IT ソリューション塾

クラウド時代のソフトウエア開発

http://www.Ask3S.com

2014/6/25

株式会社　戦略スタッフ・サービス
Strategic Staff Services Corporation

社団法人TPS検定協会

ScrumAlliance transforming the world of work

Copyrights@2013_Strategic Staff Services

1. アジャイル開発とは何か？

2. アジャイルでどのようなビジネスが可能になるか ？

Copyrights@2013_Strategic Staff Services

ユーザーは何を求めているのか？　何が欲しいのか？

サービス

必要な時
必要な物
必要なだけ

JIT

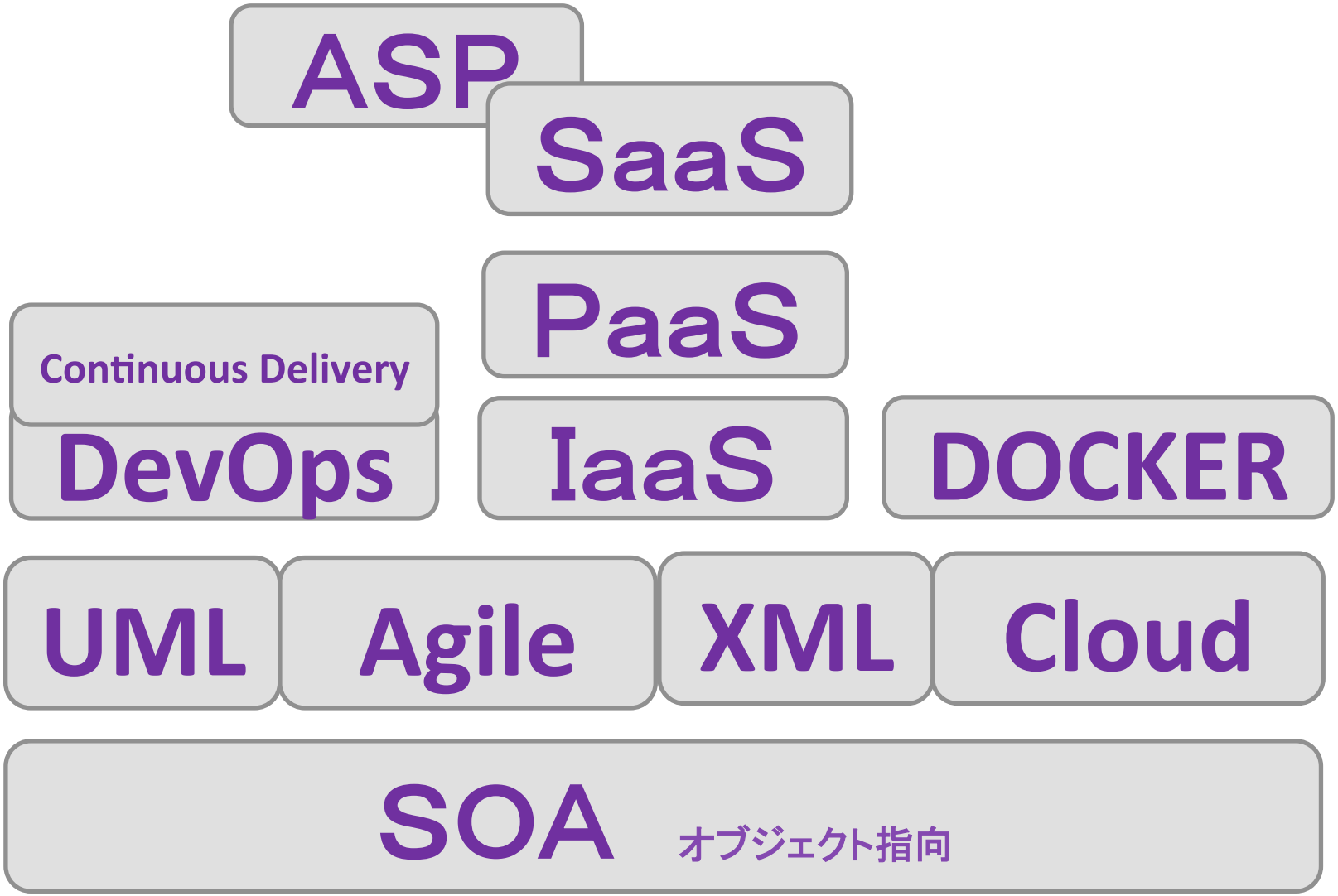
ITの必要悪（不都合な真実）

設備＋ソフトウエア＋IT要員

- ハードウエア
- オペレーティング・システム
- ミドルウエア
- アプリケーション・ソフトウエア
- Webソフトウエア（UI）
- 情報システム要員（IT技術者、運用技術者）

Copyrights@2013_Strategic Staff Services

何が大事な技術なのか？



Copyrights@2013_Strategic Staff Services

第一部

アジャイル開発とは、

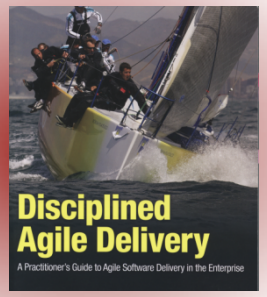
Copyrights@2013_Strategic Staff Services

アジャイル開発に対する偏見、疑念

- ✓ アジャイル開発に適したプロジェクトとは小さなWeb系のシステム？？？
- ✓ 全てを開発者に任せてプロジェクト管理が出来ず納期を守れないのでは？？？
- ✓ 変更を受け入れると、ユーザーからの変更要求が増加しエンドレスなプロジェクトになるのでは？？？
- ✓ 開発者（プログラマー）だけが楽しく仕事できる方法？？？
- ✓ 二人でプログラムを作成（ペアプロ）して本当に生産性が上がるのか？？？
- ✓ ユーザーとの協業なんて言っても、ユーザーが協力してくれる訳がない。
- ✓ インキュベーション（気の合った仲間）や小さな組織には向いているが、大企業では不向き？？？
- ✓ しっかりとしたアーキテクチャーやDBを設計できない。
- ✓ 品質管理が開発チーム任せで品質が保てない？？？
- ✓ エキスパートがやる開発手法では？？？
- ✓ ドキュメントを作らないって無管理状態？？？それでプロジェクトが上手く行く筈がない。
- ✓ PMBoKやCMMI, ISO（ガバナンス）の基準に合わないから使えない。
- ✓ プログラミングとは創造的な仕事だから、クリエーティブな作業ができる静かな環境（個人ブース）が必要。

Copyrights@2013_Strategic Staff Services

アジャイル開発の発展



規律あるアジャイル開発

スケーリングアジャイル開発

古典的アジャイル開発

DSDM
XP
FDD
スクラム

ビジネス価値とアジャイル開発の融合

リーン

2000年 2002年 2004年 2006年 2008年 2010年 2012年

Copyrights@2013_Strategic Staff Services

アジャイル開発とは。

高品質でムダの無い、且つ変更要求に対応できるソフトウエアを作成する為の適切な一連の手順に従い高い協調と自律的なプロジェクト関係者によって実施される反復（周期）的、インクリメンタルなアプローチである。

■ダイナミックシステムズ開発技法（Dynamic Systems Development Method）
デイン・フォルナー（Dane Faulkner）ほか

■アダプティブソフトウェア開発（Adaptive Software Development）
ジム・ハイスミス（Jim Highsmith）

■クリスタルメソッド（Crystal Methods）
アリスター・コックバーン（Alistair Cockburn）

■スクラム(Scrum)
ケン・シュエイバー（Ken Schwaber）、ジェフ・サザーランド（Jeff Sutherland）

■ XP（エクストリームプログラミング）
ケント・ベック（Kent Beck）、エリック・ガンマ（Eric Gamma）ほか

■リーンソフトウェア開発（Lean Software Development）
トムおよびメアリー・ポッペンディーク（Tom and Mary Poppendieck）

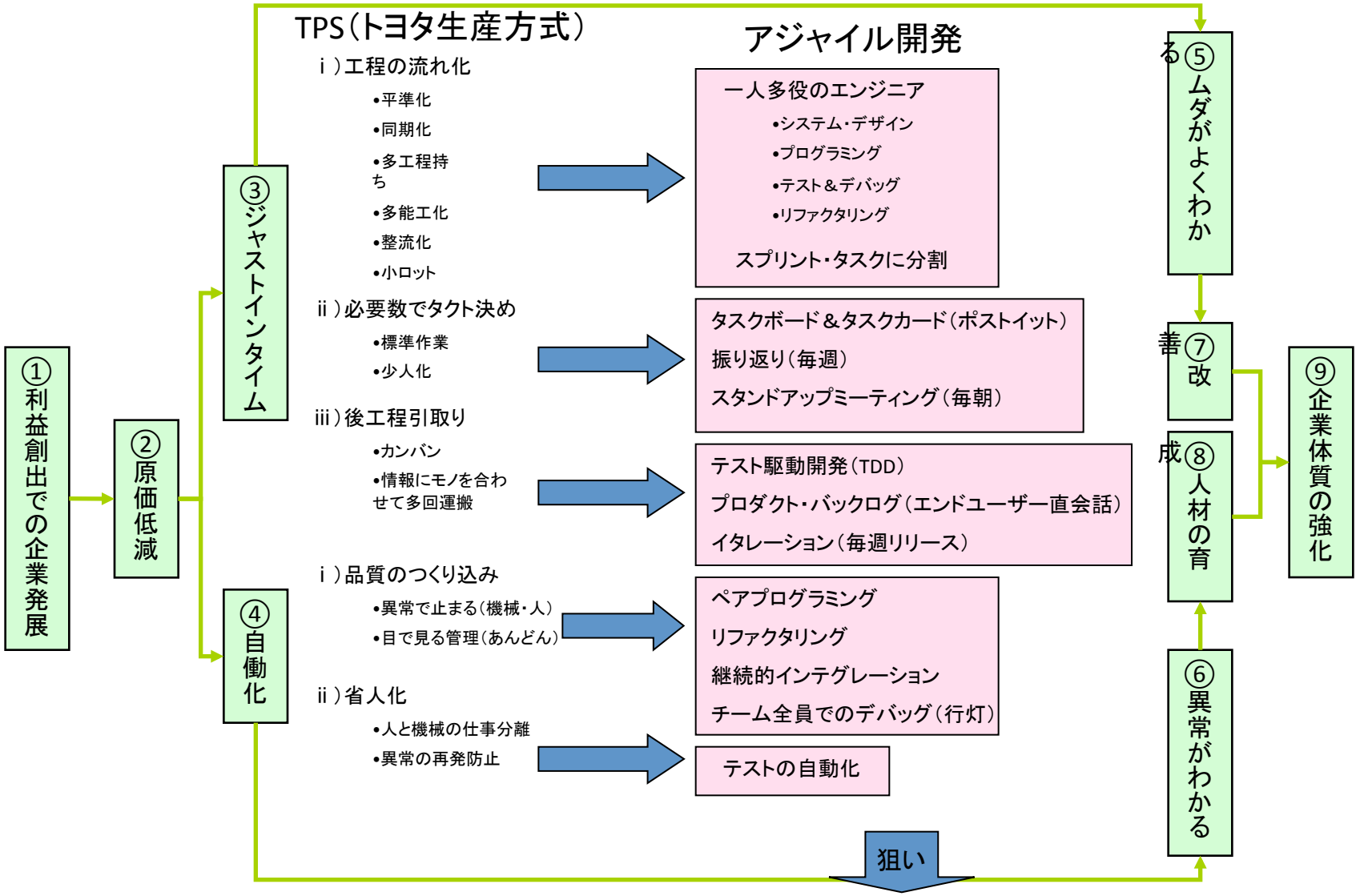
■フィーチャ駆動開発（Feature-Driven Development）
ピーター・コード（Peter Code）、ジェフ・デルーカ（Jeff DeLuca）

■アジャイル統一プロセス（Agile Unified Process）
スコット・アンブラー（Scott Ambler）

- 反復（周期）的(Iterative) --- 定期的なリリース
- 漸進的(Incremental) --- 徐々に機能を増加
- 適応主義(Adaptive) --- 変化に対応（即応）
- 自律的(Self-Organized) --- 学習する組織
- 多能工(Full Stack Engineer) --- 一人多役（SE、プログラマー、テスター）

Copyrights@2013_Strategic Staff Services

アジャイル開発と「トヨタ生産方式（TPS）」

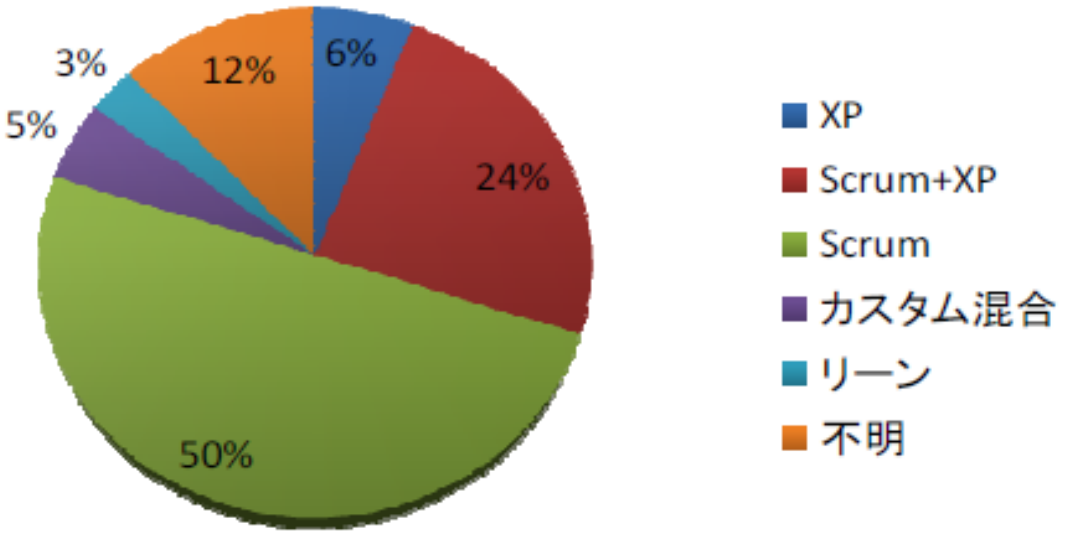


1. 徹底的なムダ排除による原価低減
2. 必要なモノを、必要なときに、必要なだけ
3. 良い考えでより良い品をより安く

Copyrights@2013_Strategic Staff Services

アジャイル開発のポピュラーな手法

現在では、右のグラフにもある様に、
・XP
・スクラム＋XP
・スクラム
の3つで約80%となっている。



■スクラム
プロセスの骨格となる期間や体制、タスク管理方法のみを定める 。
具体的な開発方法は特に規定していないので、様々な手法と組み合わせ可能。

■XP
個々のエンジニアが実践すべき、プラクティスの集合体である。
個々のプラクティスを、別のプロセスと組み合わせて利用可能。

Copyrights@2013_Strategic Staff Services

アジャイル・マニフェスト（2001）ー共通の信念ー

www.agilemanifesto.org より抜粋　　2001年11月13日に作成 （2001年2月、17人の同士がユア州スノーバードにて）

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.

大事な事は、人、人と人との交流と協調、適応力、動作するソフトウエア

我々は自ら実践し、また実践しようとしている人を支援することを通して、よりよいソフトウエア開発の方法を見出そうとしている。この活動をとおして我々は下記のようなことを価値あるものと考えるに至った。

・プロセスやツールよりも、**個人や相互関係を**。

・包括的なドキュメントよりも、**働くソフトウエアを**。

・契約折衝よりも、**顧客とのコラボレーションを**。

・計画に従うことよりも、**変化への対応を**。

※これは太字の項目をより重視すると言うことであり、細字の部分を無視して良いと言う事ではない。

Copyrights@2013_Strategic Staff Services

マニフェストの思想を支える重要な方針（原理）

1.我々の最優先事項は素早いそして継続的な価値あるソフトウエアの提供を通して顧客の満足を得る事である。

2.開発局面の後半でも要求変更を歓迎する。アジャイルなプロセスを顧客の競争優位の為の変化に利用する。

3.稼動するソフトウエアをより短かい期間を優先して、数週間から数ヶ月で定期的に提供する。

4.プロジェクト期間を通して業務ユーザーと開発者は共同して作業をしなければならない。

5.やる気のある個人を集めてプロジェクトを組織し、彼らが必要とする環境と支援を与え、仕事が完了するまで信頼する。

6.部門内、間のコミュニケーションで最も効率的かつ効果的な手法は、フェイスツーフェイスの会話である。

7.ソフトウエアが正常に機能するということが進捗の基本的な評価である。

8.アジャイルプロセスは持続可能な開発を促進する。

9.スポンサー、開発者、ユーザーは無期限かつ不断に保守できるようにしなければならない。

10.技術的に優れた良い設計に継続的に配慮する事は機敏性（アジリティー）を増長させる。

11.簡素が基本

12.最良の構想、要求仕様、設計は自己統制された（自律的）チームより出現する。

13.定期的にチームが振り返りを行い、より効果的に出来る方法を思案し、チームの行動に協調と調整が働く。

Copyrights@2013_Strategic Staff Services

アジャイルの本質 ― 様々なパラダイムの変更

プロセス	ウォーターフォール開発		アジャイル開発
成功の測定	計画への適合	→	変化への対応 動くコード
マネジメント文化	命令と制御	→	リーダーシップ 協業的
要求と設計	大きくて前払い	→	継続的・発現的 ジャストインタイム
コーディングと実装	全機能を同時にコーディング後でテスト	→	コーディングとユニットテスト 順番に納品
テストと品質保証	大きく、計画に基づく 遅くテスト	→	継続的・同時発生 早期にテスト
計画とスケジューリング	PERT・詳細・範囲固定 時間と工数を見積もり	→	2レベル計画・期日固定 範囲を見積もり

Copyrights@2013_Strategic Staff Services

ウォーターフォール型開発（従来）の課題

ユーザーと開発者の信頼関係が希薄であり、長期間のプロジェクトの間に何処まで進行しているか？要求を満たしているのか？確認する術が無い。完成して初めて解る。

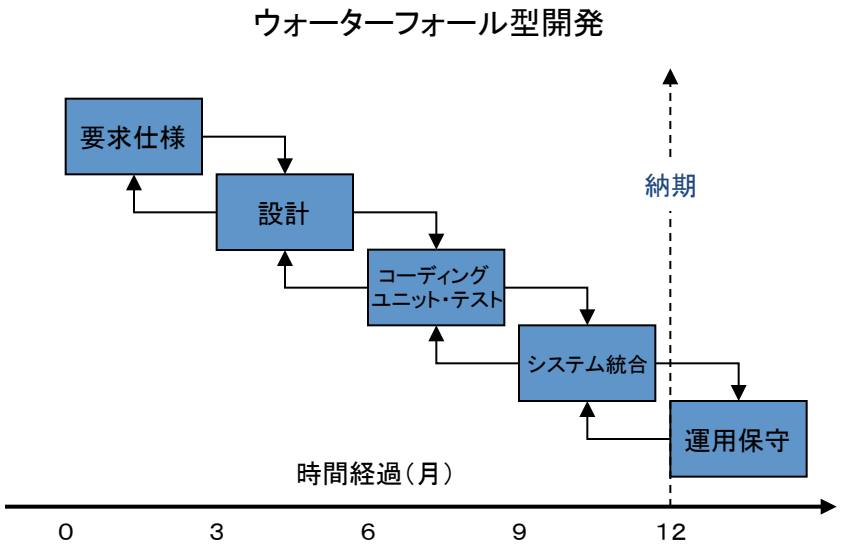
また昨今のビジネス・スピードの短期化により設計時点での要求内容そのものも時間経過と共に時代遅れになる。要求内容が変質する。

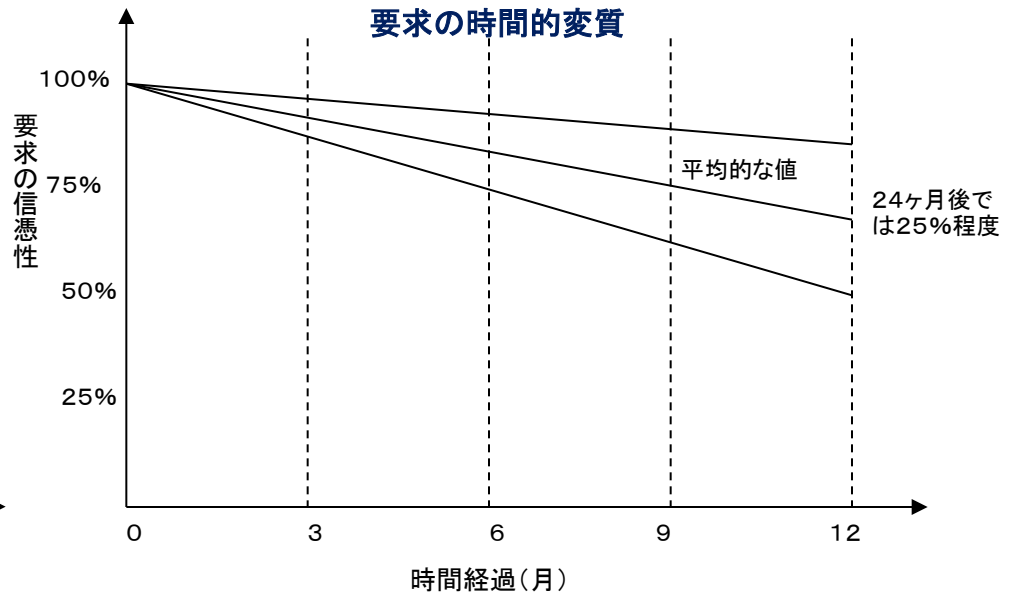
シリアルな開発プロセスで、開発プロセスの中にボトルネックとなる工程が発生する。

大型化してプロジェクトの成功率が低い

Standish GroupのCHAOS surveyより

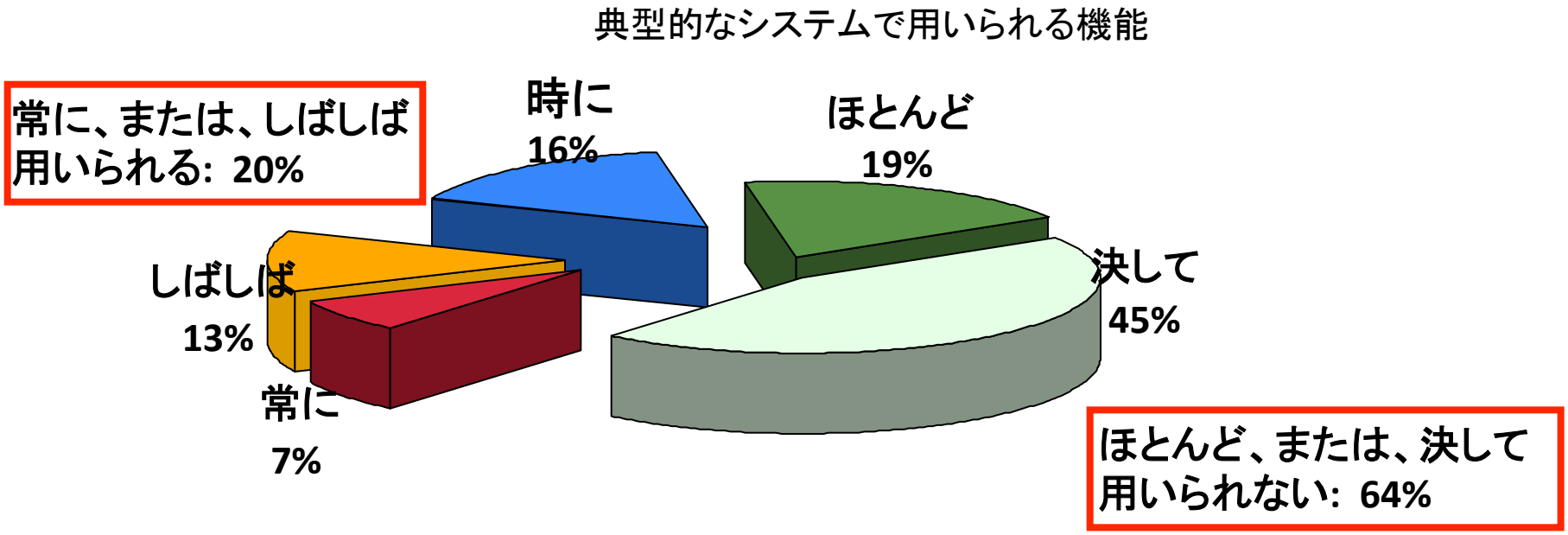
Successful project	16%	プロジェクトサイズを適正化（小口化）1998年	26%
Failure projec t	31%		28%
Challenged project	53%		46%





Copyrights@2013_Strategic Staff Services

早期の仕様確定がムダを減らすというのは迷信



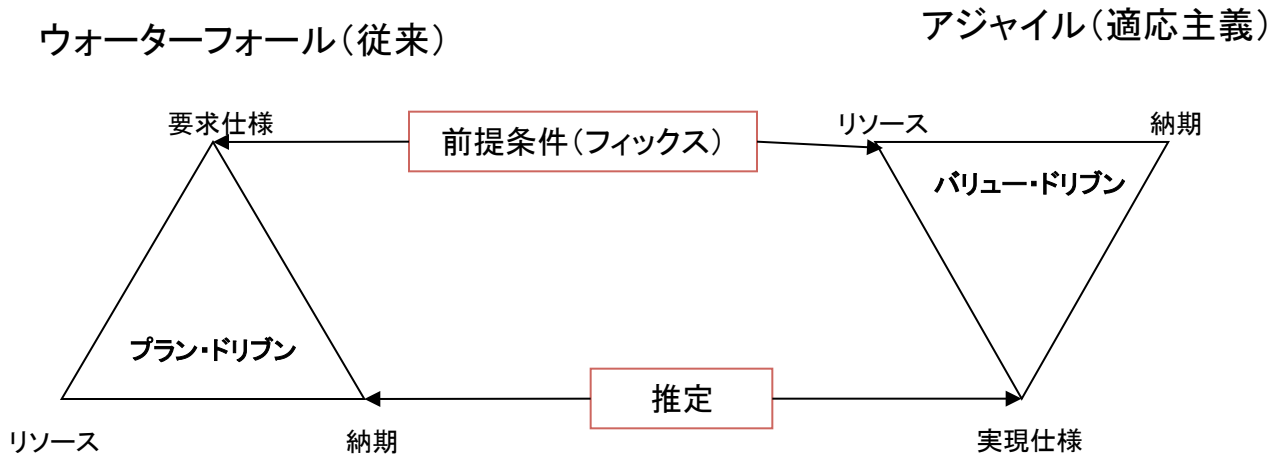
早期に仕様確定しようとすると、ユーザーはリスクを見越して何でも仕様に入れたがる

その結果、使われない機能が盛大に作り込まれる

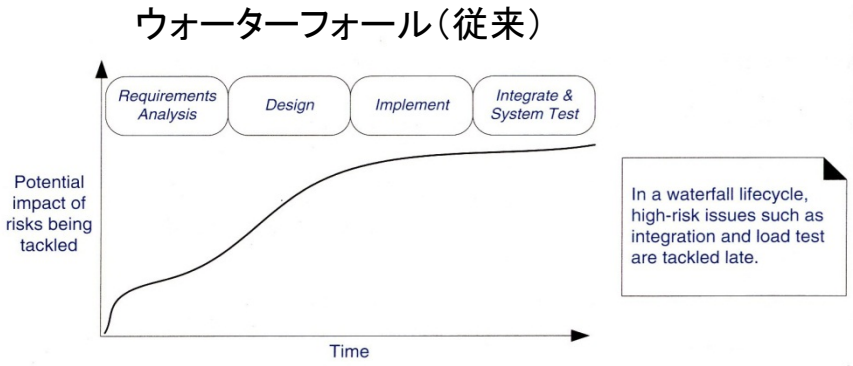
Standish Group Study Reported at XP2002 by Jim Johnson, Chairman

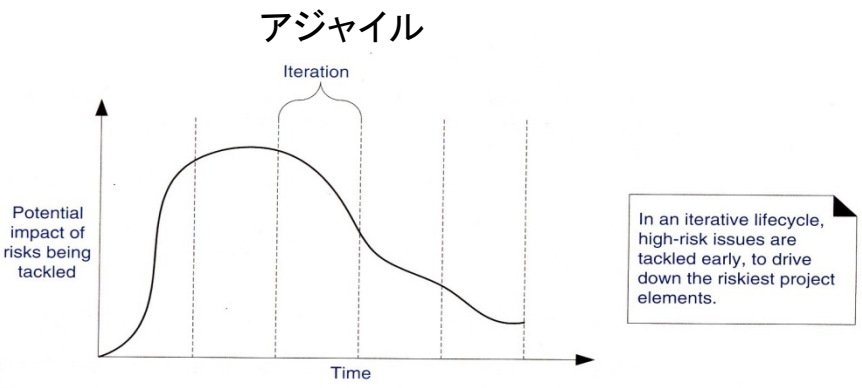
Copyrights@2013_Strategic Staff Services

基本的価値観の相違



リスクに捉まる潜在的な影響度

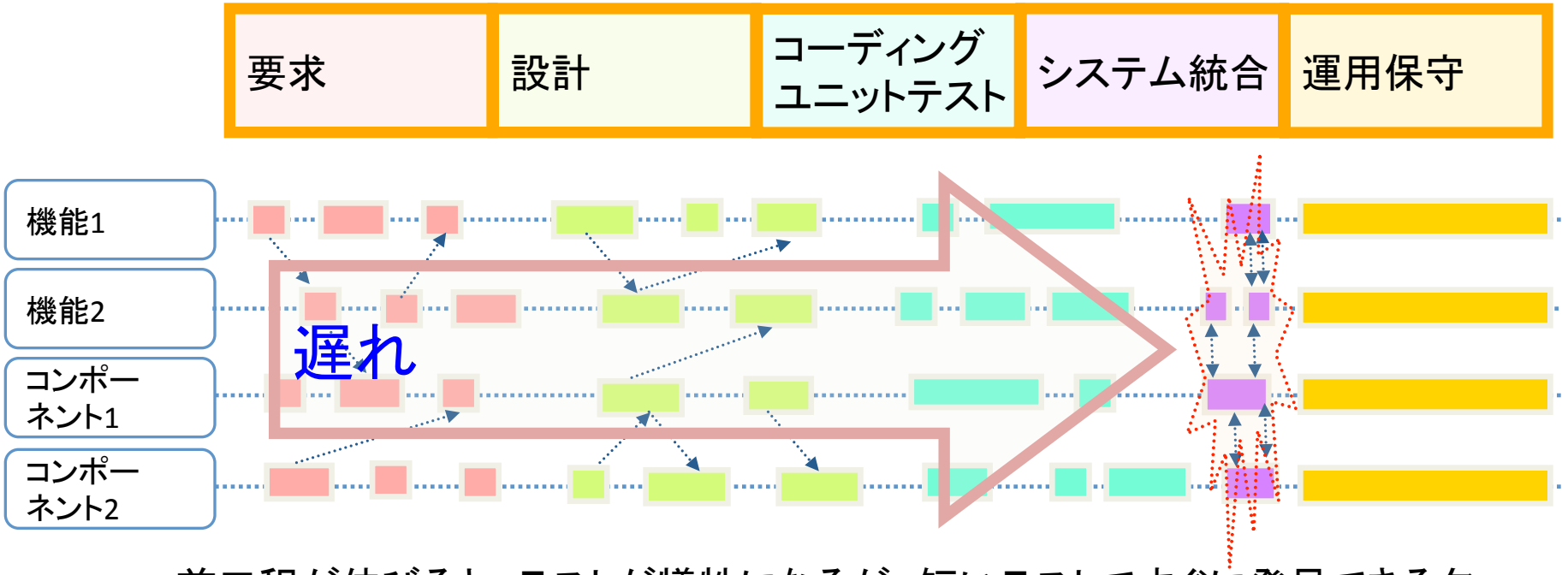




Copyrights@2013_Strategic Staff Services

ウォーターフォールプロジェクトの解剖学

なぜウォーターフォールプロジェクトで品質が犠牲になるのか

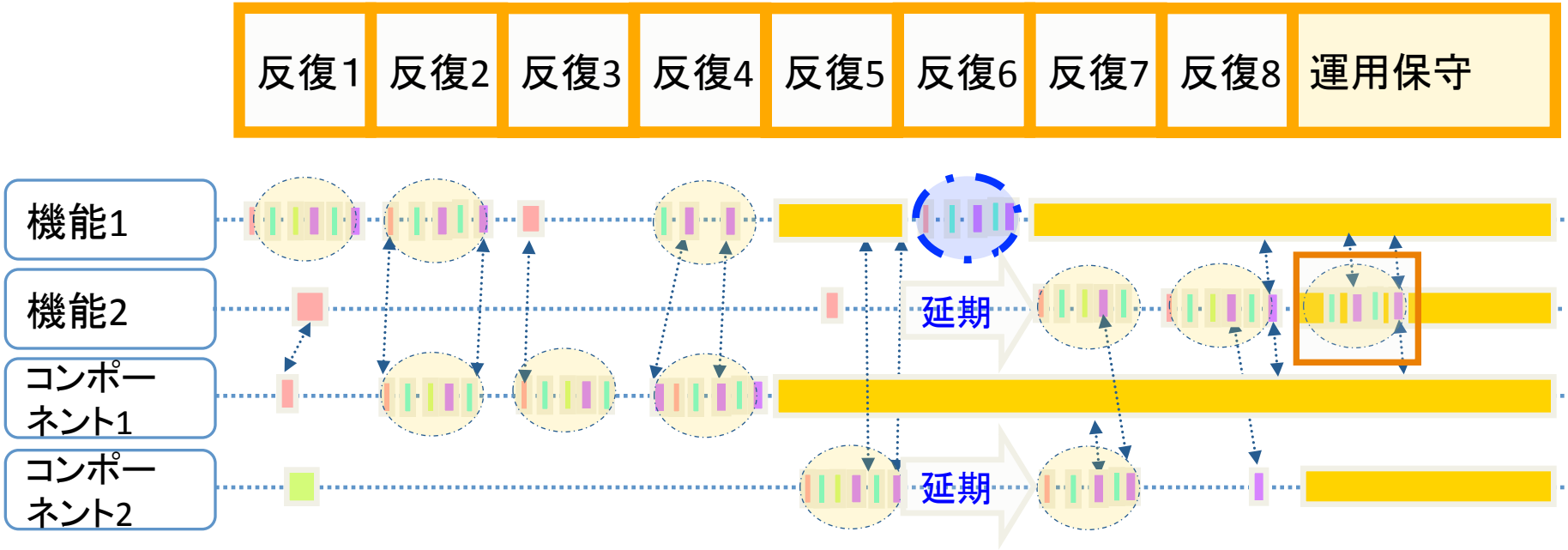


- 前工程が伸びると、テストが犠牲になるが、短いテストですぐに発見できる欠陥はつぶせるため、表面上はOKなシステムが納品される
- テストに時間をかけていないので、多大な潜在的欠陥が残る
- システム統合時点でリスクが顕在化しても、そのとき範囲を調整することは、非常に困難

Copyrights@2013_Strategic Staff Services

アジャイルプロジェクトの解剖学

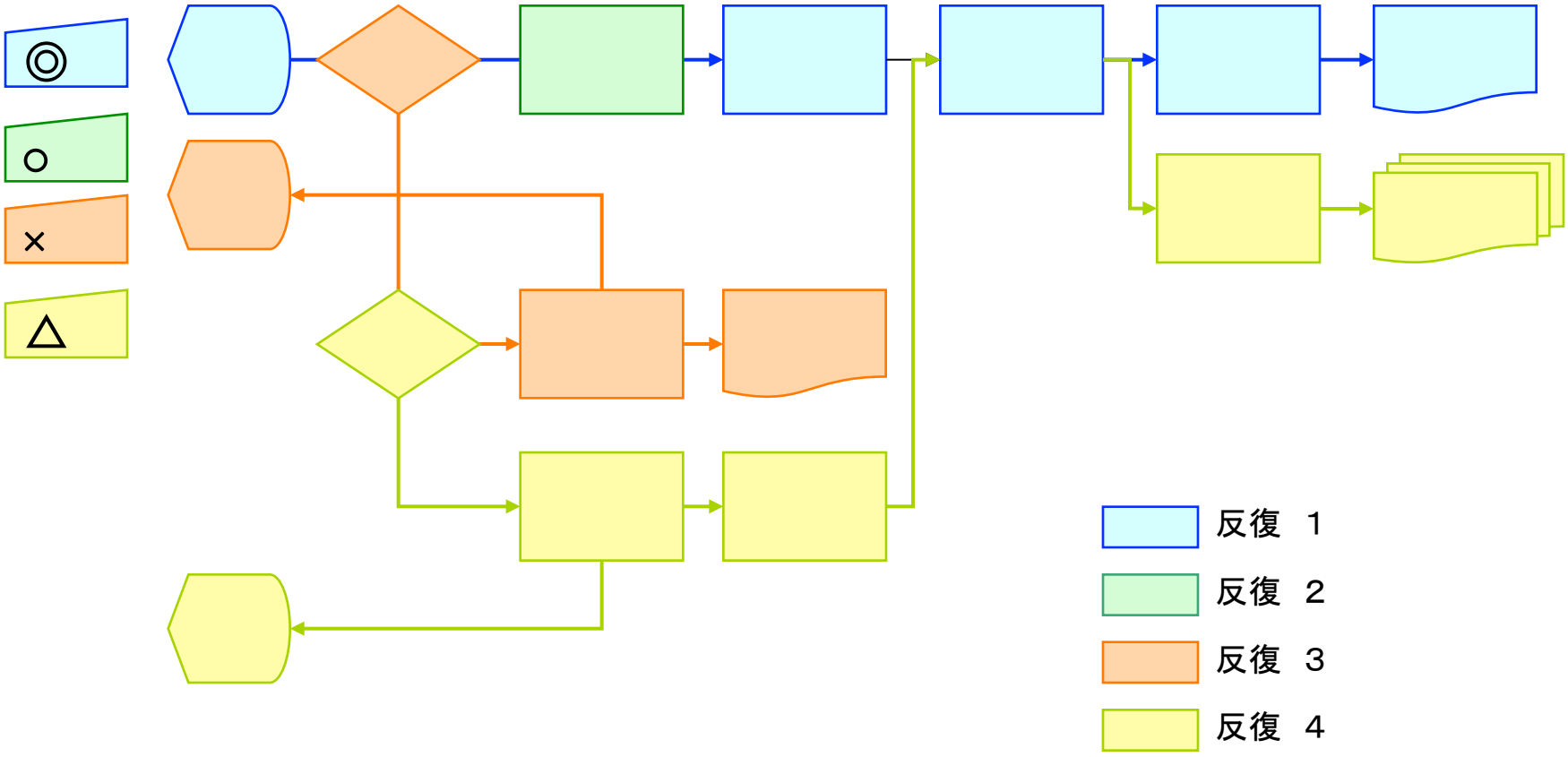
変更が入った場合の対処



- すでにリリースした機能1に関して新たな要求が生じた場合 、機能2とコンポーネント2の反復を遅らせて、その対応に当てることができる（全体の優先度評価による）
- 機能2の最後の反復は 、反復を増やすことで対応してもよいし、その部分を開発範囲外とすることで対応してもよい
- ここで重要なことは、常に優先度を見直して、優先度の順に開発されているということ
  - 最後の機能は、範囲からはずしても大きな影響はない

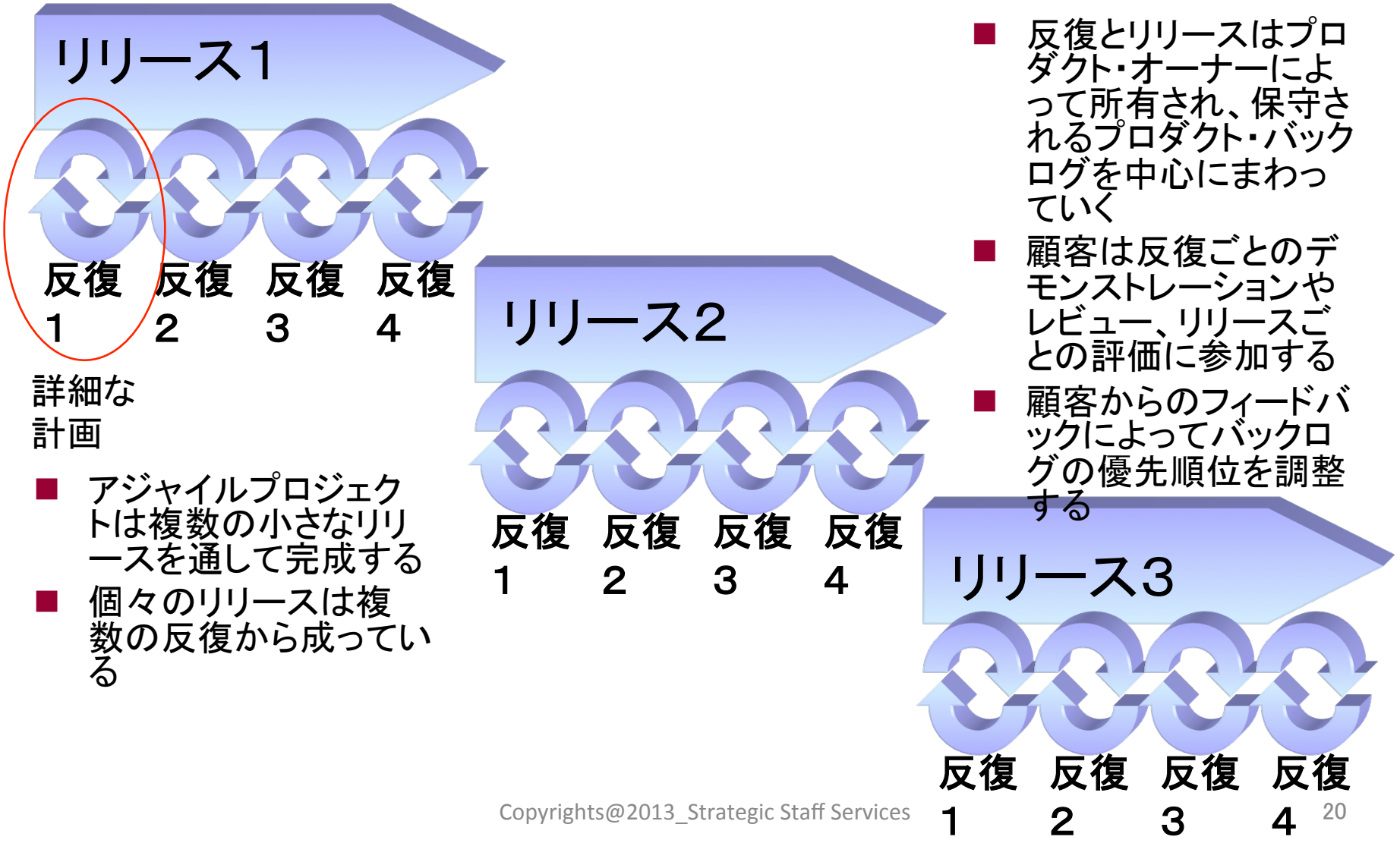
Copyrights@2013_Strategic Staff Services

アジャイル開発におけるソフトウエアの作り方



Copyrights@2013_Strategic Staff Services

アジャイルの計画（リリース計画と反復計画）



- アジャイルプロジェクトは複数の小さなリリースを通して完成する
- 個々のリリースは複数の反復から成っている

- 反復とリリースはプロダクト・オーナーによって所有され、保守されるプロダクト・バックログを中心にまわっていく
- 顧客は反復ごとのデモンストレーションやレビュー、リリースごとの評価に参加する
- 顧客からのフィードバックによってバックログの優先順位を調整する

Copyrights@2013_Strategic Staff Services

Scrumによるアジャイル・プロジェクト運営（管理）

Product Owner（お客様）

Product Backlog
1.----------
2.----------
3.----------
4.----------
5.----------

確実な優先順位付け（同位は無し）

追加、修正、変更可能（Sprintに入っている物以外）

Product Ownerが全て責任を持つ

Sprint Planning

Part 1	2時間
Part 2	2時間

Scrum Master（ファシリテーター）

Team　デザイン・開発・テスト

Developer (Pair Programming)

毎朝15分のスタンドアップ・ミーティング

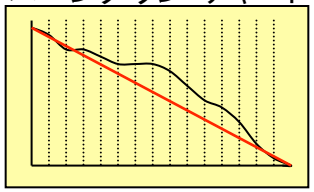
Daily Scrum

Sprint Task List (Sprint Backlog)
a.----------
b.----------
c.----------
d.----------
e.----------
f.----------

タスクボード

To Do	On Going	Done
c, d	b	a
e, h	g	f

バーンダウン・チャート



Sprint Review 2時間

Retrospective 2時間

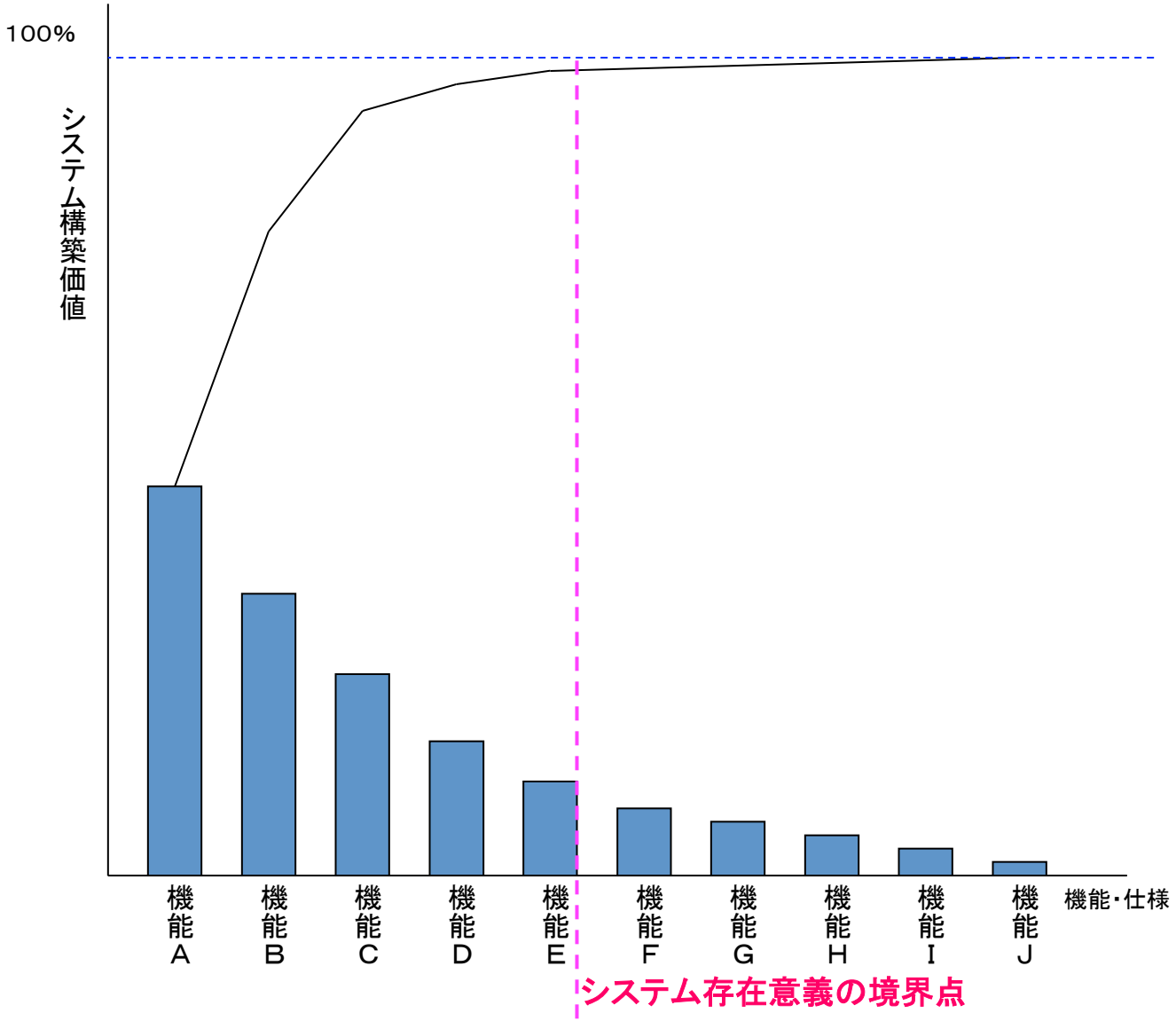
Continuous Integration (継続的インテグレーション)

Sprint (Iteration)

1週間～4週間

Copyrights@2013_Strategic Staff Services

J. Juranのパレート図を用いたプロダクト・バックログ管理



Copyrights@2013_Strategic Staff Services

スクラム・チーム構成例

管理者

人事管理、業績管理

開発チーム（スクラム）

仕様確認、設計、作成、テスト、文書化、スケジュール管理

ペアプログラミング

スクラム・マスター

（Scrum Master）

プロジェクト関係者のファシリテーション

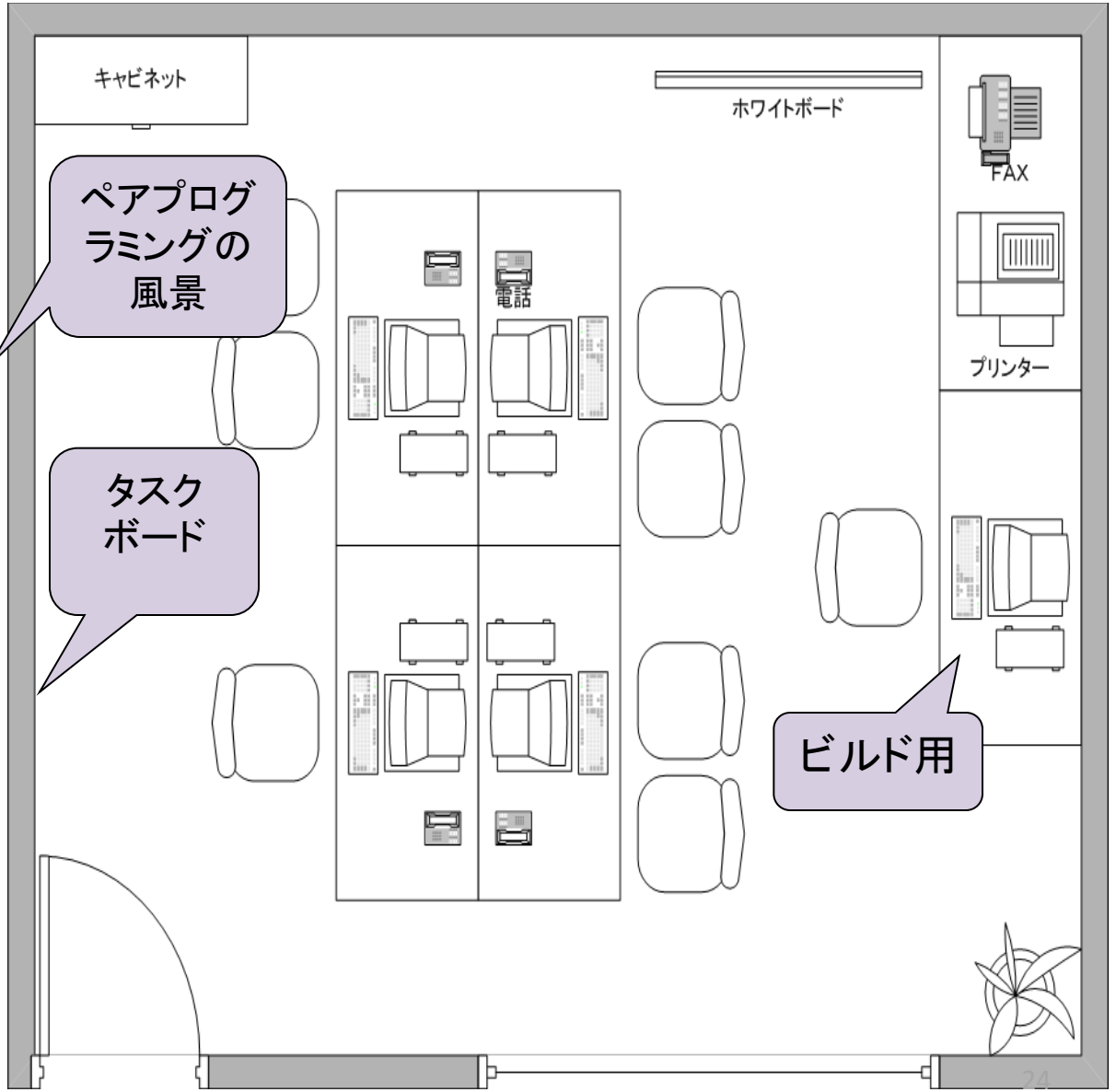
支援スタッフ

（必要に応じて配置）

特定知識・スキルの提供支援

Copyrights@2013_Strategic Staff Services

スクラムの環境



Copyrights@2013_Strategic Staff Services

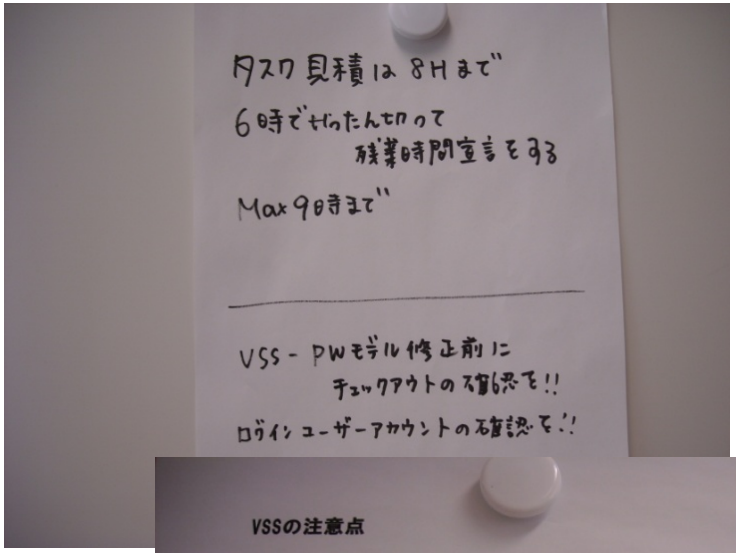
情報共有の環境

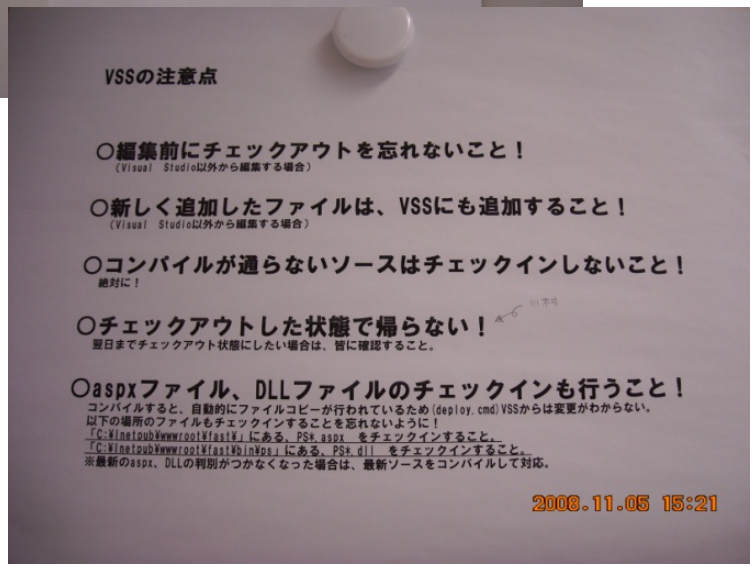
- 情報共有
  - ホワイトボードや壁に張付け
  - 特別なツールよりも身近な方法選択

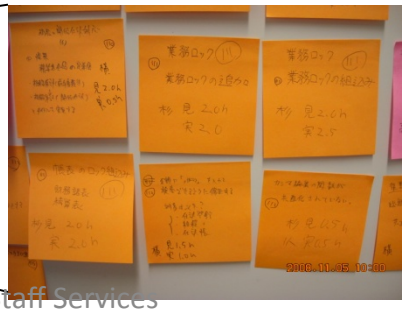
Copyrights@2013_Strategic Staff Services

状況の見える化を推進

- ルールを壁に張る
  - タスク分割・残業・作業環境等
- 見える化
  - タスクボード（ストーリの状況）
    - 作業予定・作業中・作業終了
      - 作業前後に見積と実績を記入
      - **色分けで遅れタスクが一目然**
    - タスクチケットはポストイットを使用



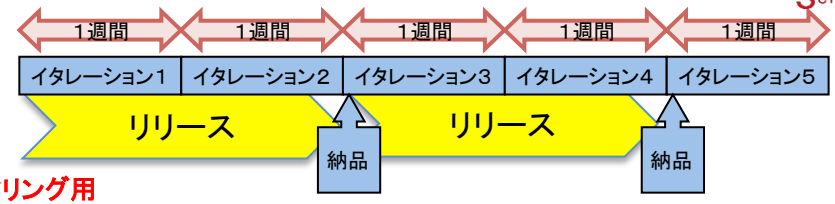


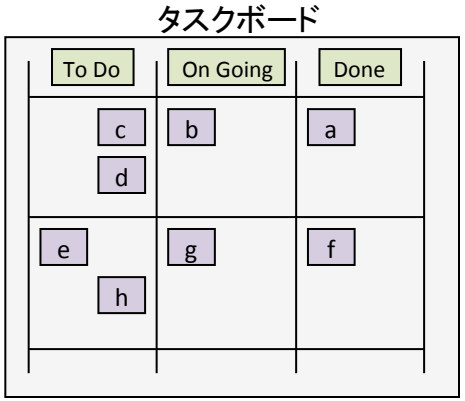


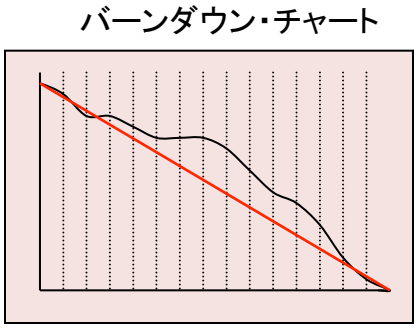
Copyrights@2013_Strategic Staff Services

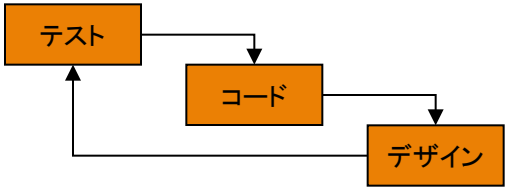
アジャイル開発の特徴

- **イタレーション／リリース**
  - 反復的（1週間）に作業
  - 周期的な納品（1週間～8週間）
- 開発チーム
  - 自己統制型、オーナーシップ（自己申告）
  - 開発プロセスの標準化
  - 技術、情報の共有化（見える化）
- 簡潔なデザインとコード
  - リファクタリング
- ペア・プログラミング
  - コードの標準化
  - 技術の共有、技術伝承
- タスクボード
  - 機能の提供（完了）をもって進捗を把握
  - 作業の宣言（タスクと作業時間、見積工数）
- **テスト駆動開発**
  - テストの自動化
  - アクセプタンス・テスト（完了・終了の定義）
- **継続的インテグレーション**（基本的に毎日昼）
  - 行灯
  - 自動テスト
- 簡潔な文書化
  - ユースケースか？ユーザーストリーか？
  - UML

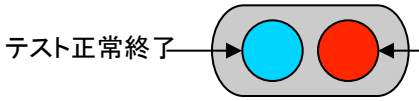








小さいテストのコード作成
次にソース・コードを作成
最後にデザインをリファクタリングする。
このプロセスを廻す。



Copyrights@2013_Strategic Staff Services

ペア・プログラミング（Pair programming）

- 二人で一台の端末に向かってプログラム製造を実施する
  - 毎日違ったペア
    - 緊張感の維持
    - この瞬間にこの作業しか出来ない！！
  - 役割を一時間半前後で交代
    - 作業者とチェッカー
    - 曖昧な作業が無くなる
    - 出戻りの軽減
  - 品質の向上
    - 動けば良いと言うような安易なプログラム製造ではない
  - 頭脳労働のムダとり
    - 一人で考え込む時間が無くなる。

ペアの交替（定期的：毎日）
作業の交替（タスク、時間毎）：ドライバーとナビゲーター

Copyrights@2013_Strategic Staff Services

継続的インテグレーション

- 開発したプログラムを毎日統合して動作確認を行う
- システムの進捗を日々の動作機能で測る事が可能
- 開発チーム内での統合動作不良・改善を日々行う事が可能
- 自動化ツールを使用して開発チームが休息している間も統合試験が可能
- 品質向上に貢献

Copyrights@2013_Strategic Staff Services

リファクタリング（Refactoring）

- 正常に動作確認したプログラムに対して下記目的でソースコードの調整を行う作業
  - プログラムの可読性を高める
  - クラス化・部品化
  - 変更容易
- 仕様レベルでの調整
  - 使い易さ
  - 誤操作抑止
  - 仕様変更

Copyrights@2013_Strategic Staff Services

デイリー・スクラム（スタンドアップ・ミーティング）

- 毎日行われる15分程度の状況確認ミーティング
  必ずチームメンバー全員が参加する

- チーム内での最良のコミュニケーションと見える化の仕組み
  - 非常に簡単な事だが、毎日継続して着実に実施する事が大事
    （たかが朝礼という認識は誤り）

- **チームメンバーは以下の三つの質問に対し、簡潔にポイントを絞って答える**
  1. **昨日は何をした？**
  2. **今日は何をする予定？**
  3. **今現在抱えている、若しくは気になる問題、課題は何？**
     - » スタンドアップ・ミーティングに参加する人は全員が等しく順番にこの発表を行う。管理職の方が参加される場合も例外ではない。スタンドアップ・ミーティングはチーム全員（管理職もチームの構成員です）の情報共有、見える化の重要なコミュニケーションである。

- 必要に応じてフォローアップ・ミーティングを行なう

この活動は、報告、連絡、相談。（報・連・相）
いわゆる作業現場でのコミュニケーションの基本ですから、しっかりと毎朝時間を決めて実施する。

Copyrights@2013_Strategic Staff Services

スプリント・レトロスペクティブ（毎週の振返り）

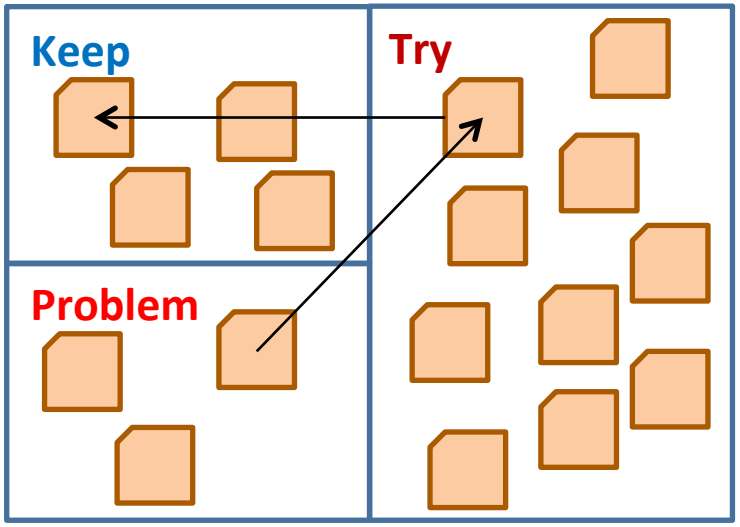
- 毎週行われる進捗報告ミーティング、評価とプロセスの改善の重要な機会
- PTKをチームが共有する
  - P（Problem：問題となっていること）
  - T（Try：トライしたいこと）
  - K（keep：キープしたいこと）

- 毎週の振返りは、管理職は参加せず、チームが自主運営する事が大事
- チームの中で自由に何でも話し合える環境、雰囲気作りが重要
  この環境が構築されないとチーム内で透明性を確保できない
- 振返りの話題、議論は、仕事に直接的に関係する話題だけで無く、チームの置かれている環境など間接的、補助的な事柄や、環境条件面についての話題も取り上げる
- 話題に上がった問題、課題は、チーム全員で解決、改善して行く事になるが、チームが解決できないものは要望事項として後日チーム・リーダーから管理者等へ依頼する
- 管理者は振返りの会議の場に参加してはいけない
  後日、チームリーダーから報告や要望を聞く
  注意すべき事は、指示、命令的な発言を差し控える、コメントは直接的な表現を避けてチームに気付きを与えられる言葉に止める、褒める言葉を多用する事
  基本的にはチームの要望事項を聞きその解決を図る

Copyrights@2013_Strategic Staff Services

（参考）TMSの振り返り（レトロスペクティブ）

- 毎週行われる進捗報告ミーティング
- 評価とプロセスの改善の重要な機会
- PTKをチームが共有する
  - P（Problem：問題となっていること）
  - T（Try：トライしたいこと）
  - K（keep：キープしたいこと）



小さなPDCAを回すための方法です。

KPTボードは振り返りのツール

①気づきや問題、課題を付箋紙に書き、Problem（問題）のエリアに貼る。

②Problemのエリアに出てきた気づきや問題に対してTry（対策）を立ててTryのエリヤに移動させる。

③実際にやってみて、対策が不十分だった場合には、再度、Problem（問題）のエリアに移動し課題を追記する。再度、対策を立て直す。

④Problemに対する効果が十分出たと判断できた場合には、Keep（継続）のエリアに移動し習慣化、定着化を図る。

ここで大事な事は、**人を責めずにやり方を責める**事です。（誰でも同じようにできる様にする。）

Copyrights@2013_Strategic Staff Services

IBM社内でのアジャイルプラクティスの適用とその効果

		2007	2008	2009
1	ステークホルダー -ゴール		40%	37%
2	ユースケース	51%	61%	86%
3	セルフ-ダイレクション	51%	64%	75%
4.0	反復	62%	75%	81%
4.1	チーム全体		73%	100%
4.2	バックログ		73%	90%
4.3	ベロシティでの見積もり		63%	65%
4.4	タイムボックス			90%
4.5	早期でのテスト	59%	67%	80%
4.6	Stability / High Quality	61%	74%	75%
4.7	ステークホルダー -フィードバック	53%	64%	74%
4.8	リフレクション		63%	55%
5	デイリースクラム	34%	62%	76%
6	継続的インテグレーション		66%	71%
7	単体テスト	55%	55%	63%
8	テスト駆動型		26%	26%
9	レビュー			
9.1	静的解析	34%	42%	56%
9.2	レビュー	56%	63%	89%
9.3	インスペクション			44%
9.4	レビュワー			44%
9.5	ペア -プログラミング			13%
10	サステイナブル -ペース		64%	71%
11	顧客中心のテスト			13%
12	バリューストリームマップ			6%

		2009	
		適用	効果
1	ステークホルダー -ゴール	37%	+0.0%
2	ユースケース	86%	+42.9%
3	セルフ-ダイレクション	75%	+5.3%
4.0	反復	81%	+40.0%
4.1	チーム全体	100%	+52.6%
4.2	バックログ	90%	+52.6%
4.3	ベロシティでの見積もり	65%	-5.3%
4.4	タイムボックス	90%	+21.1%
4.5	早期でのテスト	80%	+10.5%
4.6	Stability / High Quality	75%	+42.1%
4.7	ステークホルダー -フィードバック	74%	+26.3%
4.8	リフレクション	55%	+31.6%
5	デイリースクラム	76%	+15.0%
6	継続的インテグレーション	71%	+30.0%
7	単体テスト	63%	+42.1%
8	テスト駆動型	26%	-29.4%
9.1	静的解析	56%	+16.7%
9.2	レビュー	89%	+33.3%
9.3	インスペクション	44%	-16.7%
9.4	レビュワー	44%	-25.0%
9.5	ペア -プログラミング	13%	-46.7%
10	サステイナブル -ペース	71%	+25.0%
11	顧客中心のテスト	13%	-80.0%
12	バリューストリームマップ	6%	-81.8%

Copyrights@2013_Strategic Staff Services

ソフトウエア開発手法と欠陥（バグ）の防止＆その除去
----米国での調査結果----

欠陥と開発手法-1　（Capers Jones氏の資料より抜粋）
(このデータはファンクション・ポイント当たりの欠陥を表している。1000 ファンクション・ポイント規模のプロジェクトの例)

開発手法	想定される欠陥数	欠陥除去率	残存の欠陥数
ウォーターフォール	5.50	80%	1.10
イタラティブ（反復型）	4.75	87%	0.62
オブジェクト志向	4.50	88%	0.54
RUP（Rational Unified Process）	4.25	92%	0.34
アジャイル	**4.00**	**90%**	**0.40**
PSP ／ TSP	3.50	96%	0.14
再利用（85% certified）	1.75	99%	0.02

欠陥と開発手法-2　（Capers Jones氏の資料より抜粋）
(このデータはファンクション・ポイント当たりの欠陥を表している。10,000 ファンクション・ポイント規模のプロジェクトの例)

開発手法	想定される欠陥数	欠陥除去率	残存の欠陥数
ウォーターフォール	7.00	75%	1.75
イタラティブ（反復型）	6.25	82%	1.13
オブジェクト志向	5.75	85%	0.86
RUP（Rational Unified Process）	5.50	90%	0.55
アジャイル	**5.50**	**87%**	**0.72**
PSP ／ TSP	5.00	94%	0.30
再利用（85% certified）	2.25	96%	0.09

Copyrights@2013_Strategic Staff Services

有益な欠陥防止策（手法）　（Capers Jones氏の資料より抜粋）

- ◆ Joint Application Design (JAD)
- ◆ Quality function deployment (QFD)
- ◆ ソフトウエアの再利用(高品質コンポーネント)
- ◆ Root cause analysis（真因分析）
- ◆ シックスシグマ品質プログラム（ソフトウエア）
- ◆ TSP／PSP 手法の利用
- ◆ SEI CMMIのレベル３以上
- ◆ スタティック分析、検査
- ◆ ライフサイクル品質測定
- ◆ カイゼン、ポカ除け、カンバン、QCサークル
- ◆ プロトタイプ作成（プロトタイプは廃棄される）
- ◆ 欠陥追跡ツール
- ◆ 正規のデザイン・インスペクション
- ◆ 正規のコード・インスペクション
- ◆ 開発チームとユーザーの協業（アジャイル手法）
- ◆ スクラム（各種チームでの課題志向のミーティング）

Copyrights@2013_Strategic Staff Services

アジャイル開発で品質が向上する理由

- ムダ取りの原則
  - 作業にムラがあるから、ムリをするようになる。
  - ムリな作業をした結果、ムダが生じる。
  - ムラを防止するのは、一定の作業リズム（タクトタイム＝タイムボックス）

- タスクの粒度を小さくする。
  - 作業手順（工程設計）を考えなければタスクは小さくできない。
  - タスクが小さくなれば、ミスを容易に発見できる。手戻りも小さい。
  - タスクを小さくするとムダが見えてくる。
  - 正味（本来の価値を作り込む）作業、付帯（事前、事後の）作業、ムダな作業
  - タスクを小さくすることで、整流化が容易になる。（ボトルネックの解消： 一個流し生産）

- 全員での作業で透明性が高まる。
  - 一人で抱え込む仕事がなくなる。
  - 事前に他人の目が届く（チェック）

- トヨタ生産方式（TPS）の自働化の思想をプロセスに組み込める。
  - 不良作業をしない。不良品を流さない。不良品を受け取らない。（自工程完結）
  - 不良が起こったらラインストップをして、関係者全員が異常に気づく。（見える化）

- 作業者（開発者）に直接フィードバックする仕組みが構築できる。
  - 『擦り合わせ』をしながら作業が進む。

Copyrights@2013_Strategic Staff Services

ソフトウエア製造工程におけるムダの廃除

1. 作りすぎのムダ
   顧客に使用されない機能、実際には不要な機能、真のビジネス価値を生まない機能などの余分な機能を作らない。
   StandishのCHAOSレポートによるとソフトウエアの全機能の64%は全くあるいは殆ど使用されていない。
2. 手待ち（停滞）のムダ
   仕様の提示遅れによる製造開始遅れでの待機、許可待ち、ビルド待ち、障害発生によるテスト待ち
3. 運搬のムダ
   複数プロジェクトでの作業の切り替え、仕様入荷チェック、納品出荷チェックの提供＆受領双方での重複チェック
4. 加工そのもののムダ
   開発現場で機能していない作業項目例えば余分な事務処理、報告書作成や作業分担の誤りによる過剰な作業
5. 在庫のムダ
   最終工程で使用されない文書や計画、コンポーネントなどの中間的な作業成果物と待ち状態で仕掛中のプログラム
6. 動作（作業）のムダ
   関係者の作業場所の移動や複数の開発ツールを使用して開発ツール間の切替・移行
7. 不良をつくるムダ
   バグの作り込み---要求仕様(要件)、設計、コードの欠陥

Copyrights@2013_Strategic Staff Services

（参考）アジャイル開発におけるタイムボックスの価値

＝　やる気と集中力

『仕事の量は、完成の為に与えられた時間を全て使い切るまで膨張する』

イギリスの歴史学者・経済学者であるパーキンソンの言葉

時間には弾力性がある。
時間は、何となく使ったのではいくら有っても足りない。
同じ仕事量でも、意識の違いでかかる時間は全く異なる。

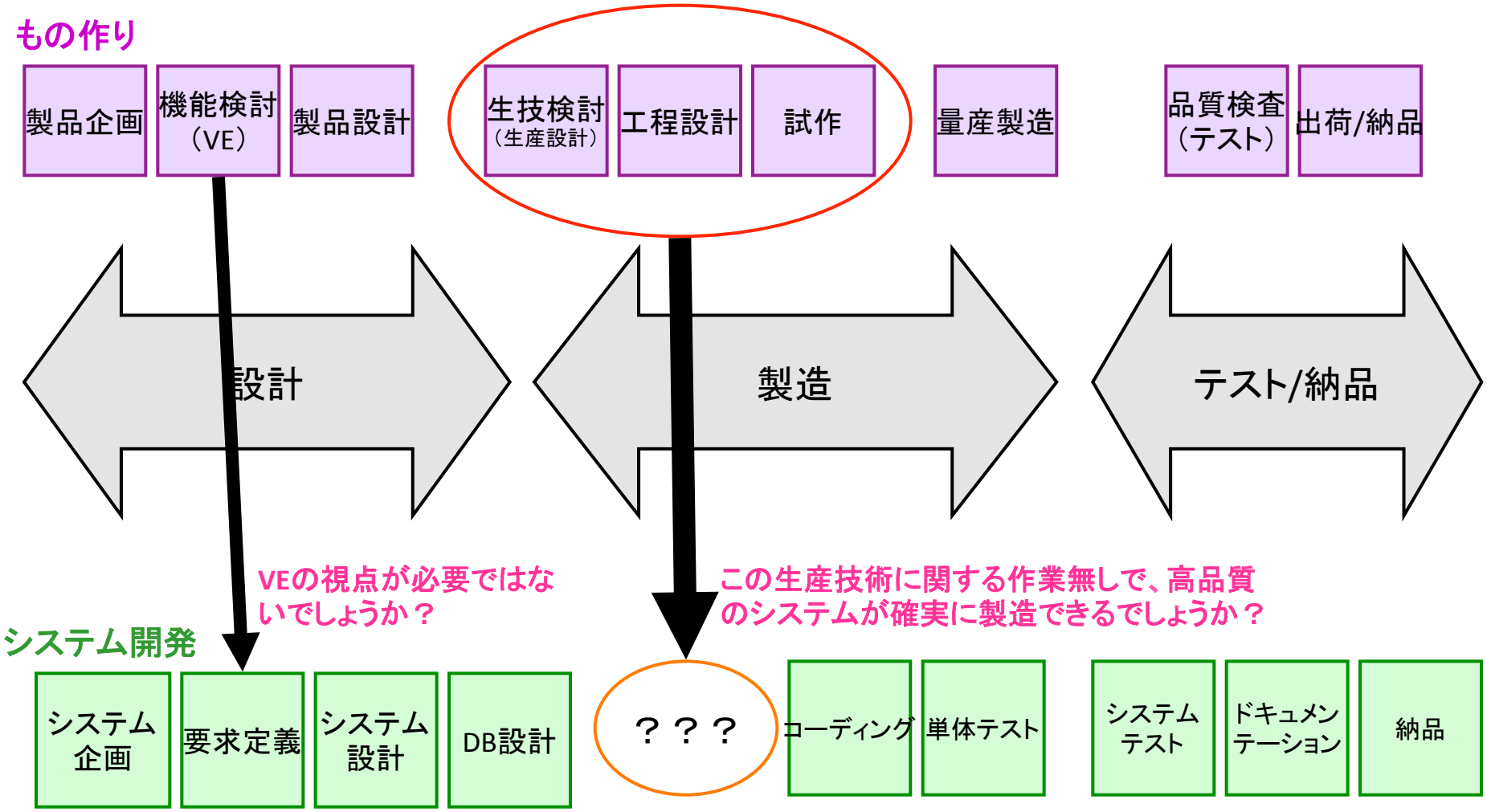
生産性はやる気と集中力で高まる。
やる気のホルモン　＝　ドーパミン
ドーパミンは、ご褒美によって放出される。
やる気はご褒美の事を考えるだけで出る。しかし、裏切られると一瞬で低下する。
ご褒美の60秒ルール　＝　ご褒美は直ぐに貰える事が重要。楽しく想像できる事が重要。

スピードを上げるほど、脳は活性化する。
集中していればミスは少ない。
時間を計ればムダに気づく事ができる。
集中力は長く続かない。休む事で充電される。
時間が読めるからリラックスできる。

『やる気と集中力の高め方』 東京大学　医学博士　森田敏宏著より抜粋

Copyrights@2013_Strategic Staff Services

『もの作り』と『システム作り』の相違



設計仕様を図面上のみの検討で、十分でしょうか？

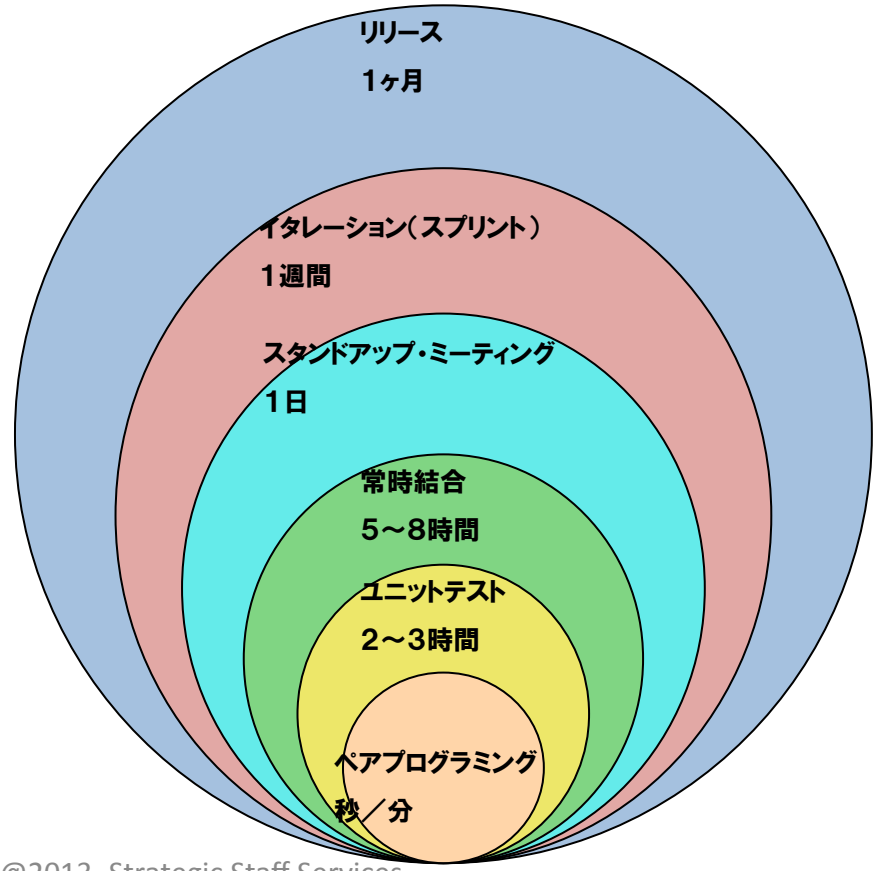
擦り合わせ、微調整が必要ではありませんか？（誰が、何時、どの様に作業できますか？）

Copyrights@2013_Strategic Staff Services

品質の向上のためのプラクティス

- ペア・プログラミング
- 継続的インテグレーション
- リファクタリング

多重のフィードバック・ループによる品質のチェック



Copyrights@2013_Strategic Staff Services

利用者（ユーザー）から見たアジャイル開発のメリット

顧客（ユーザー）から見てこのアジャイル開発手法はどの様なメリットを享受できるのでしょうか？もうシステム屋の言う事は信用しないという経営者の方も多いのではないでしょうか？そうです従来の主流であるウォーターフォール型開発では米国の調査でも計画期間内、計画予算内でプロジェクトが完了した成功率は僅か１６％です。

こうなりますとシステム開発プロジェクトは失敗して当たり前と言う評価を甘んじて受けざるを得ません。アジャイル開発では、この様な失敗を回避可能です。と言いますのも、アジャイル開発とは現有リソース（資源：人、金、時間）を前提に計画を立て実行する手法だからです。

- プロジェクト進捗の見える化

  完了した働くプログラム本数（実現した機能の数）で管理

- 製作している機能の見える化

  ユーザー用語での機能定義（理解できる言葉での設計確認）

  働くプログラムを稼動させて確認

- 絶え間ない擦り合わせ技術での品質向上

  品質とは、要求品質、設計品質、実装品質、検査（テスト）品質

- 優先、重要機能（システム）の早期実現

  システムを構成する重要機能から優先して開発

- 開発プロジェクト期間中での柔軟な変更対応

  プログラム製作期間（イタレーション）に入る前までは、変更自由

- ユーザーが参加すべき作業の見える化

Copyrights@2013_Strategic Staff Services

経験事例の報告（エンジニアの声）

エンジニアAさん： アジャイル初体験、ウォーターフォール開発経験約15年（主任クラス）37才
＞情報処理技術者　1種　＞UMTP　L1
＞業務SE（物流／生産管理／公共サービス）　8年　＞ホスト汎用機テクニカルSE　7年
＞今回　．NETは初めての経験　＞JAVA（web）　3年
＞PL／SQL　2年　＞VB．NET　（2週間：本アジャイル開発で初）
＞COBOL、C＋＋　etc（細かい開発は多数）

アジャイル開発の効果： コーディングの生産性は変わらないが、開発作業内で手戻りが無く、結果的に早く完了できる。

体験した感想： とにかく頭が疲れる。集中する。

・品質が高くなる。
・技術的な問題や、方式で悩む時間が少ないので効率は良い。
・気を抜く暇がないので、稼働率は高い
・二人でやっているので、生産性は倍まではいかない。ただし品質が高いので、改修やテスト時の修正工数は少なくなる
・ペアプロ／クロスファンクションにより　ソースコードレベルで情報を共有するため、自然に　可読性／ロジックのシンプルさが感じられる実装となる。
・随時に動かしながら機能拡張をするため、潜在バグ／デグレードのリスクは低い。
・実装が不慣れな要員がいても、ペアの組み合わせにより品質の高い実装が可能となる。
・作業の完了が、視覚的に理解できる。実装の成果がすぐに見れる。
・スタンドアップミーティング／振り返り／タスクの割り振りによりメンバー全員が全体の作業を見渡せる。司会を持ち回りすることにより参加意識が強調される。
人に見られているのに、適当な（動けばいい）コーディングはできない。
・悩んでいる時間が少ない。（随時相談／調査）
・具体的な目標を随時持つことができる。
・タスク担当を明確にすることにより、責任範囲の当事者意識を持つことができる。

Copyrights@2013_Strategic Staff Services

経験事例の報告（エンジニアの声）

エンジニアBさん：　アジャイル初体験、ウォーターフォール開発経験約5年　28才
＞ Oracle Master Bronze 10g　　＞ VB.NET：1年
＞ HTML、JavaScript、CSSなどWeb関係：1年（随時）　＞ PHP：1年
＞ VB6：3年　　＞ PL/SQL：1年
＞ XML：1ヶ月

アジャイル開発の効果：　生産性が上がった（体感）。

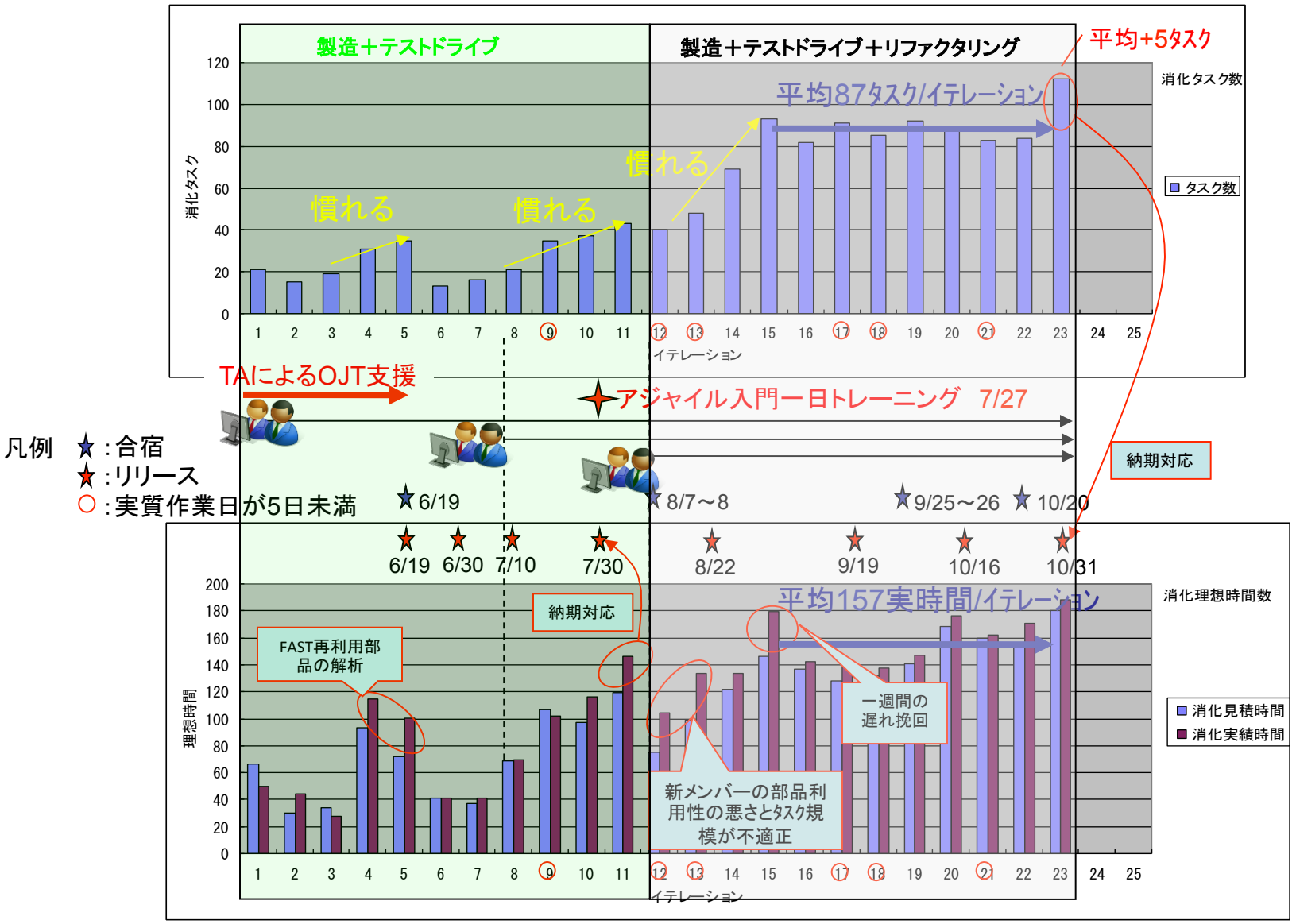
体験した感想：　個人のコミュニケーション能力が高く問われる
・二人で作業を行っているため、単純ミスも少なく精度が高い。
・思った以上に疲れる。気を抜く暇がない。

エンジニアCさん：　アジャイル初体験、ウォーターフォール開発経験約4年27才
＞ 初級アドミニストレータ
＞ 詳細設計・PG・/テスト（物流／在庫管理／Web）　4年
＞VB．2年　　＞ JAVA（web）　3年
＞ PL／SQL　三ヶ月

アジャイル感想：
・一人で開発をしている時は、技術的にわからないことがあるとネットや本等で調べて解決し、作業を進めていくが、ペア・プロの場合横で他の人が見ているので、気軽に調べ物がしにくい。
（まだメンバーに慣れていないため、気軽に質問ができない）
・仕様をよく知っている人とペアを組むと、プログラミングの道筋を立てて開発ができるので、良いと思う。
・反対にある程度わかっている人が作業を行い、ほとんどわかっていない人が横で見ている場合、作業者はどこまで説明しながら進めればいいのかわからない。
・ペア・プロだと常に他の人と一緒に開発を行うので、緊張感が保てる。その反面、気を抜くことができないので、一人で作業をするよりもかなり疲労度が高い。
・タスク毎に見積もり時間を出して作業を行うことと、プログラム1本の見積もり時間しか出ていない状態で作業を行うことと比べると、タスク毎の方が作業を進めやすい。
（プログラム1本の見積もりの場合、ペース配分が上手くいかないことがある）
・ウォーターフォール型の開発に慣れているので、要件定義からいきなりプログラミングに入ることにまだ戸惑いを感じる。外部設計書、内部設計書があった方が開発がしやすい。
・ペア・プロに慣れてきたので、作業進捗が早くなってきた。
・常に隣に他人がいる環境でずっと開発を行うのは疲労度が高く、辛い時もある。

Copyrights@2013_Strategic Staff Services

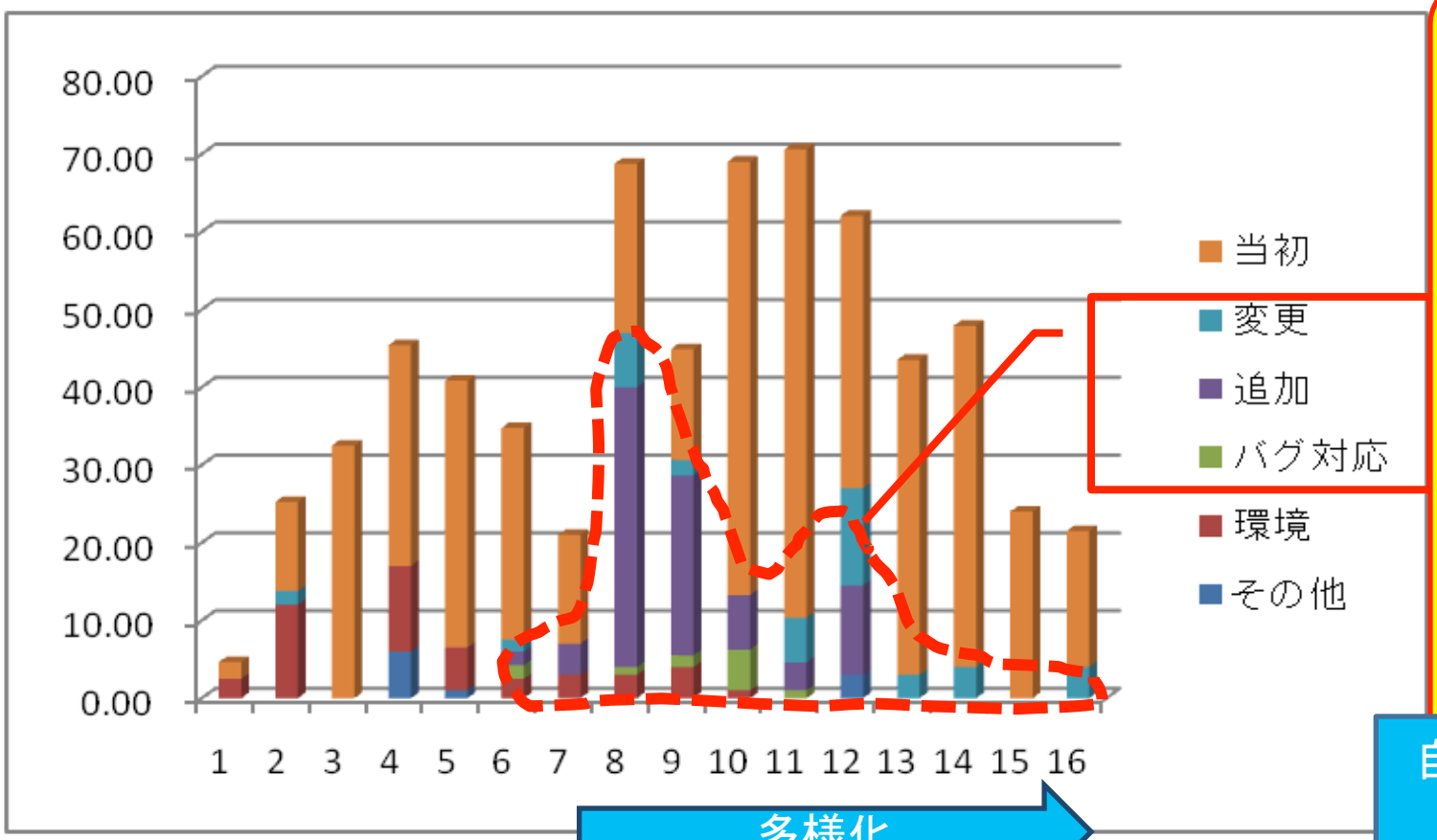
スクラムの管理者（スクラムマスターの仕事）
スクラム・マスターは何を見るのか？



テストドライブは
全体の25%

Copyrights@2013_Strategic Staff Services

毎週の作業要因別実績時間



納品後のバグは二件のみ
全て機器調整に対応する為

自工程完結の
結果

Copyrights@2013_Strategic Staff Services

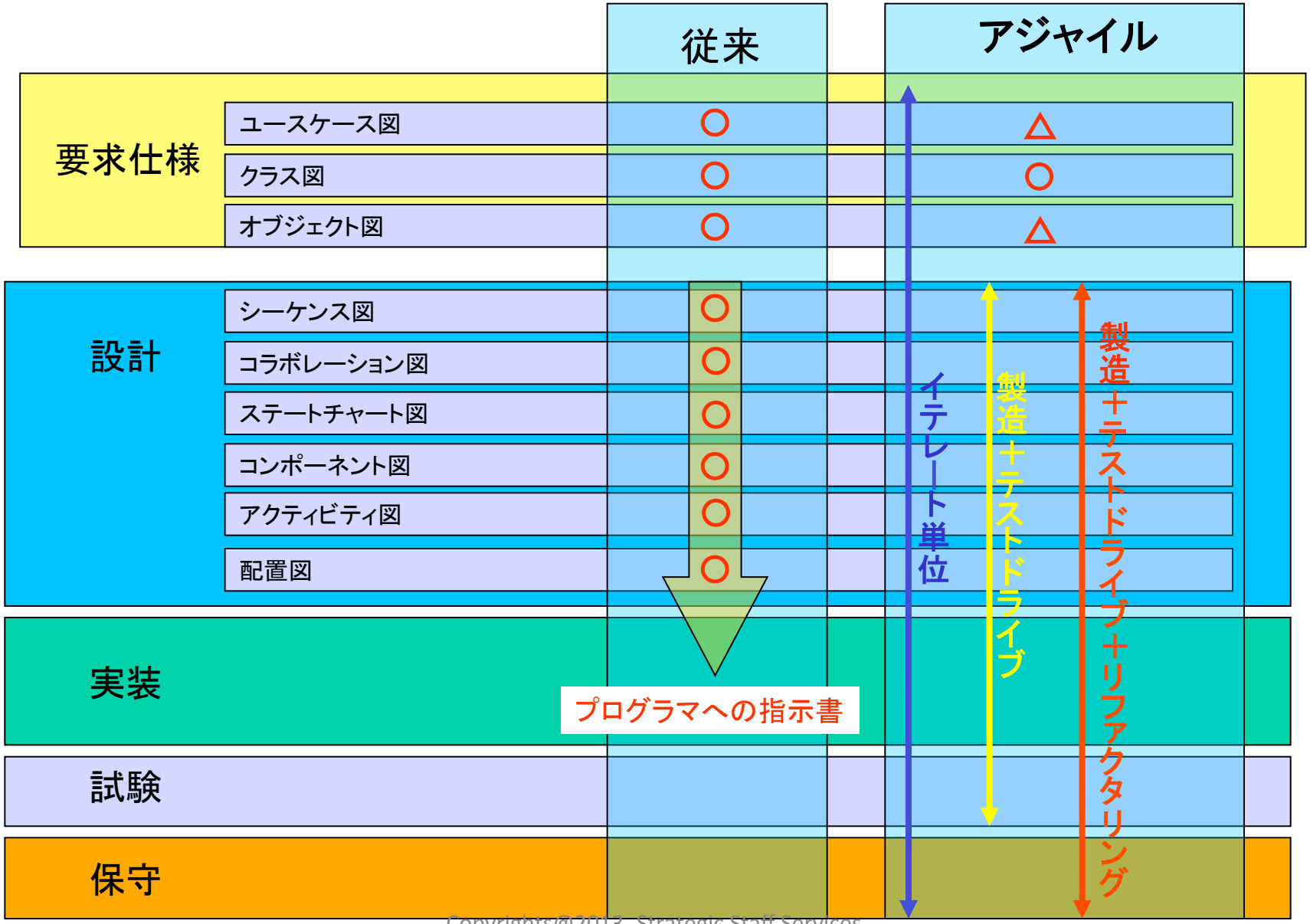
タスクの粒度と平準化（実績）

	見積総時間	実績総時間	タスク総数	見積粒度（H)	実績粒度（H)
第１週～第７週　イテレーション **ユーザー操作系**が主な機能１回目	197.8	211.9	134	1.476	1.581
第８週～第１６週　イテレーション **機械制御系**が主な機能　２回目	418.4	452.2	295	1.418	1.533

全く異なった機能実装イテレーションでも同じ粒度で平準化している

Copyrights@2013_Strategic Staff Services

ドキュメンテーション（UML）



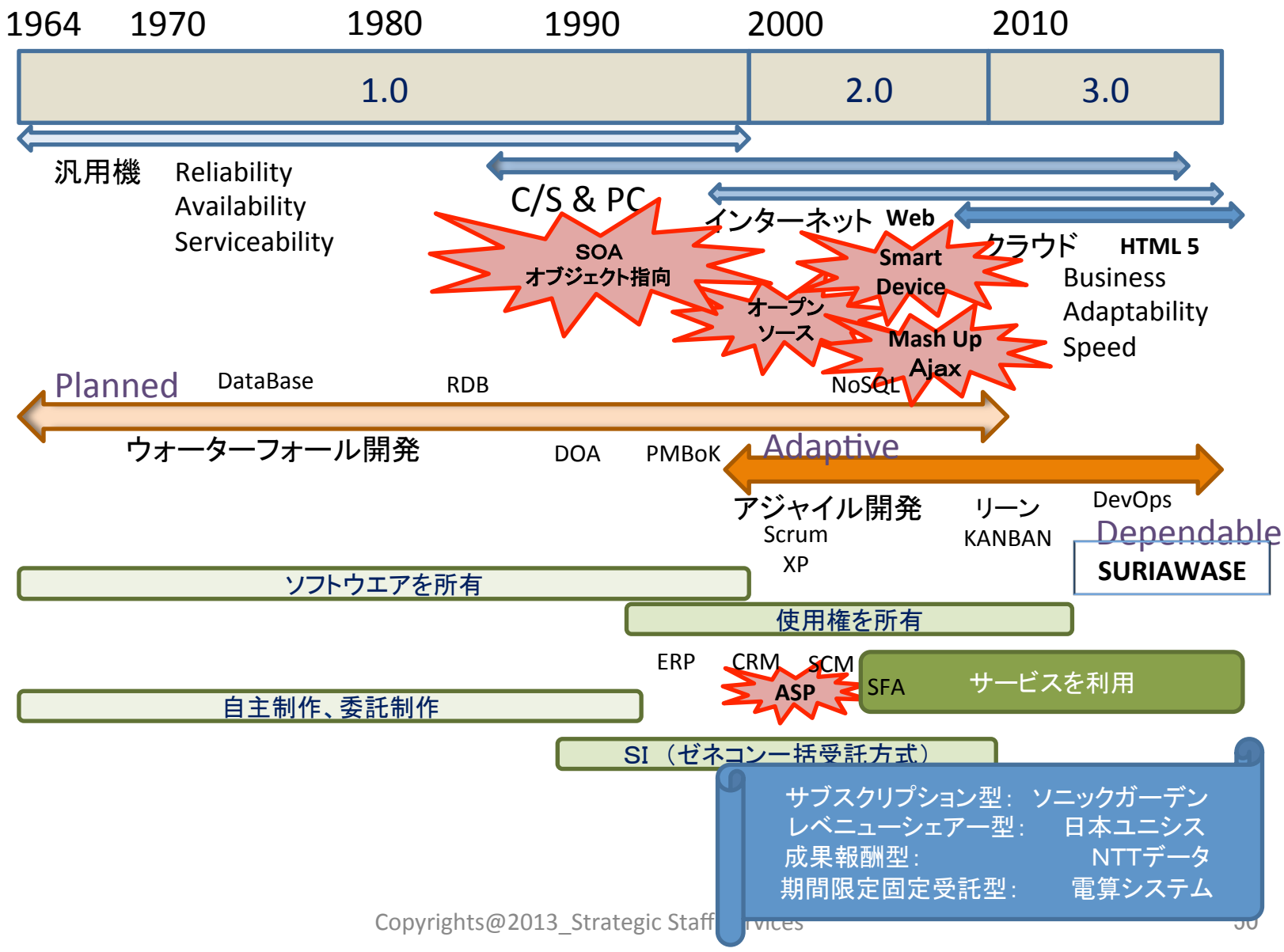
Copyrights@2013_Strategic Staff Services

第二部

アジャイル開発でシステム開発ビジネスはどうなる？

Copyrights@2013_Strategic Staff Services

システム開発の変遷



Copyrights@2013_Strategic Staff Services

SI 3.0の世界

- **ウォーターフォール の経営学**

量の経済
売上＝原価＋(期待)利益
売上は目指すもの、原価は既定値(原価積上げ)、利益で調整

- 大量生産(ベルトコンベヤーライン＋単能専任化)

人間の機械化(ロボット化)
管理視点:統計的品質と稼働率
プログラム化
大規模で採算を取る

フォード
家電業界

- **アジャイル の経営学**

採算の経済
利益＝売上ー原価
利益は目指すもの、売上は既定値(お客様の懐)、原価で調整

- 一個流しの流れ生産(多能化＋作業責任、U字ライン＆セル生産)

人間性の尊重
管理視点:お客様満足と可働率(べきどうりつ)＝流れ(整流化)
人間の思考力
小規模でも採算が取れる

トヨタ
キヤノン

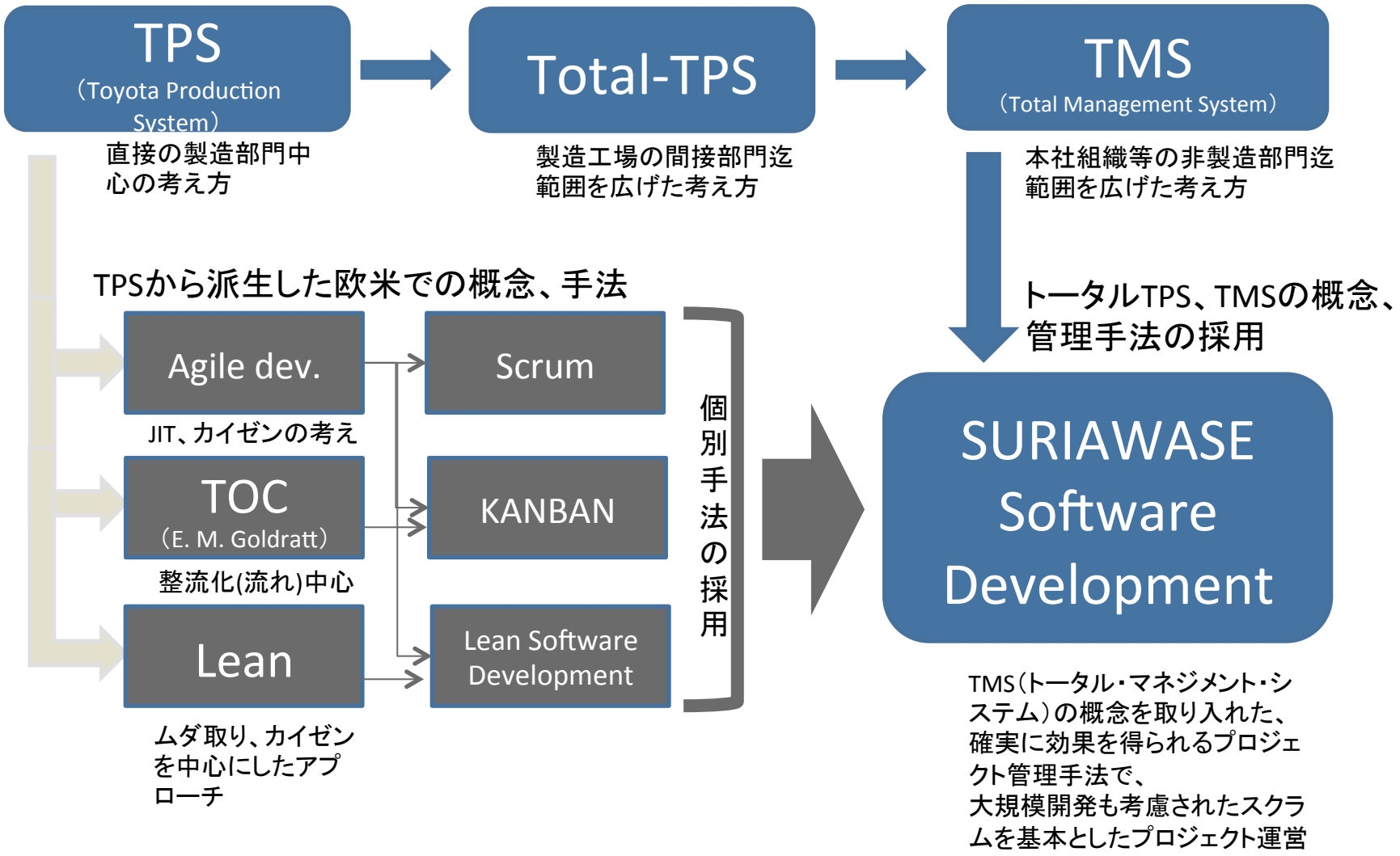
Copyrights@2013_Strategic Staff Services

IT企業にとっての効果

- お客様満足度の大幅な向上
  - 開発プロジェクト進捗の見える化
  - 高品質なプログラム
  - 短期間でのシステムの安全稼動
- プロジェクト失敗のリスクの大幅な軽減
  - 確認しながらの確実なシステム構築方法
  - スケジュール管理の容易さ(目で見る管理)
  - 品質の向上に拠る手戻り作業の撲滅
- 生産性の向上＝コスト･ダウン
  - プロジェクト運営の柔軟性
- 従業員のモラル向上
  - 残業、休出ゼロでの納期遵守
  - 提案型でエンジニア魂の高揚

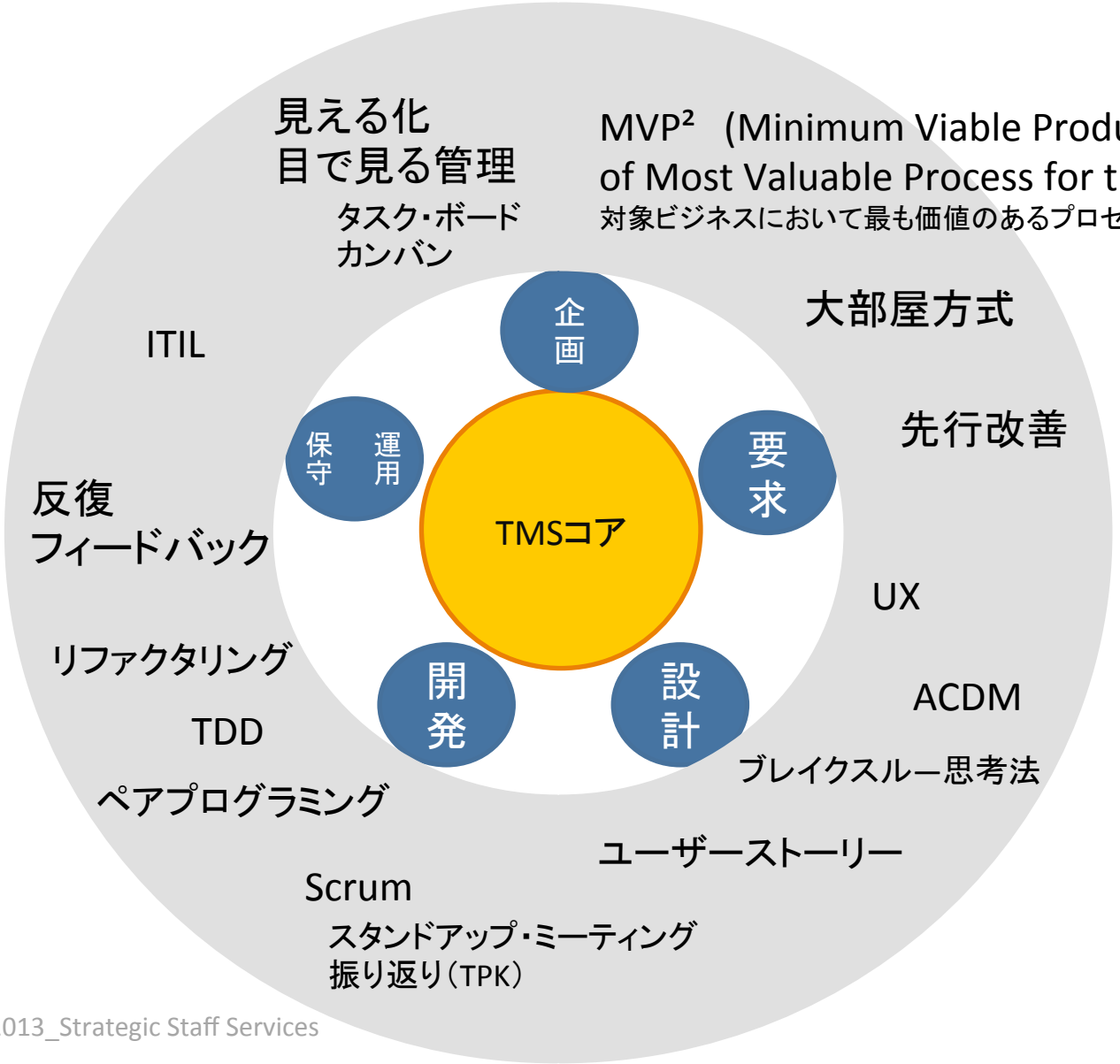
Copyrights@2013_Strategic Staff Services

すりあわせソフトウエア開発の位置付



Copyrights@2013_Strategic Staff Services

すりあわせ開発で活用する手法



Copyrights@2013_Strategic Staff Services

ケース・スタディー

プロジェクト概要：

- 年商30億円（年率約10%成長）、JSDAQ上場企業（全国チェーン展開のホームリノベーション）
- 基幹業務システムの再構築（リプレース）
- 約半年掛けて外部コンサルタント（ITIL）を交えてRFPを準備し、7社での公開入札コンペ
  3社が辞退し、4社での入札（内1社は現行システム構築）

- 新任のシステム管理部長が半年前に入社、（元大手インターネット証券企業システム管理者で前職はITコンサルタント）　PMBOKに精通し、細かな指示、監督を行う

プロジェクトの特長：

◆ ゼネコン一括受託方式に対抗して設計事務所＋工務店方式での受託開発
◆ アジャイル開発を前面に出して提案　（すりあわせソフトウェア開発の実践）
◆ リーンスタートアップ（E・リース）の実践

結果（評価）：

□ 予定通りでの新システムの稼働
□ 非常に高いお客様の満足度（評価）で、
  ①プロジェクト実施途中での予算の増額（+3,000万円）
  ②プロジェクト実施途中に次期開発案件受託決定（約6,000万円）
  ③プロジェクト実施途中に運用サービスの受託決定
□ プロジェクト損益は営業利益ベースで30%~40%

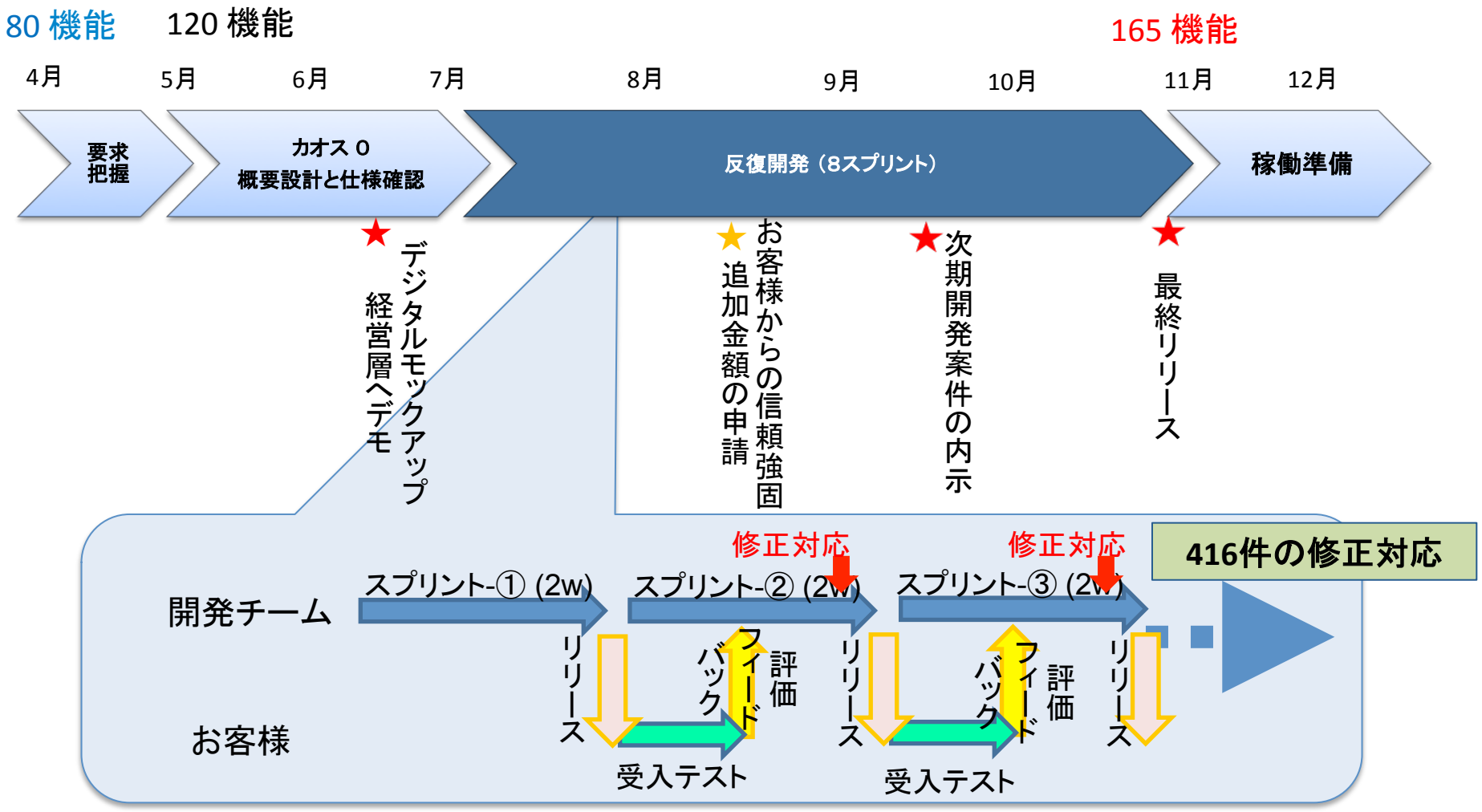
Copyrights@2013_Strategic Staff Services

プロジェクト成功の理由

- RFP記載の280に上る実装機能要求を期間を限定して80機能という制限を提案し、要求の重要度＆優先度、根拠、ビジネス価値、リスク、条件等の要求属性の明確化でリーンスタートアップのアプローチ
- 6ヶ月間で80機能の実装と言う開発チーム（10名）の明確なコミットメント
- カオス0（仕様確認期間）でのユーザー要求と実装方法の仮説検証をデモで確認（デジタル・モックアップ）
- プロジェクト期間を通して、開発チームの作業（進捗）の透明性を確保し、ユーザーからの強力な信頼感を獲得（チームの行動特性）
- 実装したコード（現物）を反復的（定期的）にユーザーにリリースし、『すりあわせ』による要求確認と変更への開発チームの素早い対処（修正実施）
- 徹底的にムダなドキュメンテーションを避けて開発（コーディング）作業に専念、しかしお客様のニーズ（要求）に応えたドキュメンテーション（データ）のJITでの提供
- 自律した開発チーム・オペレーション（毎週実施された現地現物での振り返り）

Copyrights@2013_Strategic Staff Services

プロジェクトの期間と運用



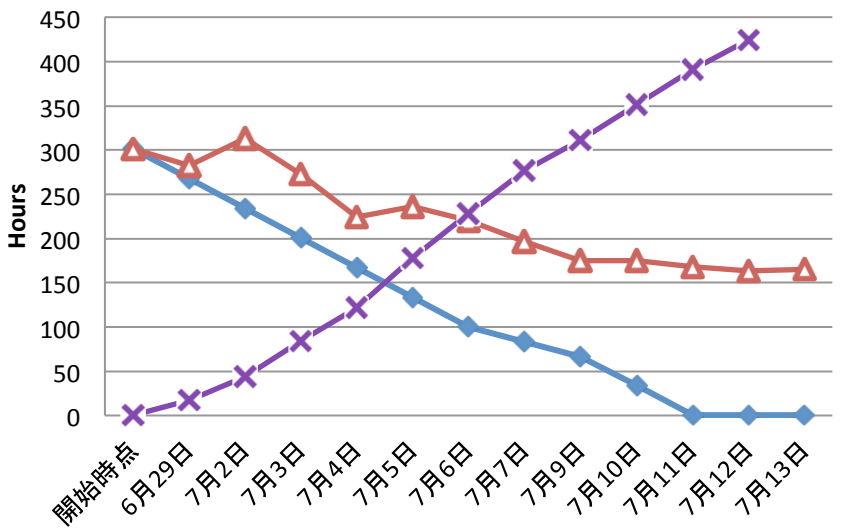
Data from DSK

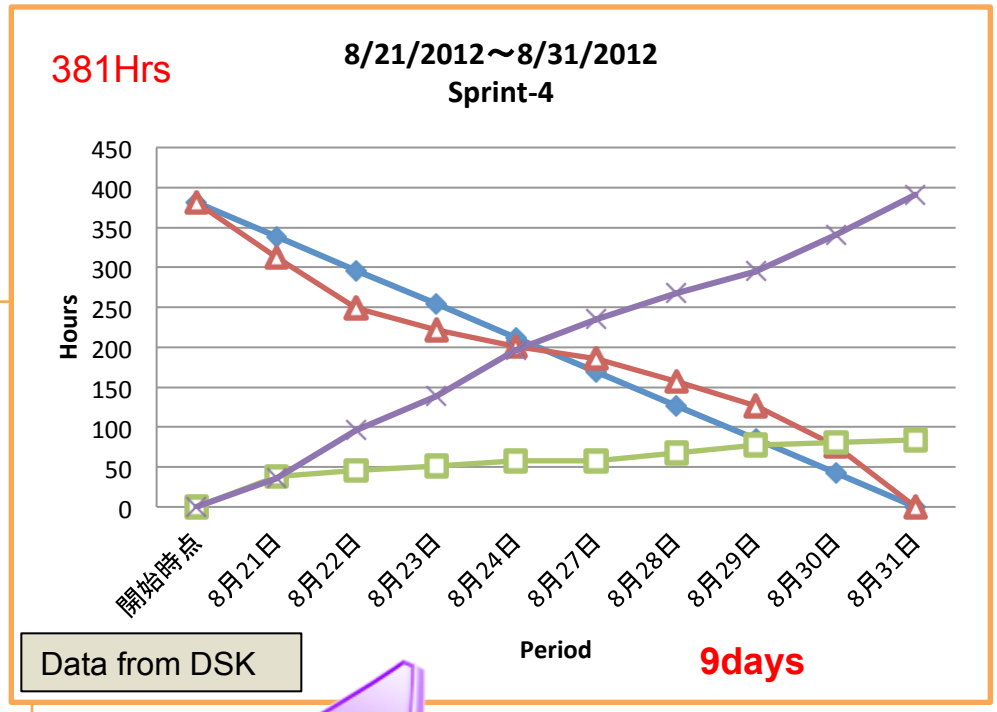
Copyrights@2013_Strategic Staff Services
Copyrights@2013_Strategic Staff Services

チームの成長 （カイゼン）

同一プロジェクト、同一のチーム、同一の手法（プラクティス）で実施しても
何故、生産性が変わるのか？

△ △ △ :Actual
◆ ◆ ◆ :Plan
× × × :Actual (Accumulation)
□ □ □ :Correspond to Feedback





生産性の向上
6週間で168.7% 向上

Copyrights@2013_Strategic Staff Services

Agile Development Session Program

開始時間	終了時間	正味時間(分)	講演者	会社名	組織・役職	題名
13:00	13:05	5	戸田孝一郎	(社)TPS検定協会	理事	オープニング
13:05	13:35	30	和田憲明	富士通株式会社	SI技術本部　システム技術統括部	『富士通グループの取組みとペアプロの考察』 富士通グループの取組みについて傾向と事例をご紹介した後「ペアプロとTPS」について私の経験談を話します
13:35	13:55	20	芦澤 寛樹	富士通株式会社	行政システム事業本部 第一ソリューション統括部　第二ソリューション部	『スクラムの"ス"くらいをやってみた！！～振り返り編～』 途中でウォーターフォールからアジャイルに切り替えたプロジェクトが終了しました。開発者、お客様にそれぞれヒヤリングして振り返りし、良かった点、悪かった点、次回に向けたTryなどをご紹介します。
13:55	14:00	5	セットアップ			
14:00	14:30	30	山田静香	株式会社オージス総研	ソリューション開発本部	『中部電力様のアジャイルへの挑戦～プラクティスを超えて～ 』 理想のチームとプロダクトオーナーとは？ アジャイル初挑戦のメンバーが集まったプロジェクトでの、アジャイル文化浸透の経緯を報告します
14:30	14:40	10	休憩			
14:40	15:10	30	英　繁雄 長瀬嘉秀	株式会社日立ソリューションズ 株式会社テクノロジックアート	技術開発本部　生産技術センタ担当部長 代表取締役	『ハイブリッドアジャイルの実践』 日本の受託開発に合ったハイブリッドアジャイルの適用について、ポイントと事例をご紹介いたします
15:10	15:15	5	セットアップ			
15:15	15:45	30	小畑陽一	株式会社エムティーアイ	Package Solution事業部長	『経営をアジャイル化し、事業成果を上げる』 アジャイル開発手法を経営成果へ昇華させる！マネジメントシステム全体を改善してきた軌跡を振り返える
15:45	15:55	20	休憩＆セットアップ			
15:55	16:55	60	松島桂樹	松島桂樹研究室　代表	(社)クラウドサービス推進機構　理事長	パネルディスカッションのモデレーター 『アジャイル開発は、人つくり、チーム作り』 パネラーは発表者各位
			和田憲明	富士通株式会社		
			芦澤 寛樹	富士通株式会社		
			山田静香	株式会社オージス総研		
			英　繁雄	株式会社日立ソリューションズ		
			長瀬嘉秀	株式会社テクノロジックアート		
			小畑陽一	株式会社エムティーアイ		
16:55	17:00	5	戸田孝一郎 (敬称略)	(社)TPS検定協会	理事	クロージング＆案内

http://training.softopia.or.jp/event/20140704/

ご清聴ありがとうございました。

戸田　孝一郎

米国スクラムアライアンス認定の公認スクラムマスター

（社）TPS検定協会　理事

http://www.Ask3S.com

株式会社　戦略スタッフ・サービス

（本社）〒100-0004 東京都千代田区大手町1-7-2 東京サンケイビル27F 電話：03-3242-6282 FAX：03-3242-6283

お問い合わせはメール（lktoda@ask3s.net）にてお願いいたします。

Copyrights@2013_Strategic Staff Services

講演･コラム　等の実績

- 2009年
  - SODEC 専門セミナー　講演
  - SODEC　IBM社ブースにてアジャイル開発のセッションを担当
  - 日本IBM Rational Software Conference 2009　講演
  - 日経BP　ソフトウエア技術者人材育成シンポジウム　講演
  - マイクロソフトのアジャイル開発支援サイトへの寄稿　（10回）（2009年～2010年）
  - マイクロソフト　最新．NETシステム開発の実践セミナー　講演
- 2010年
  - 日本IBM　Innovate 2010　講演　（アジャイル･トラックの企画支援）
  - ソフトウェア品質管理シンポジウム　2010　ワークショップ提供
  - 日経BP　X-over Development Conference 2010　講演
  - Agile フォーラム　in　岐阜　2010　講演
- 2011年
  - Agile Japan 2011　講演
  - Agile Conference Tokyo 講演
  - ソフトウェア品質管理シンポジウム　2011　講演とワークショップ提供
- 2012年
  - Agile Japan 2012　講演
  - 電子情報産業協会　JEITA ソフトウェアエンジニアリング技術ワークショップ2012　講演
- 2013年
  - SODEC 専門セミナー　講演
  - IBM　Innovate2013　The Technical Summit　（Orlando, FL）講演
  - Agile Conference Tokyo 講演

Copyrights@2013_Strategic Staff Services

弊社アジャイル開発の足跡

- ～2007年3月：
  岐阜県産業振興・再開発プロジェクトに参画
- 2007年3月～：
  岐阜県下のIT企業様と生産技術力向上を目指したNPO団体（BCA）の設立（発起人）に関わり、同会員企業となる。
  現在；弊社社長戸田孝一郎が理事長
- 2007年4月～2008年3月
  同NPOの初のアジャイル開発プロジェクトをPM管理者（スクラムマスター）として実施
- 2007年6月～2008年3月
  同NPOのアジャイル研修プログラムを開発
- 2008年4月～
  Scrum＆XPでのアジャイル開発に関するプロジェクト管理の指導育成
  アジャイル・チーム育成コーチング　と　アジャイルトレーニング　の提供
- 2010年8月～
  Scrumのプロセスにユーザーストーリー（マイク・コーン提唱）を組み合わせる
- 2011年10月～
  UX（ユーザーエクスペリエンス）とアーキテクチャ設計（ACDM)を組み合わせる
- 2012年1月～
  データベース・リファクタリングを検証
- 2012年8月～
  トヨタのTMS思想を全面的に採用し、Scrumプロセスの運用に組み込む
- 2013年1月～
  豊田マネジメント研究所（TMSの専門指導）との協業でITマネジメント塾（カイゼン塾）を指導
  すりあわせ開発を提唱
- 2014年4月～
  社団法人　TPS検定協会　理事、 認定TMS講師（TCP）

Copyrights@2013_Strategic Staff Services

アジャイル開発の指導・支援実績

- 会計パッケージ開発
  - アジャイルによるパッケージ機能拡張
  - 発注会社：受注会社間の遠隔プロジェクト遂行
- 生産管理システム
  - ハード＆ソフトの並列同時開発
- マッシュアップ＆アジャイルによる地域ICTの実現
  - 買い物弱者支援システム
    - 総務省の標準モデルに採択（2011年7月）
- 保険金融系情報企業のアジャイル化支援
  - 複数のチームを並行支援指導
  - Scrum Master育成支援
- 携帯向けコンテンツ企業のアジャイル化支援
  - スマートフォンへの戦略的移行についてもアジャイル化支援
  - 複数事業部：複数チームの平行支援指導
  - Scrum Master育成支援
- アジャイル手法を応用した開発系以外での支援指導
  - 通信会社営業企画部門
  - ITIL適用部門での現場ボトムアップ（プロセス改善・ムダとり）
- すりあわせ開発の実証
  - 中堅企業の基幹系システム再構築プロジェクトに応札し、アジャイル開発で提案、お客様満足最大化を実証 （スクラムマスター支援）
- TMSでのIT職場活性化指導
  - 保険金融系企業、コンテンツ・プロバイダー企業、大手IT企業での指導
- 日本経営品質賞（能率協会）とTMSで協業
  - トップダウンとボトムアップの連動アプローチ（DJEプログラム提供）

Copyrights@2013_Strategic Staff Services