アジャイルプロセス入門　第Ｉ部　テキスト
～　アジャイルプロセスを知る　～

第１版
2011/11/1

一般社団法人
西日本アジャイルプロセス協議会

アジャイルプロセス入門　第Ｉ部
～　アジャイルプロセスを知る　～

一般社団法人
西日本アジャイルプロセス協議会

WJAPC
West Japan Agile Process Consortium

WJAPC
West Japan Agile Process Consortium

目次

第1章　アジャイルとは

アジャイルの意味とは

ここでのお話

- アジルとは
- アジャイルプロセスとは
- システム開発プロセスの歴史
- アジャイルプロセスを考える

WJAPC

アジルとは

- Agileとは「俊敏」「機敏」という意味である。
- アジルと言えば
  - **ひところは、アジル経済、アジル経営**
  - **アジル・コンペティション 「速い経営」**
- 昨今では、Agility(アジリティ)という言葉 「俊敏性」
- アジルからアジャイルへ

WJAPC

アジャイルプロセスとは

- アジャイルプロセスとは、一言でいうとシステムに対する要件の変化や追加を積極的に受け入れ真の要求に見合った価値のある開発を実施するプロセスである。
- アジャイルプロセスとは特定の開発手法を指すものではない。

システム開発プロセスの歴史

1960年代	1970年代	1980年代	1990年代	2000年→

開発プロセス

職人技→建築や製造業を手本とした開発プロセスの実施へ

ウォータフォール

ウォータフォール型プロセス

プロトタイプ・スパイラル

プロトタイプ・スパイラル型プロセス

重量級プロセス

大規模開発に耐えうるRUPを代表とする反復型プロセス→重量級プロセス

軽量級プロセス

インターネットの普及による小規模案件の増加→軽量級プロセス

アジャイルプロセス

アジャイルプロセス

アジャイルプロセスを考える

アジャイルプロセスのマニフェスト

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	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

WJAPC

アジャイルプロセスを考える

「安い、早い、旨い」ではない

- 単に、開発費用が安いということではない
- 単に、開発期間が短いということではない
- 単に、機能が提供されるということではない

WJAPC

アジャイルプロセスを考える

システムに求められるもの

- 情報システムに対する要求は、あらかじめ存在しているものではなく、ビジネス価値にもとづいて「開発」されるべきものである。
- 情報システムは、それ単体ではなく、人間の業務活動と相互作用する一体化した業務プロセスとしてデザインされ、全体でビジネス価値の向上を目的とするべきである。
- 情報システムの存在意義は、ビジネス価値の定義から要求開発を経てシステム開発にいたる目的・手段連鎖の追跡可能性によって説明可能である。
- ビジネス価値を満たす要求は、直接・間接にその価値に関わるステークホルダー間の合意形成を通じてのみ創り出される。
- 要求の開発は、命令統制によらず参加協調による継続的改善プロセスを指向すべきである。
- 「ビジネスをモデルとして可視化する」ということが、合意形成、追跡可能性、説明可能性、および継続的改善にとって、決定的に重要である。

「要求開発アライアンス（http://www.openthology.org）」より抜粋

アジャイルプロセスを考える

ビジネスの変化に対応する

- ビジネスの変化に対応するとは、
  - **利益、売上、コスト**
  - **投入時期、ビジネスのライフサイクル期間**
  - **必要とされる要求、ビジネスの仕掛けの実現**
- つまり、
  - **適正な投資額**
  - **機会損失しない開発期間**
  - **今必要とされる機能で保たれる品質**
- Just in , Right in ・・・ Cost、Time、Quality
- この意味での「安い」、「早い」、「旨い」

アジャイルプロセスを考える

価値

顧客

ビジネス戦略への整合性

投資対効果

IT活用による儲かる仕組み

開発者

コスト

期間

品質・機能

ビジネスの変化に対応できるシステム創り

第2章　開発の型

開発の型(プロセス)は多くある。

WJAPC
West Japan Agile Process Consortium

WJAPC

ここでのお話

- ウォーターフォールの原文を知っていますか
- V字モデル
- W字モデル
- 段階的開発–漸進型と反復型
- プロトタイピィングモデル
- スパイラルモデル

WJAPC
West Japan Agile Process Consortium

ウォーターフォールの原文を知っていますか

- 要求定義→設計→プログラミング→テスト→運用、の順に後戻りなくシステムの開発を進めていくポピュラーなモデル
- しかし、このモデルの元になった論文は、このような開発手法の危険性と改善策を指摘する内容
- 不幸にもこのモデルはすべての軍事ソフトウェアに関する米国防総省の仕様書に「一般的に行われているソフトウェア開発手法」として引用されてしまい、フィードバックや改善策が欠落した状態で世界に広まってしまった
  - 「Managing the Development of Large Software Systems」IEEE WESCON, August 1970, pp.1-9 Winston W.Royce
  - http://portal.acm.org/citation.cfm?id=41801
  - http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf
- 論文中ではウォーターフォールという言葉は登場せず、一般的に行われているソフトウェア開発手法がモデル化され、このモデルがリスキーかつ失敗を招くものであり、どのような結末になるかが述べられている。

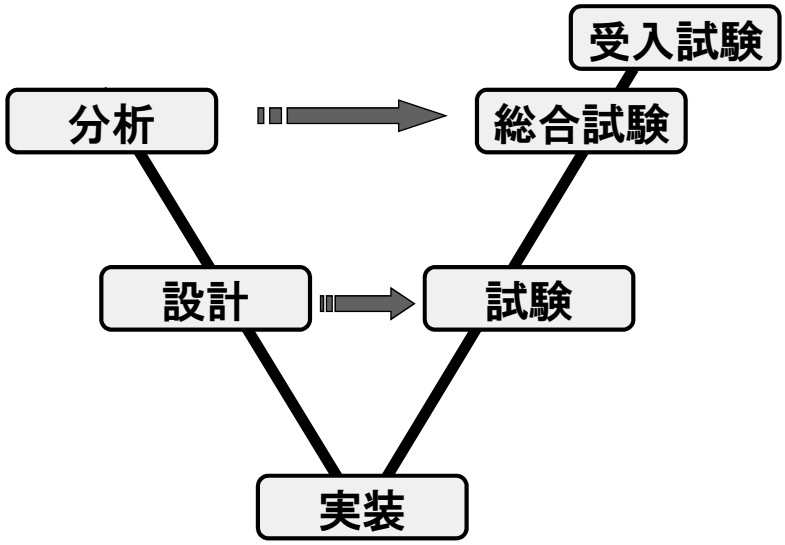
WJAPC
West Japan Agile Process Consortium

ウォーターフォールの原文を知っていますか

- 一般的に行われているソフトウェア開発手法に対して、下流工程を予備的に行って上流工程へ戻るフィードバックループが提唱され、さらに失敗を招くリスクを回避するための5つの改善策が提言されている。
  - PROGRAM DESIGN COMES FIRST
    ソフトウェア要求分析の前に予備的なプログラムデザインを行え
  - DOCUMENT THE DESIGN
    デザインをドキュメントとして残せ
  - DO IT TWICE
    2回繰り返せ
  - PLAN, CONTROL AND MONITOR TESTING
    テストを計画し、制御し、監視せよ
  - INVOLVE THE CUSTOMER
    顧客を巻き込め
- いずれも、それぞれの行程を一回で完了させることが困難であること、そして、計画と早期に顧客のコミットメントを得ることの重要性を示唆している。
- これらの改善策は40年近く前に提唱されたものにも関わらず、現在のアジャイル開発が目指しているものと通じる部分がある。

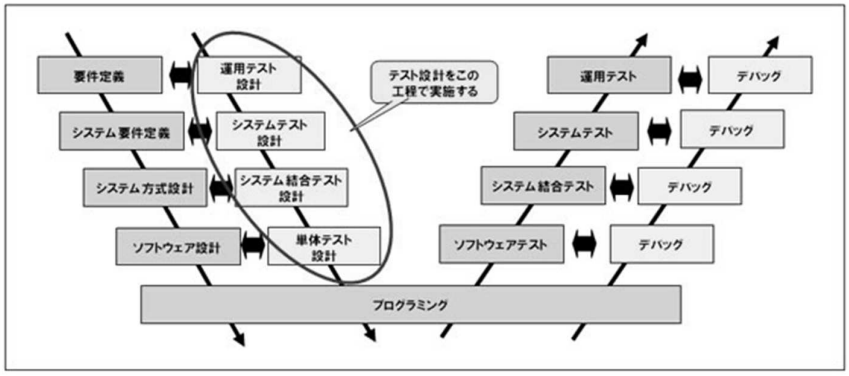
V字モデル

V字形式



V字モデル

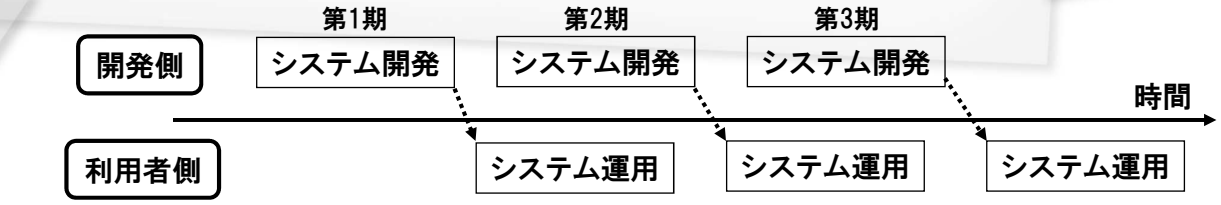
W字モデル

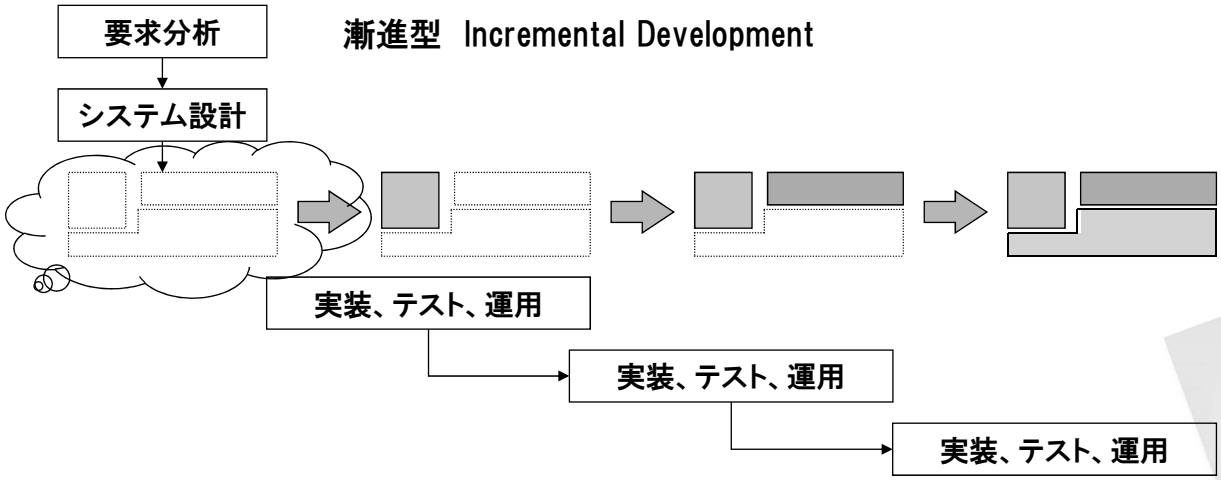


図は、http://journal.mycom.co.jp/photo/column/pm2/009/images/002l.jpgより。

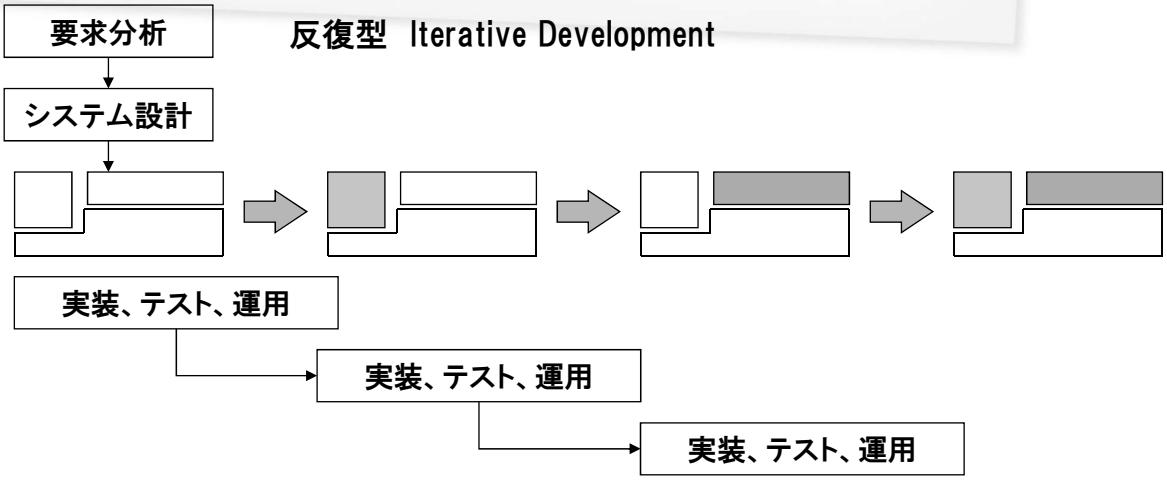
【概要】
従来の、設計を経て実装後にテストを実施する開発プロセスに対して、設計時にそれぞれのフェーズの検証工程のテスト計画を実施するプロセス。
【利点】
上流工程からテスト項目を考えることで、要件の齟齬や漏れに気付くことができる。
要件や設計に対してのトレーサビリティを確保できる。
テストの実施規模が早期に把握できる。

段階的開発−漸進型と反復型



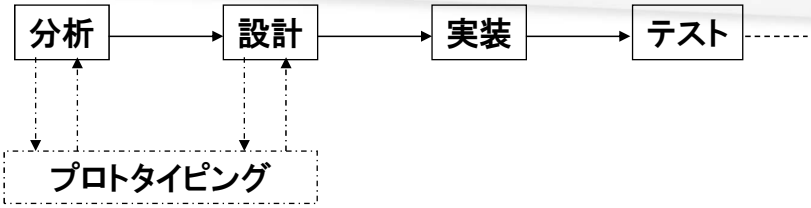


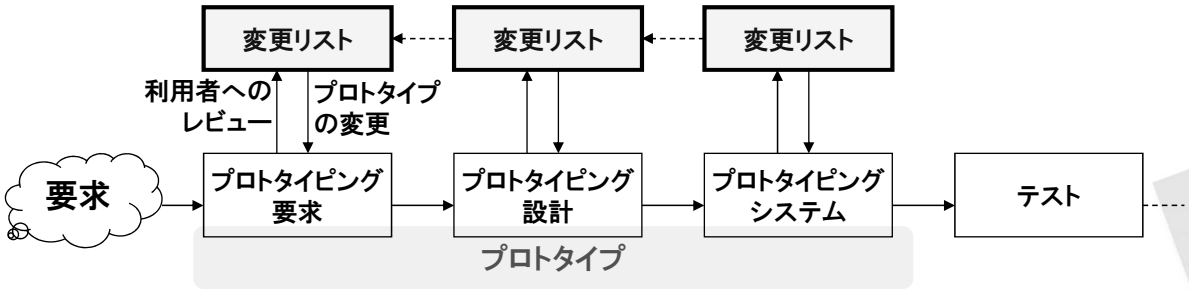
段階的開発−漸進型と反復型



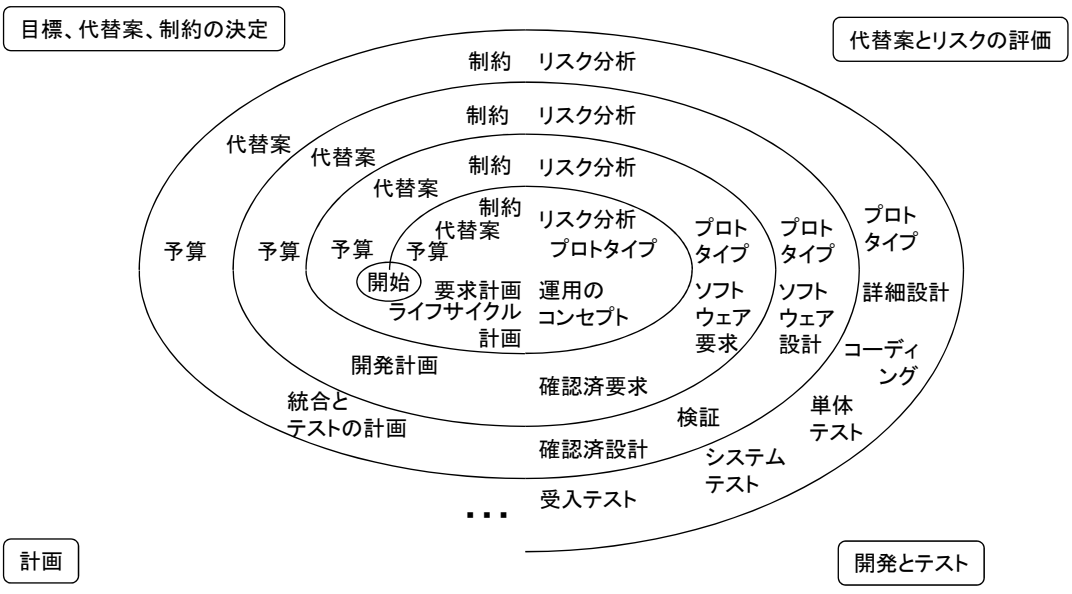
実際の段階的開発は、漸進型と反復型を組み合わせて行う

プロトタイピィングモデル





スパイラルモデル



Boehmのスパイラルモデル

第3章　代表的なアジャイル開発手法

アジャイル開発手法はいろいろある

ここでのお話

- 主なアジャイルプロセスの方法論
- エクストリーム・プログラミング
- スクラム
- リーンソフトウェア開発

主なアジャイルプロセスの方法論

方法論名称	概　要
エクストリーム プログラミング：XP (Extreme Programming)	提唱者：Kent Beck。コーディング、テストファースト、リファクタリング等、技術プロセスが中心。
スクラム：Scrum	提唱者：Ken Schwaber, Jeff Sutherland。マネジメントにフォーカスした方法論。
クリスタル(ファミリー)： Crystal (family)	提唱者：Alistair Cockburn。ワイドスペクトラムな方法(小規模～大規模)。継続的なプロセス改善。
フィーチャ駆動型開発：FDD (Feature Driven Development)	提唱者：Jeff De Luca, Peter Coad。モデル中心の古典的な繰り返し型開発プロセスで、かつ、軽量。
適応的ソフトウェア開発：ASD (Adaptive Software Development)	提唱者：Jim Highsmith。RADを発展させ、カオス適用理論(CAS)を用いたフレームワーク。
動的システム開発方法論：DSDM (Dynamic Systems Development Method)	RAD、JADをベースとして、プロトタイプを多用する。
リーン ソフトウェア開発：LSD (Lean Software Development)	提唱者：Mary Poppendiek。トヨタのカンバン方式(最小在庫＝ドキュメント)の原理応用。
エクストリーム モデリング：xtUML (Executable and Translatable UML)	提唱者：OMG-MDA等。検証実行可能なモデリング(ツール)を利用。マネジメント的側面はない。

エクストリーム・プログラミング:XP

- 繰り返し型開発のひとつでユーザーが要求する機能のなかで、ビジネス価値を生み出す機能から少しずつすばやくリリースを行う手法。
- また、ソフトウェア開発におけるユーザー側及び開発側の不安を互いに認識させ、その互いの不安を解消するための権利と責任を受け入れる環境を作り上げていくプロセス手法でもある。
- 目標：優れたソフトウェアを開発する
- 考え：常に注意を払い、状況に適応し、変更する
- XPを進めるにあたって、各プラクティスを実施することが目的ではなく、各プラクティスが求める理想、アクションを意識し、改善を繰り返していくことが必要。でなければアジャイル開発から乖離してしまう。

エクストリーム・プログラミング:XP

- 共有する価値:
  - コミュニケーション
  - シンプル
  - フィードバック
  - 勇気
  - 信頼
- プラクティス
  - 全員同席
  - ペアプログラミング
  - 常時結合
  - テスト駆動型開発
  - リファクタリング
  - コードの共同所有
  - 継続可能な作業時間
  - 計画ゲーム
  - 短期リリース
- 4つの変数
  - コスト
  - 納期
  - スコープ
  - 品質

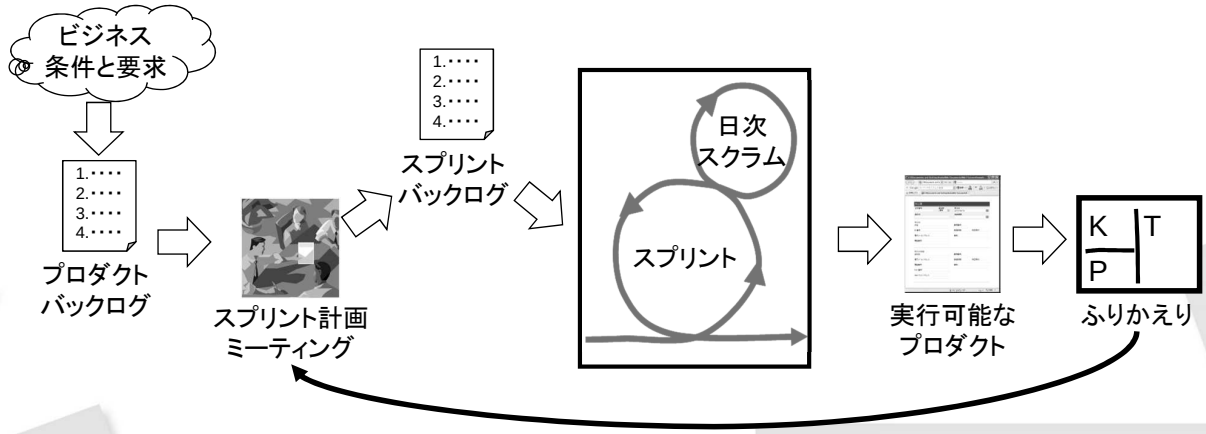
SCRUM

- SCRUMは、ソフトウェア開発のプロセスや技術そのものではなく、採用したいプロセスや技術を取り込む事の出来るソフトウェア開発フレームワークである。
- SCRUMセオリー
  - SCRUMではリスクを予測、コントロール可能にするため反復型開発のアプローチを取り、以下の3つをポイントとしている。

見える化	検証 （ふりかえり）	適応

SCRUM フレームワーク

- ロール
  - スクラム･マスター　　他のメンバーにプロセスを理解させ、実行を支援する
  - プロダクト･オーナー　　リリースに含めるプロダクトバックログを決定する。
  - チーム　　スプリントを通じて、バックログを消化し、ソフトウェアを開発する。
- スプリント
  - 1つのスプリントは1ヶ月未満。
- ソフトウェア以外で作成する物
  - プロダクトバックログとスプリントバックログ。バーンダウンチャート（進捗を見える化できる）。



リーン ソフトウェア開発

- XPやScrumとは異なり、それ自身がソフトウェア開発プロセスやプラクティスを定義しているのではなく、アジャイルプロセスにおける基本的な考え方と、より分かりやすい実践的指針を提供している。

- 書籍などで紹介されるアジャイルプラクティスの多くはコンテキストを考慮しなければ、うまく使うことができないが、リーン ソフトウェア開発は、個々の分野向けの適切なアジャイルプラクティスへと変換するのに役立つ7つの原則と22の思考ツールを提供している。

リーン ソフトウェア開発　7つのリーン原則

- ムダを排除する
  - ここで言う「ムダ」とは顧客が認める価値を、製品に負荷しないもの、すべてのこと。リーン思考では『ムダ』という概念を最大の問題としている。
- 学習効果を高める
  - ソフトウェア開発は複数のロールの異なるメンバーが協調し、複雑なものを作ることが求められるため、学習効果を高めることがソフトウェア開発そのものの改善には必要となる。
- 決定を出来るだけ遅らせる
  - 不確定要素の多い分野では、決定に必要な情報が揃うまで決定を遅らせることが、誤った決定を避けるのに効果的なアプローチとなる。
- 出来るだけ早く提供する
  - 設計、実装、フィードバック、改善のサイクルを早めることは学習効果を高め、また不確実性に対応するためにも効果的である。
- チームに権限を与える
  - 改善のサイクルを早め、不確実性に対応するためにも、現場での意思決定ができるようにすべきである。
- 統一性を作りこむ
  - 優れた製品には見た目や使用感などに統一性が感じられる必要がある。また統一性を産み出すために、ユーザーから開発メンバーに至るまでコンセプトの統一性が必要になる。
- 全体を見る
  - 部分を順番に開発すると局所最適に陥る危険性がある、全体感を見失わないようにすべき。

第4章
アジャイル開発する上で

やはり、技術や知識は必要です

オブジェクト指向は必要だ

- オブジェクト指向とは
  - オブジェクトとは、あなたが関わり合うであろうもの(Thing)……「もの」「こと」「場」
  - オブジェクトは振る舞い、状態、それに識別性を持っているもの
  - オブジェクト指向とは
    実世界の概念構造をそのままソフトウェア構造に取り込むこと
- アジャイル開発プロセスを実践したいなら
  - 知識としては必要
  - 実践があればさらに良い
  - UMLは知っているほうが良い

プログラミング言語は、・・・

- 少なくとも、ひとつのプログラミング言語を極める
  - Java、C#、Smalltalkのようなプログラミング言語
  - C++、C、VB、COBOL、Fortranでも良い
- プログラミング言語を駆使できるということは、
  - アルゴリズムがわかる
  - プログラミングとは何かがわかる
- プログラミングの達人
  - 「オタク」ではなく、現代の職人 → 技術者

最後に

- 技術団体に参加して技術を磨こう
  - 例えば、アジャイル開発なら西日本アジャイルプロセス協議会

子曰、學而時習之、不亦説乎、
有朋自遠方来、不亦楽乎、
人不知而不慍、不亦君子乎

論語より

其心を知て、直なる所を本とし、
實の心を道として、兵法を廣く行ひ、
正しく明かに、大きなる所を思ひ取て、
空を道とし、道を空と見る所也。

五輪書より

http://www.wjapc.jp/

著者
新保康夫、猪原信彦、谷本誠、
前野公孝、山根英次、日野数司、
松本真一、神谷厚輝、八木希仁、
塩田英二

アジャイルプロセス入門　第Ⅰ部　テキスト
～　アジャイルプロセスを知る　～

発行　2011 年 11 月 1 日　第 1 版
監修　一般社団法人西日本アジャイルプロセス協議会
著者　西日本アジャイルプロセス研究会


WJAPC
West Japan Agile Process Consortium