idea is that rational processes do not exist to control people, but to more clearly define the nature of the work products expected.

Howard: Interesting! A purely technical approach to what most of us, including XP, assume is a management problem. No wonder Freedom is just shrugging off all those XP management punches. Now let me see if I understand this −− Freedom is using it’s own process chart as a technical definition mechanism, not a behavior control mechanism. Does that mean developers are free to ignore the Freedom process charts?

Don: Just so long as the work products they create are the same as what following the process rigorously would have produced. Developers can perform process steps in any sequence they desire, repeat steps in order to experiment, or even invent their own steps or processes to produce the needed work products. This only applies to the Freedom technical process, though. The companion management process still has to be followed rigorously because management processes *are* intended to control people. But, as we already saw, Freedom is not a management methodology, so has no management processes with which to control anyone.

Howard: Wham! Freedom just plastered XP with that punch again! XP is dazed and moving into its corner. Just in time, too, because there is the bell ending round 2.

Don: Look! Now XP is petitioning the ref. Peter, what’s he saying?

Peter: This is Peter at ringside. XP is saying that Freedom shouldn’t be allowed to do non−standard things like apply technical semantics to process charts to solve a management problem! Almost sounds like the same sort of ploy Freedom tried in round 1.

Howard: Yeah, and it’s working about as well. The ref isn’t buying this one either. He’s saying that since Freedom has no people control rules, it wins the "Developer trust" principle hands down. The Parnas "Fake it" philosophy clinches the case since Freedom’s process charts also do not imply lack of trust in developers but, if anything, further empowers the developers. Round 2 to Freedom. We now have a tie match −− XP 1, Freedom 1.

Don: Of course, we also know that Freedom doesn’t usually operate alone, Howard. Normally, there is going to be that management methodology lurking in the background constraining the developers....

Howard: True, but remember −− this is a one−on−one bout, Don. XP versus Freedom only.

Don: Right.

Howard: You know, Don, XP could also apply the Parnas "Fake It" approach to its own process charts. That would help it stand up to Freedom on this "Developer trust" principle.

Don: It’s not quite that easy, Howard. Remember XP is a combination management−technical methodology. Look at the XP Map and you’ll see most of the processes are management related; they deal with things like planning sessions, stand up meetings, etc. Management charts are useless if they are not followed rigorously because their only purpose is to control people. It may, in theory, be possible to apply the "Fake It" approach to the technical part of the XP Map, but XP process charts interleave technical and management tasks so tightly as to make segregating them very difficult. It would require a big revision to XP, Howard. It’s highly doubtful if XP’s coaches would even consider it.

Howard: Then it sounds like Freedom is going to continue to clobber XP on the "Developer trust" principle every time they meet.

Don: It would seem so, Howard.

Round 3

Sustainable development

Howard: We are ready for the start of round 3. Round 3 will pit XP and Freedom against each other for best support of Agile Principle 3, Sustainable development:

> Agile processes promote sustainable development.
> The sponsors, developers, and users should be able
> to maintain a constant pace indefinitely.

Don: The two contenders are approaching the center of the ring for the third time. Not surprisingly, XP is limbering up to throw its "No Overtime" punch. Meanwhile, Freedom is extending his glove to...

Howard: I don’t believe it!

Don: ...to shake hands!! Howard, Freedom wants to shake hands with XP on this principle!

Howard: Ooof! XP let Freedom have right between the eyes! Doesn’t look like XP wants to shake hands, Don.

Don: I wouldn’t think so, Howard. XP is really ticked after loosing round 2... "cheated out" of round 2 is what Peter overhead at ringside. XP is out for revenge, Howard.

Howard: XP isn’t content with just "No Overtime" punches, either. Now he’s starting to mix in some "Iteration Planning" jabs as well.

Freedom has gone into defensive mode. Why isn’t Freedom hitting back?

Don: Well, Howard, Sustainable development is another management principle...

Howard: Oh! Yeah. Looks like Freedom is in for another beating, Don.

Don: And he’s getting it too, Howard. Under the circumstances, he’s taking it pretty well, but the bruises are mounting. Freedom is looking back at his corner for his management methodology partner, but of course there isn’t one there for him today.

Howard: There’s the bell. End of round 3. No question about who won this round, Don.

Don: Look! Freedom is offering to shake hands with XP on the "Sustainable development" principle again, Howard. XP is just ignoring him.

Howard: There’s the official signal from the ref. The score is now −− XP 2, Freedom 1.

Round 4

Self−organizing teams

Howard: Round 4 will be decided by which contender best supports Agile Principle 4, Self−organizing teams:

> The best architectures, requirements, and designs emerge from self−organizing teams.

Don: Hey, look! Freedom’s coach is talking to the ref. We’ll switch to ringside and see if Peter can tell us what’s going on. Peter...?

Peter: That’s right, Freedom’s coach is in conference with the ref. Looks like a pretty heated discussion... Now the conference is over, and Freedom’s coach doesn’t appear to be very happy. Coach! coach! Can you tell us what you were just talking to the ref about?

Freedom Coach: Yeah. This 4th agile principle is flawed. Everyone wants the best architectures, requirements, and designs, but self−organizing teams by themselves are insufficient to produce them. Those teams also need clear technical guidance on what constitutes good architectures, good requirements, and good designs. Without that guidance, or with poor technical guidance, the best teams in the world will just produce crap. Heck, today XP and his followers don’t even know what a requirement is! How can they, or anyone, be expected to produce the "best" requirements if the methodology doesn’t give them a decent clue as to what requirements are?! I was asking the ref to change the principle to read

> The best architectures, requirements, and designs
> emerge from rationally−defined products developed
> by self−organizing teams.

Even acknowledging that self−organizing teams might be useful, giving the teams clear, rational technical guidance on what to produce has to come first. Principle 4 doesn't say anything about that. It's an incomplete principle, flawed.

Peter: You lost me coach. Sure XP tells developers what to produce. For requirements, XP has User Stories, about 3 sentences that describe a customer need in enough detail to cost and plan it. And User Stories are not limited to describing the user interface, either.

Freedom Coach: That's a great example of my point, for three reasons. First, it is highly unlikely that one can plan anything accurately based on three sentences!   Second, the Parnas information−hiding team stated that "Prose is the sign of an error" in a requirements document, so XP's use of sentences is fundamentally erroneous. Everyone knows that prose is ambiguous, prone to incompleteness, and unverifiable. Third, and most important of all, requirements are "the black box view of the software system," by definition. Harlan Mills' work clearly indicates this. What is a "black box view?" Mills proposed that a black box view consists of the stimulus−response behavior of the software. Also, the external interface that detects the stimuli and returns the responses is part of the black box view. That's the sole extent of a black box view, which is synonymous with requirements. To blither on about anything else under the auspices of requirements just shows XP provides poor technical guidance, which in turn leads to substandard, if not totally erroneous, work products. There is no way the "Self−organizing teams" principle is going to fix that and somehow create the "best" requirements if the technical foundation is defective.

Peter: Wow! That’s pretty heavy stuff coach. What did the ref say?

Freedom Coach: He said that even if that’s all true, he is powerless to change the Agile Principles. The wimp! Excuse me, the round is about to start.

Don: Thanks for that report, Peter. Yes, the contenders are moving back into the ring. XP is swinging with "Move people around". Freedom is throwing the Parnas "Fake it" thrust again. Are either of those the same as "Self−organizing teams"? Both opponents look pretty weak this time.

Howard: Could be a problem for both of them, Don. Now XP is dancing around with the "Fix XP When It Breaks" dodge.

Don: Freedom keeps going into a stance like he wants to flatten XP with a "Definition of requirements" blow, but the coach must have gotten word to him that it won’t do any good with the principle worded the way it is. It would just be hitting below the belt.

Howard: Right, Don. If the principle were changed to read the way the Freedom coach suggested, Freedom’s "Definition" punches could be devastating. But with the principle stating that only self−organizing teams create the best work products, Freedom’s technical prowess doesn’t count. It’s pretty clear now why Freedom’s coach conferred with the ref.

Don: Both sides are lunging and dodging, but no one is connecting.

Howard: XP doesn’t seem to give the developers independent control over their organization. XP is pretty tightly managed.

Don: And Freedom doesn’t address management issues at all. The "Fake it" punches are not landing home as they hit on process initiative rather than team organization.

Howard: There’s the bell ending round 4.

Don: Wonder who the ref will say won this one? Pretty poor showing for both sides.

Howard: Here’s the call −− a draw! No winner on principle 4!! The score is still XP 2, Freedom 1.

Don: I think the ref called that one right, Howard. Neither supports self−organizing teams well, but for the opposite reasons. XP is overly management driven, and Freedom not enough.

Howard: Right, Don. But if the principle were ever restated to emphasize the real goal of high quality architectures, requirements, and designs rather than an assumption about how best to attain them, then Freedom would probably knock XP flat on this principle.

Don: If wishes were horses, Howard.

Round 5

Team tuning

Howard: Since no one took any blows on round 4, the contenders are roaring to go for round 5. Round 5 will see which contender best supports Agile Principle 5, Team tuning:

> At regular intervals, the team reflects on how
> to become more effective, then tunes and adjusts
> its behavior accordingly.

Don: Oh, oh. This is another management principle. I'll bet Freedom is in for another drubbing, Howard.

Howard: They're both circling each other in the middle of the ring. There goes XP. He's throwing the "Fix XP when it Breaks" punches, and this time they are connecting. That's XP's way of improving the teams behavior, alright.

Don: But look! Freedom isn't taking it laying down this time. He's fighting back with "Process metric checklists!" Those aren't in any of our info sheets about Freedom. Peter is right on top of it though. What have you got, Peter?

Peter: I'm here with Freedom's coach. Coach, what are Process metric checklists?

Freedom Coach: Process metric checklists are a very simple way of capturing quantitative metrics for process improvement. Every time a developer starts work on a new work product, be it a requirement, design, code module, documentation or other work product, he makes a note of the start date. When he finishes the work product, he notes the completion date next to the start date. If he modifies the work product, he notes the start and stop dates of the modification. Later on, the lists can be analyzed to determine how long each individual product, or groups of products, took to develop. The data can be used to determine which parts of the development process may be in need of more automation or other improvement. They can also be used to determine individual developer productivity, perhaps signal that a developer needs more training in some part of the process, or whatever. A good CM tool could easily automate this sort of record keeping, making the whole thing quite transparent to the development team, although I'm not aware of a CM tool that fully supports the concept at this time.

Peter: I see. But where do the checklists come in?

Freedom Coach: The list of work products with start and stop dates *are* the checklists. We call them that because they look a bit like checklists, except with actual dates instead of check marks next to the work product entries. It may not be the best name, but that's what Freedom calls them.

Peter: Ok. Thanks coach. Back to you, Don.

Don: Freedom seems to be getting a few licks into XP with the checklists, but is taking some "Fix XP When It Breaks" hits in return.

Howard: XP sees it can't get a big advantage on Freedom with the Fix It jab, so he's trying a different move. There's a "Move people around" punch! That's not as strong a way of adjusting the process as Fix It, but XP is looking for any edge it can get.

Don: Freedom is responding with another move of it's own. Hah! Freedom is trying to use "Separation of technical and management methodologies" to counter XP's move. Clever! Turn an apparent weakness into a strength. Since management methods are likely to change independently of technical ones, separate the two concerns to ease the impact of the change!

Howard: Trouble is, Don, that doesn’t directly address the principle of adjusting team behavior. It’s Freedom’s management methodology partner, and not Freedom itself, that has to change to address the principle. And that partner is not represented here today. XP is getting through Freedom’s defense, Don. As you say, it was a clever move on Freedom’s part, but the specific nature of principle 5 negated its effectiveness.

Don: There’s the bell ending round 5. Both contenders took some lumps that time! Looks like Freedom took a few more, though.

Howard: The ref agrees, Don. He’s signaling the round goes to XP on a technicality. The score is now XP 3, Freedom 1.

Don: Freedom needed that round, Howard. He didn’t get it, and now XP is really starting to pull ahead.

Howard: That’s right, Don. But I’m starting to like Freedom’s style. That separation of methodology concerns thing looks like it might have some real potential. It didn’t work out just now in this head−to−head, but give Freedom a good management methodology partner, and he could walk away with the tag−team crown, as well as clobber actual jobs. Any ideas who might make a good management methodology partner for Freedom?

Don: Well, I can tell you it isn’t likely to be XP! XP would have to give up most of its technical aspects, like User Stories and CRC Cards, to get along with Freedom. I don’t ever see that happening, Howard. XP isn’t interested in getting along with Freedom. XP is the champ, and he’s got a 3−1 lead. XP is out to pulverize Freedom by the new Agile rules, and he’s well on his way to doing it!

Round 6

Frequent releases

Howard: Ok, we're ready to start round 6. Round 6 will be decided by best support for Agile Principle 6, Frequent releases:

> Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale.

Don: And there they go! Freedom is getting in first this time. He knows he's behind, and is trying to take the initiative to catch up. He's letting XP have it with Freedom's "Prioritization of Requirements" punch. Direct hit to XP's midsection.

Howard: XP recovered quickly, and is coming right back with "Iteration Planning" punches. They're connecting on Freedom. XP just drew some blood.

Don: Freedom's connecting, too. XP is bleeding. Well, we promised you a slugout, and now it looks like we've got one! Freedom is really going after XP. XP is backing up, trying to get some room.

Howard: Now XP is starting to mix it up with "Make Frequent Small Releases". These are connecting, too. Now Freedom is starting to back up.

Don: Freedom's "Prioritization of requirements" can handle short or long release periods, depending on what the customer wants. So Freedom is still holding his own.

Howard: But XP’s "frequent small releases" addresses the Agile Principle more explicitly, Don. That may not make any difference, though. If you meet the principle, you meet the principle, and both of them are able to do it. Looks like a bloody standoff to me.

Don: Finally, there’s the bell. Round 6 is over, and both of these pugnacious pugilists are headed back to their corners to catch their breath.

Howard: The ref is ready to announce the call.

Ref: Round 6 to XP due to "Make frequent small releases" more directly addressing the "Frequent releases" principle.

Don: Ouch! That really hurts Freedom.

Howard: You got that right, Don. Score is now XP 4, Freedom 1.

Don: Look over at ringside. Freedom’s coach is talking to Peter. We’re switching over to pick up on it...

Freedom Coach: ....principle assumes a priori that all customers want and need frequent releases. This is a false assumption. For example, government agencies which operate under complex and rigid contracting rules may not be able to accept software developed using a frequent release model. Even if this is not in the customer’s best interest, these customers are nonetheless bound to contracting laws. Until such time as the law grants them the flexibility of conforming to Agile, Agile must adapt to these customers. Freedom’s flexibility with respect to life cycle model and long or short release schedules better meets the underlying goal of this principle, which is to meet actual customer release needs. This is yet another poorly stated Agile Principle. Freedom therefore files a formal protest on the "Frequent releases" principle, and on the ruling for round 6!

Howard: Is he ticked, or what!

Don: He made a good point, though, Howard. Some of the Agile Principles do seem a bit presumptuous with regard to customer needs. Methodologies that can meet a wide variety of customer needs appear inferior under the Principles to less flexible methodologies that are specific to the assumptions of the Agile Principles.

Howard: Be that as it may, Don, this bout is head−to−head based on the current Agile Principles. Those are the rules, and the ref has made it clear he’s not changing them. Only the Agile community can change them.

Don: I agree, Howard, but it sure is making things bleak for Freedom.

Round 7

Early and continuous delivery

Howard: We're almost ready for round 7. Round 7 will determine best support for Agile Principle 7, Early and continuous delivery:

> Our highest priority is to satisfy the customer
> through early and continuous delivery
> of valuable software.

Don: That sounds like "Frequent releases" all over again, Howard. This may just be a repeat of bloody round 6.

Howard: One difference, Don, the emphasis on "early" delivery could change things a bit. But I agree with you. There is a lot of overlap in principles 6 and 7, so the result may well be the same.

Don: The contenders are sauntering into the ring again. Freedom is winding up for another "Prioritization of requirements" punch, as you'd expect. Hey, what's XP doing?

Howard: He's got his arms up motioning for a hold. Looks like XP might have a problem!   The ref is right there telling Freedom to hold off. XP is saying something to the ref and Freedom. Maybe Peter at ringside can clue us in as to what's going on.

Peter: Yes, I can, Howard. XP is formally asking Freedom to concede!

Howard: I guess we shouldn’t be too surprised in light of the score. Thanks, Peter.

Don: Wow! Everyone just saw Freedom’s answer −− a massive "Prioritization of requirements" blow to XP’s jaw! XP is momentarily stunned.

Howard: Now he’s recovered, and is coming back with those "Iteration Planning" and "Make frequent release" punches just like before. Looks like a repeat of round 6, like we expected. They’re both beating up on each other, each trying to prove he’s better at "continuous releases."

Don: Yup. Looks like "frequent releases" all over again.

Howard: Oow! Maybe not, Don. Freedom just did a fast 360 spin and nailed XP with a "UI prototyping" hammer blow! XP flew backward onto the ropes. Freedom is continuing to pound him with both "Prioritization of requirements" and "UI prototyping", but it’s the "UI prototyping" hits that are keeping XP from recovering.

Don: Smart move, Howard. User Interface prototyping is about the quickest way there is to get code into the the customer’s hands early on. A UI Prototype is just a facade for the purpose of having the customer verify that the requirements are correct and complete, and the interface look and feel is ok. Little or no functionality actually works, but rule 7 doesn’t say it has to. It just says "early release." It’s hard to beat a UI Prototype for an early release.

Howard: Having the customer verify correctness and completeness of the requirements is critical, so UI Prototypes serve an important purpose even if none of the functionality works. Looks like Freedom finally got one over on XP. XP is still on the ropes, taking a beating.

Don: There’s the bell. Round 7 is over. I think we know who won this round.

Howard: No question, Don. There’s the call −− round 7 to Freedom.

The score is now XP 4, Freedom 2.

Howard: Don, can you explain why XP didn’t throw any "Spike solutions" punches to counter Freedom’s "UI prototyping?" Spike solutions are really just prototypes, right?

Don: They are, Howard, but a different kind. Spike solutions evaluate different design or implementation techniques during implementation of a user story. A user story serves as requirements for XP, so by the time XP implements a spike solution the team is well past the requirements stage for the Release. Also, spike solutions are not intended to be delivered to the customer, whereas a UI Prototype is. So spike solutions would not have helped XP in demonstrating support for the "Early and continuous delivery" principle.

Howard: Ok, but couldn’t XP deliver a UI Prototype early also?

Don: A UI Prototype does not match XP’s paradigm. The two methodologies have fundamentally different notions of requirements.

For Freedom, requirements are "the black box view of the system," meaning the software external interface and its stimulus−response behavior. Freedom first delivers a UI Prototype in which little or nothing works but which implements part of all of the user interface. This permits the customer to verify correctness of the human user interface, which remember is at least partially synonymous with requirements to Freedom. The UI Prototype can be delivered and evaluated in increments if the UI is extensive, which gets UI code to the customer even faster. Once the user concurs the requirements are ok, Freedom delivers functionality in increments of "stimulus sets", which are collocated groups of stimuli in the user interface. So a UI Prototype follows naturally from Freedom’s interface−based approach, and also serves as a code framework for implementation and delivery of functionality.

For XP, requirements are User Stories. XP implements and delivers the system in increments of User Stories, which are process−centric, not interface−centric. A UI Prototype may not match any XP User Story. Also, a UI Prototype implements no actual functionality. whereas a User Story always has some associated functionality. UI Prototypes simply do not match XP's approach, so XP does not use them.

Howard: Thanks for clarifying that, Don. These two contenders are really different!

Round 8

Software progress

Howard: We are almost ready for round 8. Round 8 will determine which methodology best supports Agile Principle 8, Software progress:

> Working software is the primary measure of progress.

Don: Peter has Freedom’s coach on the mike again. Let’s switch over to hear what he has to say.

Freedom Coach: This principle is a backlash against document driven methodologies, which have had their share of problems in the past. While Freedom concurs that working code is a better measure of progress, we would also like to point out that this principle fails to take certain realities into account. Freedom considers producing code without accompanying requirements, designs, test cases, and user manuals to be a sign of lack of appropriate progress, and even a project out of control. Thus, this is another poorly framed Agile Principle. For the record, we suggest it be changed to state

> Working software, with accompanying customer−approved requirements, designs, tests, and user documentation,
> is the primary measure of progress.

That being said, Freedom will still win this slugout in spite of the dubious wording of this and other Agile Principles.

Peter: Ok, coach. Back to you, Don.

Don: The Freedom team clearly thinks the Agile deck is stacked against them. Given that, and their being down 4−2 this late in the match, it’s pretty bold of him to think that Freedom can still pull this out.

Howard: It’s not over until it’s over, Don. The contenders are back in the ring. Round 8 has started.

Don: They are both coming out swinging. Freedom is throwing its "Process Metric Checklists" punch again.

Howard: XP is countering with "Project Velocity". Don, can you explain Project Velocity?

Don: Sure, Howard. Project Velocity is sum of the time estimates for all the User Stories to be implemented in a given XP iteration. It’s the estimated time to implement the code for an iteration, which is a collection of User Stories.

Howard: And we have seen that Freedom checklists are the actual start and completion dates of various work products including, but not limited to, code.

Don: Right, Howard. And both sides are scoring with these punches, because they both include code as a measure of progress.

Howard: One difference, Don. Freedom’s checklists measure actual time to implement, whereas XP’s Project Velocity is estimated time. I would think that actual time is a better measure than estimated time. Also, the checklists record metrics for "working" software, whereas Project Velocity is just an estimate for planned software, not working software.

Don: On the other hand, Project Velocity measures only code, so code is the "primary" measure for XP, whereas Freedom measures all work products, so code is not a "primary" measurement for

Freedom. I think this is what the Freedom coach was complaining about just before the round started. So the word "primary" in the principle could work against Freedom, even though Freedom’s measurement approach is more comprehensive and the metrics are more accurate.

Howard: Neither is gaining an edge on the other, Don. Their moves seem to be pretty evenly matched in the ring. Looks like this is going to be a judgment call for the ref.

Don: I agree, Howard. There’s the bell ending round 8. XP and Freedom are heading back to their corners to cool down. Looks like the ref is reaching for his Agile Principles book. He knows he’s got a tough call.

Howard: I’ll bet he rules it a draw, Don.

Don: Here goes. The ref is about to announce his decision.

Ref: Because XP views software as the "primary" measure of progress, round 8 goes to XP.

Howard: Look at Freedom! He’s slapping the mat with his towel! Freedom’s coach is beating his fists on the wall with an "I knew it!" look on his face. They saw this coming, Don.

Don: I think so, Howard. They obviously feel that Freedom has a better approach to measurement, but they got shafted anyway.

Howard: Tough call. Score is now XP 5, Freedom 2.

Don: XP is grinning like a Cheshire cat.

Round 9

Technical excellence

Howard: Round 9 coming up. Round 9 will pit XP versus Freedom based on Agile Principle 9, Technical excellence:

> Continuous attention to technical excellence
> and good design enhances agility.

Don: Technical excellence and good design! This is obviously a really important principle. I think we'll see a real slugout to bag this one, Howard!

Howard: Both will clearly want to demonstrate their superiority here. After that round 8 call, I'd expect Freedom to be really boiling to win this "Technical excellence" round. Remember, Freedom is supposed to be a technical methodology. What does Freedom use to promote "good design", Don?

Don: For design, Freedom uses two main notations −− Object Model Tables (OMTs) and Abstract Interface Specifications (AIS). An OMT captures the object model for an object−oriented design. It has basically the same information as a UML Class Diagram except is tabular in nature rather than being a node−arrow diagram. Freedom proponents claim that tables are easier and less expensive to create than node−arrow diagrams, and tables are also highly amenable to consistency and completeness checking. They say these properties make OMTs lighter weight and more effective for use in actual design than UML Class Diagrams. An AIS records the design of the interface for one code module or class. The concept was originally invented by the Parnas information−hiding team, and adopted with some refinements by Freedom. Freedom practitioners claim that AIS's can be used as a basis for tools to automatically generate code, test cases, and even documentation from the AIS design. So far, such tools remain to be written for modern languages such as Java, although there is an instance of AIS auto−generation tools having been written for an earlier language.

Howard: Interesting. And how about XP?

Don: XP uses a notation called CRC Cards for design. CRC stands for Classes, Responsibilities, and Collaborators. CRCs are literally cards −− like 3x5 index cards −− used to record information about each class of an object−oriented design. The technique was invented by Kent Beck & Ward Cunningham. Incidentally, Beck and Cunningham are also the inventors of XP, so CRC cards and XP are pretty tightly bound together.

Howard: I guess we'll be seeing those techniques in this round, which is just about to start.

Don: Yup, here they come. Freedom has a fierce scowl, and XP is almost gloating. Both are remembering round 8, Howard.

Howard: Well, this is a new round, Don.

Don: XP has started by lashing out at Freedom with CRC Cards. Freedom is retaliating with Object Model Tables and Abstract Interface Specs. XP has dropped back momentarily, but now is advancing with Spike Solutions. Freedom has matched that with Partial Functional Simulations, which is a more formal name for the same design prototyping idea. Now Freedom is mixing it up with Canonical Design Architecture punches.

Howard: Canonical... what?

Don: With Freedom, all software designs follow the same high level design architectural pattern. Canonical just means all software designs can follow this pattern. The pattern is based on the different kinds of information−hiding modules, as categorized by Parnas and his information−hiding discovery team.

Howard: Freedom sure is pulling a lot of rabbits out of old hats. Seems to be working, though. XP has nothing to counter it. XP is taking severe architectural hits and is backing off. Now XP is trying to regain ground by muscling in "Coding Standards."

Don: Freedom easily countered that with its "Coding Style Guide", same thing by a slightly different name. XP couldn’t regain any ground with Coding Standards. He’s still having to back down under Freedom’s repertoire of technical punches.

Howard: XP isn’t done yet. Now he’s trying to regroup with "Refactor Mercilessly". That’s XP’s way of saying to continually rewrite the code to improve the design.

Don: Wow! Freedom must have been waiting for that. Freedom responded with an avalanche of "Software Decision Encapsulation" and "Hardware Interface Encapsulation" punches. Encapsulation, basically just another name for information−hiding, means to write the code so that is easy to change. That not only permits refactoring, but anticipates it to make refactoring easier. XP is backed up to the ropes now, Howard.

Howard: XP may be on the ropes, but he’s still got some moves left. Now XP is trying to hide behind "A System Metaphor", an XP concept that helps makes the design easier to understand.

Don: Freedom responded with of "Definition of Design", and "Definition of Implementation", which is central to understanding what information constitutes design, and which constitutes implementation. Metaphor got XP up from sagging on the ropes, but Freedom’s Definitions are keeping XP from gaining any maneuvering room. XP is still cornered in by the ropes.

Howard: Now it looks like Freedom is going into a stance for a completely different punch. He’s delivering it.... Holy cow! A real devastator!! XP trampolined off the ropes and dropped to the mat. He’s still down and groggy. The ref is counting. 1..2..3..4..5.. XP is getting up. Freedom is getting ready to level another one of those devastators. But, wait! There is the bell ending round 9. It seems XP has just been saved by the bell. I can’t believe he would have survived another one of those. What did Freedom hit him with, Don?

Don: Quality requirements, Howard. Quality Requirements are Freedom’s way of systematically and continuously infusing quality into the design.

Howard: Here’s the official call. The round goes to Freedom. No surprise there, after what we just saw. Score is now XP 5, Freedom 3. Tell us more about those Quality Requirements, Don.

Don: Quality Requirements are ’ilities’ like reliability and usability, ranked to reflect the customer’s needs for different types of quality. Freedom uses these ranked ’ilities’ as decision criteria to select among competing design and implementation alternatives at every step in the development process. When used in this way, Quality Requirements directly address the "Continuous attention to technical excellence" part of the "Technical excellence" principle. Quality Requirements are also the heart of what Freedom calls its "service−oriented" approach to development, where service−oriented isn’t narrowly defined as just "web services," but more broadly encompasses the customer’s total needs for quality. It is a very powerful concept, and XP has nothing like it. Freedom showed a lot of poise by holding that punch back until XP was off balance. There was then no way for XP to recover. XP almost went down for the full count.

Howard: XP narrowly dodged a fatal bullet. When it comes to the technical parts of development, Freedom certainly does look impressive. It remains to be seen if that will be sufficient to prevail here today, though, Don.

Don: That's right, Howard. XP still has a 2 point lead, and there are only 3 rounds to go. Freedom has to make a clean sweep from here on out. It's going to be interesting to see if he can do it.

Round 10

Maximizing work not done

Howard: Start of round 10. Round 10 will determine which of our two contenders best supports Agile Principle 10, Maximizing work not done:

> Simplicity−−the art of maximizing the amount
> of work not done−−is essential.

Don: They are both back in the ring again. XP is still a little groggy after that "Technical excellence" drubbing he took in round 9. Freedom is leaning on his momentum and delivering the first punch, a "Definition of Requirements" swing.

Howard: That sent XP reeling, but also seemed to wake him up. XP is coming back with "Do the simplest thing that can possibly work". That punch definitely hits on the principle, but barely fazed Freedom. I don’t get it. That should have really knocked Freedom for a loop, but didn’t. Also, what does "Definition of Requirements" have to do with "maximizing work not done," and why is that so much more effective than XP’s direct Simplicity punch?

Don: Not sure, Howard. Doesn’t make much sense to me either. "Maximizing work not done" is supposed to be one of XP’s strong points, yet XP is definitely taking it on the chin here, too.

Howard: Now XP has gone to his "Never Add Functionality Early" punch. He’s connecting on Freedom, but Freedom seems to be just shrugging these powerful punches off. Now Freedom has come back with "Reusable Requirements" and "Reusable Classes" blows. Those, too, are having more effect on XP than visa versa. By the way, what’s a reusable requirement?

Don: No doubt about it. XP is taking a beating on the "Maximizing work not done" principle. This is a big surprise. I would have thought XP would trounce any other methodology here, but Freedom is drawing all the blood. Not exactly sure what a "Reusable requirement" is, but I can understand why Freedom’s "Reusable" swings are so effective. Software reuse is one of the most effective ways known of "maximizing work not done" by avoiding reinventing the wheel, so to speak. If Freedom can indeed reuse requirements as well as code, that would certainly increase the power of his Reuse punch. But the stunning effectiveness of Freedom’s "Definition of Requirements" punches in "maximizing work not done" is a mystery. I can’t explain it, Howard. Ouch! More gashes are opening on XP.

Howard: There’s the bell ending round 10. The ref is not hesitating with his decision. Round 10 to Freedom. The score is now XP 5, Freedom 4.

Don: I see it, but I still don’t understand it. Peter, do you have any insight as to what just happened down there?

Peter: I have Freedom’s coach here. Coach, can you explain what we just saw? Why is Freedom’s "Definition of Requirements" so effective at "maximizing work not done"?

Freedom Coach: Because the black box definition of requirements means that specifying, designing and implementing the software external interface becomes your requirements process. At some point, the external interface has to be developed, there is no way around it. By recognizing this as requirements, Freedom eliminates the work other methodologies expend on fruitless things like User Stories in the name of "requirements." Not only are these things not requirements, they are not even needed. It is entirely feasible to build software without detailing its usage scenarios, but it is impossible to build software without creating an external interface. The work savings is phenomenal. For example, XP says to create 60−100 User Stories for each Release. This takes many months, yet is totally unnecessary. In the same amount of time that XP spends writing 60−100 prose stories for one Release −− and remember "Prose is the sign of an error" in requirements −− Freedom works out the details of the software interface for the entire application with the customer, implements the code for the interface, and delivers this UI Prototype to the customer so they can verify the correctness and completeness of the interface, which remember are requirements by definition. Meanwhile, XP has yet to write a line of code, and instead has a pile of unnecessary prose. Since XP only "maximizes work not done" in the context of code, they have not yet begun to save anything, while Freedom has just saved months of development time compared to XP. In short, XP addresses the principle tactically, or "in the small" at the lines of code level, while Freedom addresses it strategically, or "in the large" at the requirements definition and process level. Freedom can additionally address the principle "in the small" as XP does, but we choose not to. Freedom believes the XP rules of "Do the simplest thing that can possibly work" and "Never Add Functionality Early" are misinterpreted by literal−minded developers so often that Freedom refuses to recommend these rules.

Peter: Can you clarify why you think those XP rules are misinterpreted?

Freedom Coach: I'd like to, but the next round is about to start. Maybe after the match is over and we've taken XP's crown.

Peter: Ok, coach. Thanks, and good luck! Back to you, Don.

Don: Well, there you have it. Freedom totally changes the rules for requirements via its definitions, thereby categorizing XP's User Stories as "unnecessary!" And he still thinks Freedom is going to pull this out and become the new champ, Howard.

Howard: He certainly seems confident. Two rounds to go and only one point separation. Freedom could do it, Don.

Don: After the last two rounds, It’s tempting to think he just might. But look at the next Agile Principle, Howard. It’s management related.

Round 11

Face−to−face conversation

Howard: Round 11 is about to begin. Round 11 will be decided by best support for Agile Principle 11, Face−to−face conversation:

> The most efficient and effective method of
> conveying information to and within a development
> team is face−to−face conversation.

Don: Management related, Howard. Not good for Freedom.

Howard: You could be right, Don. Ok, here we go. Both contenders have entered the ring.

Don: They are circling each other. XP is opening up with his "Customer Is Always Available" jab.

Howard: Freedom isn’t backing off on this one. He’s responding with "Customer POC."

Don: Of course! Freedom requires that a customer representative, or Point of Contact, participate in development of the requirements. Freedom is fending off XP’s opening punch with an equivalent one of his own.

Howard: XP sees he’s getting nowhere with that one. Now he’s shifted to "Pair Programming" punches. There’s a lot of face−to−face conversation in pair programming.

Don: But Freedom is countering that too, Howard. He’s striking back with "Pair requirements." Freedom’s philosophy is that developing requirements in pairs is even more beneficial than developing code in pairs. One of the pair must be a customer POC, though, or someone knowledgeable about the customer’s functionality needs.

Howard: Freedom is holding his own against XP for this management principle, Don. They are both getting smacks on the other. Nobody is backing down.

Don: Looks pretty evenly matched, Howard. Now XP is shifting again, looking for an edge. He’s gone to his "Daily Stand Up Meeting" punch. There is obviously a lot of face−to−face conversation in meetings.

Howard: Freedom took that blow pretty hard, then glanced into his corner, Don.

Don: I think I know why, Howard. He’s instinctively looking to his management methodology partner for support with things like meetings. Of course, Freedom isn’t paired with a management methodology today. I don’t think Freedom has a defense against XP’s Stand Up Meetings.

Howard: Looks like you’re right, Don. Freedom is taking some heavy hits now, and is backing up.

Don: Freedom is doing his best with his "Customer POC" and "Pair requirements" punches, but it’s not enough. XP’s got him, Howard.

Howard: The bell has just sounded ending round 11.

Don: The ref is about to announce the call. We know what’s it’s going to be.

Howard: Right. Round 11 to XP! Score is now XP 6, Freedom 4. A

2 point spread and one round to go.

Don: That’s the match, Howard. No way Freedom can win. The last round is just a formality.

Howard: That’s too bad. I was starting to like the looks of Freedom. He’s extremely strong and agile technically. But he needs a management methodology partner to balance things out. XP’s strengths are mostly in the management area, and that’s showing here today. XP’s management strengths have outmatched Freedom’s technical advantages.

Don: Sorry to cut in Howard, but Peter has Freedom’s coach aside for another statement. Switching over to you, Peter.

Peter: Thanks, Don. Freedom’s coach is here, and has something to say. Coach?

Freedom Coach: I just want to say for the record that Principle 11 is another flawed Agile premise. Modern communications technologies such as chat rooms, instant messaging, wikis, and video conferencing make physical presence unimportant for many projects. For example, the entire Open Source Software movement is accomplished without any physical presence. Also, there are situations where face−to−face conversation is *not* the best alternative, such as complex or controversial topics which require substantial thought and analysis to properly formulate one’s position. In such cases, remote communication such as email is more effective as it encourages "thought time" before responding, which reduces the risk of brash or ill−conceived pronouncements that often occur in face−to−face conversation. While face−to−face communication is certainly effective in some cases, it is not always the best as the principle states. Principle 11 is just plain false and should be dropped as an Agile Principle.

Peter: Coach, do you really think the Agile community will drop this principle?

Freedom Coach: They will if they are truly striving to improve software based on a rationale assessment of the world instead of stating principles and pronouncements that do not reflect reality. It's up to them.

Peter: Back to you, Don.

Don: Thanks, Peter. I detected sour grapes in the coach's statement, Howard.

Howard: That's to be expected under the circumstances, Don. Actually, I find it hard to read Freedom's mood at this point. He could be very despondent or boiling mad. The next round should show us which.

Round 12

Changing requirements

Howard: We are ready to start the last round, number 12. Round 12 will be decided by best support for Agile Principle 12, Changing requirements:

> Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

Don: The contenders are back in the ring. XP is light on his feet and jaunty. He knows he’s retained his crown even if he looses this round, which of course he will try to avoid at all costs.

Howard: Freedom’s jaw is hard−set and he has a determined, almost confident, look. Not despondent at all. He’s no doubt going to try to make a good showing in this round, knowing that folks often hang on to that last impression the longest.

Don: There they go! XP throws a "Release Planning" punch, XP’s way of adjusting to changing requirements. Freedom ducks out of the way, making XP miss. XP throws his "User Stories" punch. It glances off Freedom as he rolls with it. XP continues to throw his two requirements punches. Freedom backs up toward the ropes, staying at arms length and keeping XP’s punches from connecting. So far Freedom hasn’t thrown any himself. Strange.

Howard: Now Freedom circles back away from the ropes, keeping just far enough from XP to make his punches ineffective. XP is following, trying to get in close enough to really connect. Now XP is between Freedom and the ropes. Freedom stops backing, finally letting XP come a bit closer...

Don: A lighting quick "Definition of Requirements" and "Prioritization of Requirements" 1−2 punch from Freedom! XP bounces off the ropes. A quick 360 spin gains momentum for an "Encapsulation of Requirements" blow that catches XP on the rebound off the ropes. A devastating smash to the head!!

Howard: XP is down on the mat. He appears to be out cold! The ref is starting the count. 1..2..3..4..5.. still no movement from XP. 8..9..10!! XP is out!

Don: It’s a technical knockout for Freedom!!

Howard: Freedom TKO’d him, Don. Incredible! Freedom wins the match on a lightning quick technical knockout.

Don: The ref is signaling a win. Freedom’s arms are up in victory!

Howard: XP’s trainers are out on the mat trying to revive him. He’s still out cold. They’re signaling for the stretcher team. XP is being carried off.

Don: What were you saying about a "good showing in the end" Howard?

Howard: I wasn’t expecting anything like this!! I still hardly believe what we just saw. Freedom’s strength and agility for requirements changes almost defy description!

Post−Game

Freedom Interview

Don: Peter is with Freedom now.

Peter: That was a phenomenal showing in the last round! How did you do it?

Freedom: Requirements are what I do best, man! That's what I was created for, work out constantly for. When it comes to requirements definition, change, and agility −− I'm the best!!

Peter: For a while you were pretty far behind XP. Were you ever worried?

Freedom: Coach and I knew we'd trounce him in the end.

Peter: Ok!! Anything else you'd like to say to anyone?

Freedom: I'd like to thank Dr. David Parnas and the late Dr. Harlan Mills for their great guidance. And all the fine folks who worked on the Space Station Freedom Software Support Environment Project for bringing me up right. And of course, to my coach, who slimmed me down to lightweight class, works me out constantly, and teaches me new moves. Without them I wouldn't be who I am. Oh, and hi, Mom!

Peter: And here's Freedom's coach. Congratulations on a spectacular win!

Freedom Coach: Thank you. It's been a long road getting here, over 12 years. But I think we've shown what Freedom can do, even when judged against the new Agile rules, with all their flaws.

Peter: Freedom certainly shows a lot of technical power and agility. It's clear Freedom's strength is technical, especially in requirements, and that you defer management issues to a partner methodology. Do you have a favorite management methodology that you prefer as a partner?

Freedom Coach: We do prefer to focus on the technical side of software development, and leave the management aspects to a partner methodology. We believe that separation of management and technical concerns has important advantages for a project. No, we have no current favorite management methodology. We are investigating some of the newer ones like Crystal and Lean, but have not reached any conclusions or formed any preferences yet.

Peter: XP showed some pretty good management moves today. Would you consider partnering Freedom with XP, coach?

Freedom Coach: Yes, XP has some great management attributes. We would consider partnering with them if XP agreed to shoulder only the management role, and deferred the technical side to Freedom. But that is up to XP.

Peter: Thanks very much. coach. Again, congratulations on a stunning win!

Freedom Coach: Thank you.

XP Interview

Peter: We were finally able to catch up with XP's coach. We have him here now. How is XP doing? Is he Ok?

XP Coach: XP is just fine.

Peter: That's really good to hear. Coach, this was obviously a disappointing loss. What do you think of Freedom?

XP Coach: Totally unorthodox.

Peter: You have to admit his technical moves were effective today.

XP Coach: We don't admit to any such thing.

Peter: Is XP ready to fight Freedom again to regain the crown?

XP Coach: As far as we're concerned, XP never lost it.

Peter: What about the idea of XP being a management methodology partner with Freedom?

XP Coach: No comment.

Peter: Anything else you care to say, coach?

XP Coach: No. Excuse me, I've got to go.

Peter: This is Peter at ringside. Back to you, Don.

Don: Thanks, Peter. Here’s Howard with the recap.

Recap

Howard: Here is the summary of the action:

Round 1, Businessman−developer cooperation. XP’s Stand Up Meetings cannot be countered by Freedom. Round to XP.

Round 2, Developer trust. Freedom’s "Fake It" principle shows more relevance than XP’s Move People Around, Release and Iteration Planning, or Pair Programming. Round to Freedom.

Round 3, Sustainable development. XP’s No Overtime and Iteration Planning cannot be countered by Freedom. Round to XP.

Round 4, Self−organizing teams, is not addressed well by either methodology. Tie round; no winner.

Round 5, Team tuning, is more directly addressed by XP’s Move People Around and Fix XP When It Breaks than by Freedom’s Metric Checklists and Separation of Technical and Management Methodology Concerns. Round to XP.

Round 6, Frequent releases, is better addressed by XP’s Iteration Planning and Frequent Small Releases than by Freedom’s Requirements Prioritization. Round to XP.

Round 7, Early and continuous delivery, is better handled by Freedom’s Requirements Prioritization and UI prototyping than by XP’s Iteration Planning and Frequent Small Releases.   Round to Freedom.

Round 8, Software progress, is more directly addressed by XP's Project Velocity than by Freedom's Metric Checklists. Round to XP.

Round 9, Technical excellence, is addressed by many artifacts of both methodologies including Freedom's Object Model Tables, Abstract Interface Specs, Partial Functional Simulations, Canonical Design Architecture, Coding Style Guide, Software Decision Encapsulation, Hardware Interface Encapsulation, Definition of Design, Definition of Implementation, and Quality Requirements and XP's CRC Cards, Spike Solutions, Coding Standards, Refactor Mercilessly, and System Metaphor. However, Quality Requirements gives a decisive edge to Freedom. Round to Freedom.

Round 10, Maximizing work not done, is better addressed by Freedom's Definition of Requirements, Reusable Requirements, and Reusable Classes than by XP's Do the Simplest Thing That Will Work and Add No Functionality Early. Round to Freedom.

Round 11, Face−to−face conversation, is better handled by XP's Customer Is Always Available, Pair Programming, and Stand Up Meetings than by Freedom's Customer POC and Pair Requirements. Round to XP.

Round 12, Changing requirements, is far better met by Freedom's Definition of Requirements, Requirements Prioritization, and Requirements Encapsulation than by XP's User Stories and Release Planning. Round and match to Freedom.

Don: Thanks, Howard. The per principle score for each side is shown in the table on your screen.

```
                    Per Principle Score

     Agile Principle                 Type         Winner
  ----------------------------  ------ -------   -------
 1 Businessman-developer cooper mgmt              XP
 2 Developer trust              mgmt              Freedom
 3 Sustainable development      mgmt              XP
 4 Self-organizing teams        mgmt              tie
 5 Team tuning                  mgmt              XP
 6 Frequent releases                   tech       XP
 7 Early, continuous delivery          tech       Freedom
 8 Software progress            mgmt              XP
 9 Technical excellence                tech       Freedom
10 Maximizing work not done            tech       Freedom
11 Face-to-face conversation    mgmt              XP
12 Changing requirements               tech       Freedom
                                ----   ----
                         Total    7  +  5  = 12

                            XP    5  +  1  =  6
                                (71%) (14%) (50%)

                       Freedom    1  +  4  =  5
                                (20%) (80%) (42%)


   Freedom Homepage: http://www.jreality.com/freedom/
   XP Homepage:  http://www.extremeprogramming.org/
```

Don: It is interesting to note that either methodology by itself addresses only about half of the Agile Principles, with Freedom excelling on the technical principles and XP best addressing the management principles. Thus, both methodologies are strong, but in different areas. A near−ideal Agile methodology would combine the strengths of each.

Howard: Good summary, Don, except you forgot about those Agile Principle changes that Freedom recommended. How many were there?

Don: Right, Howard. Freedom went on record recommending changes to four of the 12 Agile Principles.

Howard: Can you summarize those for us?

Don: Sure, Howard. Freedom suggests Principle 4 be changed from

> The best architectures, requirements, and designs
> emerge from self−organizing teams.

to

> The best architectures, requirements, and designs
> emerge from rationally−defined products developed
> by self−organizing teams.

Freedom recommends Principle 6

> Deliver working software frequently, from a
> couple of weeks to a couple of months, with a
> preference to the shorter time scale.

be changed to address "actual customer release needs" rather than "frequent" releases.

They suggested Principle 8 be changed from

> Working software is the primary measure of progress.

to

> Working software, with accompanying customer−approved
> requirements, designs, tests, and user documentation,
> is the primary measure of progress.

Finally, Freedom recommends that Principle 11

> The most efficient and effective method of
> conveying information to and within a development
> team is face−to−face conversation.

be dropped because face−to−face conversation is not always best, so the Principle is "just plain false."

Howard: Thanks, Don. That’s it, sports fans. Thanks for joining Peter, Howard and Don, the PHD team, for the First Agile Slugout.

References

"A Rational Design Process: How and Why to Fake It," D.L. Parnas and P.C. Clements. *IEEE Transactions on Software Engineering*, Vol. SE−12, No. 2, February 1986.

"Agile software development methods, Review and analysis," Pekka Abrahamsson, Outi Salo, Jussi Ronkainea, and Juhani Warsta. VTT Electronics, VTT Publications 478, 2002.

Agile website: http://www.agilemanifesto.org/

Freedom website: http://www.jreality.com/freedom/

"No Silver Bullet −− Essence and Accidents of Software Engineering," Frederick P. Brooks, Jr. *Computer,* Vol. 20, No. 4, April 1987.

XP website: http://www.extremeprogramming.org/

Alphabetical Index