ソフトウェアの品質と生産性を向上させる鍵は何か

2005年 12月16日

富士通総研　経済研究所　主任研究員

（早稲田大学　国際情報通信研究センター　客員教授）
（国際大学 グローバル・コミュニケーション・センター 主幹研究員）

前　川　　徹

今日、お話したいこと

- 「ウォーターフォール・モデルは間違いだ！」
- 個人の能力をより重視すること
  優秀な技術者不足と
  恒常的な残業の悪循環を断ち切る
- ソフトウェア企業（産業）を育てるのはユーザ
  「店が客を育て、客が店を育てる」
- ソフトウェア産業におけるパラダイムシフト
  「規律」重視から「アジャイル」重視へ

0

背景あるいは問題意識

「自動車はソフトウェアによって制御されている」

- 2005年10月、トヨタは2004―05年モデルの「プリウス」約16万台のエンジン起動ソフトに不具合があり無償で修正すると発表
- 2004年5月、ダイムラー・クライスラーが、Eクラスのセダンなど68万台のリコールを発表。原因は、電子制御ブレーキ『センソトロニック』の不具合
- 高速走行中にエンストが起こる不具合があったケース　(BMW)
- 突然ギアがバックに入るという不具合があったケース　(ジャガー)

世界はソフトウェアに依存している

- 多くの社会インフラがコンピュータによって管理･制御
- 身の回りの機器もソフトウェアを内蔵したものが増加
- 二重化などの対策はネットワークやハードウェアの障害には有効だが、ソフトウェアの不具合による障害を回避することは困難

システムの不具合によるトラブル（1/2）

（金融機関）

- ＵＦＪ、公共料金など二重引き落としなど18万件（2002.1）
- みずほ、口座振替未了が250万件以上に（2002.4）
- 信金のＡＴＭ約2万台で信金以外のカード使えず（2002.5）
- ネット専業のジャパンネット銀行で全面ダウン（2003.5）
- 東京三菱など20行のカードが他行で使えず（2004.1）
- 全ての金融機関で他行カードが使えず（2004.1）

システムの不具合によるトラブル（2/2）

(交通機関、その他)

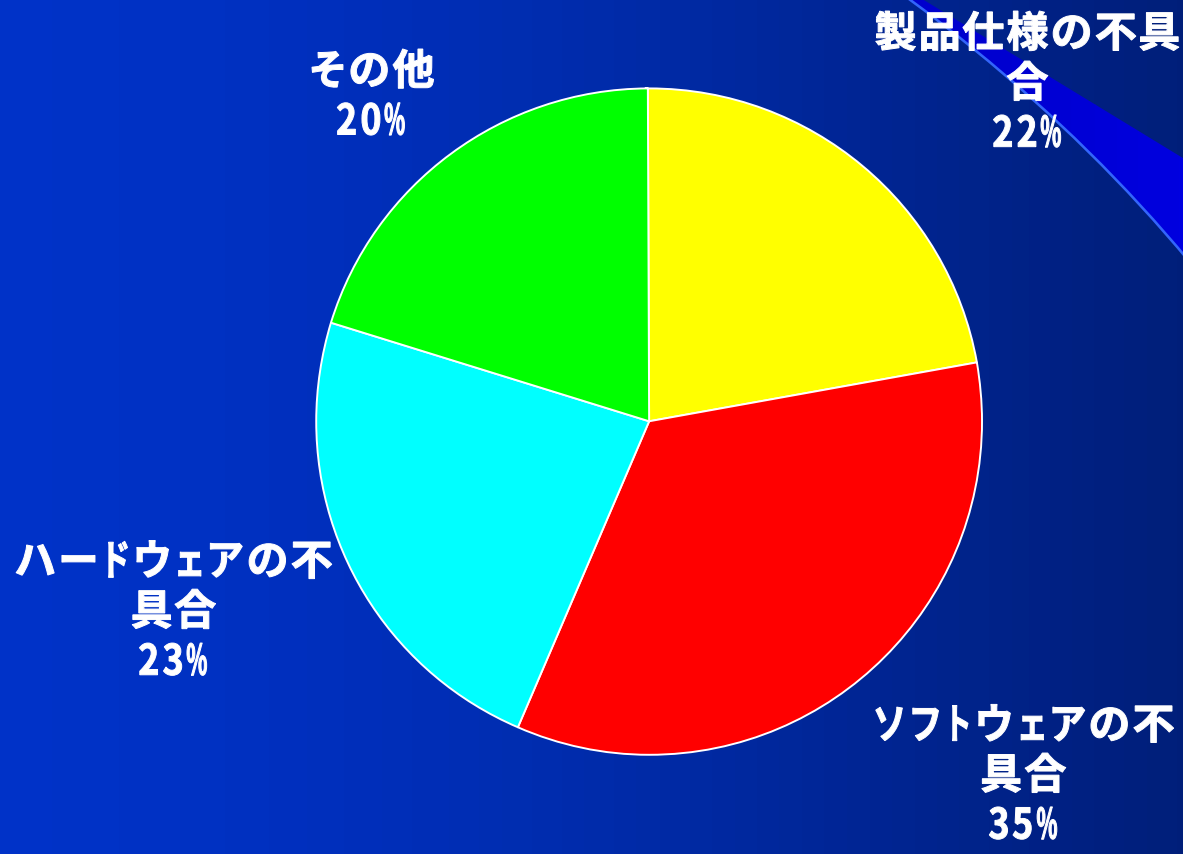
- 航空管制システムが障害、200便以上が欠航 (2003.3)
- ＪＲ西日本の運行管理システムに障害発生 (2003.8)
- 東京証券取引所でシステム障害、株取引に支障 (2003.10)

(組込みシステム)

- ＮＴＴドコモ、携帯電話を10万台以上回収 (2000.12)
- P503i 発売直後に販売停止、回収 (2001.2)
- So503i、個人データ流出の恐れで、43万台回収 (2001.6)
- ソニー、高級デジタルカメラを回収 (2003.9)

製品出荷後のトラブルの原因

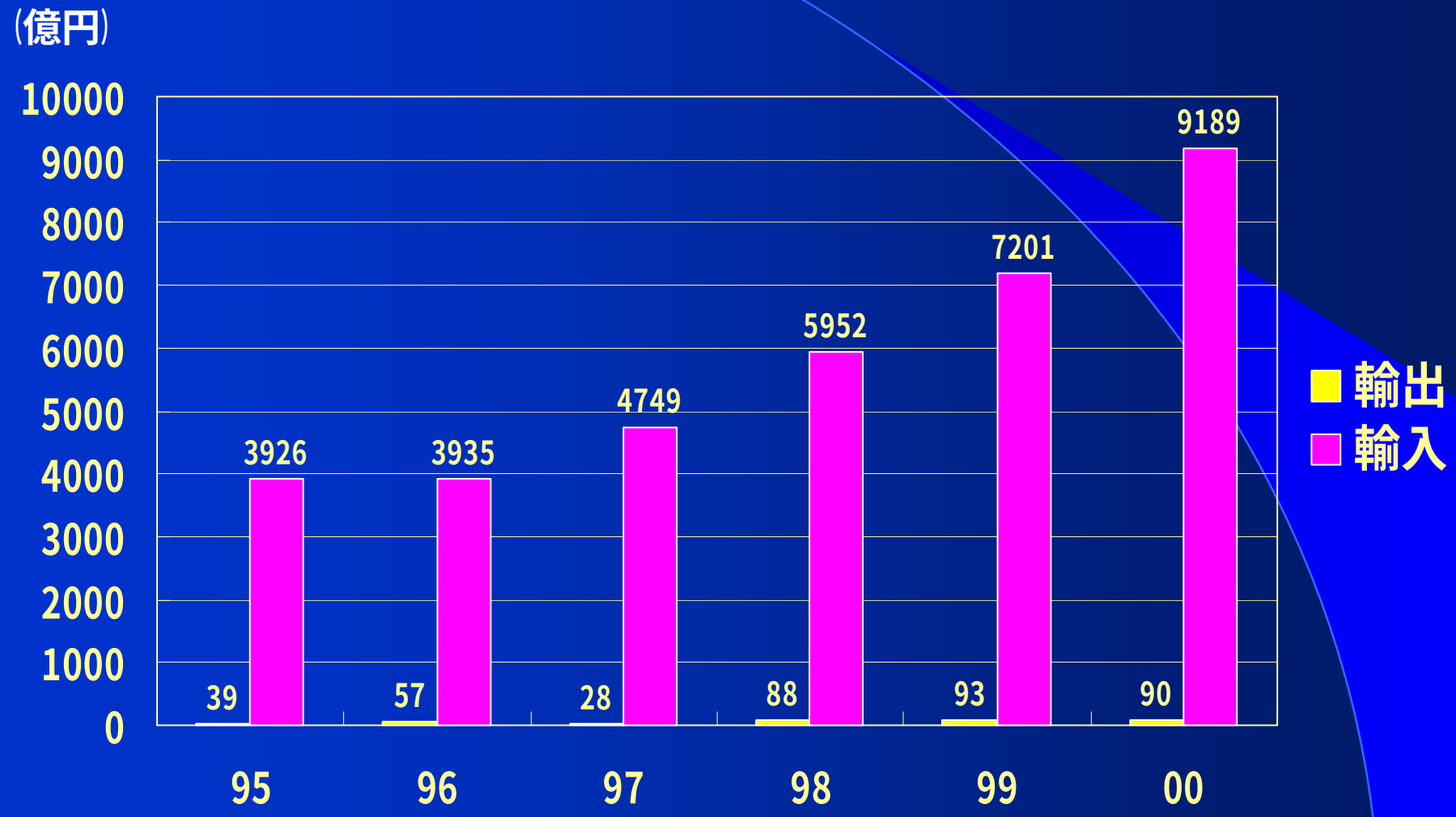
製品出荷後に生じた設計品質問題の主な原因の割合



（出典：「2005年版組込みソフトウェア産業実態調査報告書」
経済産業省商務情報政策局（2005）

日本のソフトウェアに国際競争力はない !!

日本のソフトウェア輸出額は輸入額の１%程度しかない

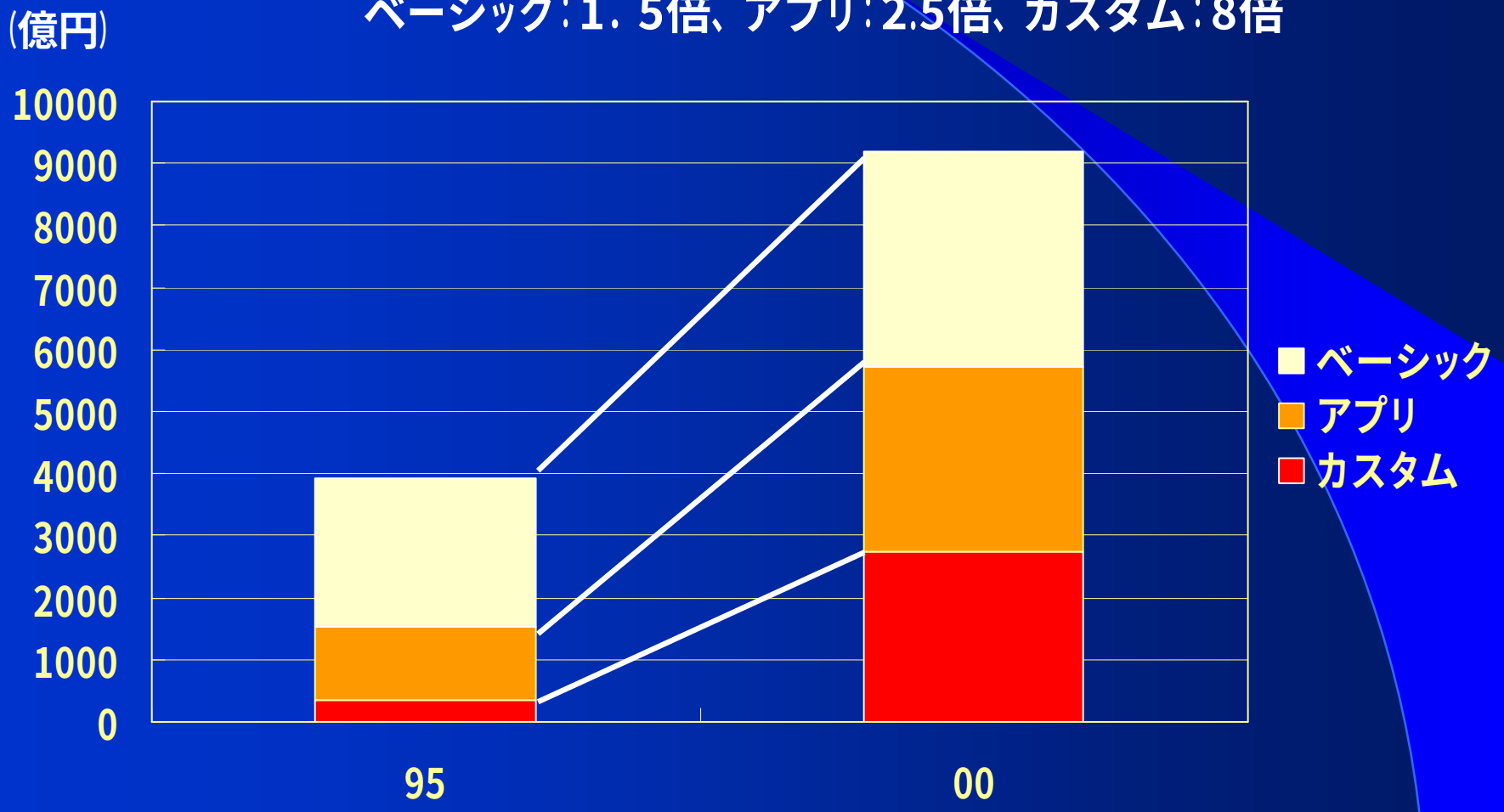


(出典:「ソフトウェア輸出入統計調査」電子情報技術産業協会 (2003)

輸入額が一番延びているのはカスタムだ！

種類別にみたソフトウェア輸入額

95→2000年の伸びは、
ベーシック：1．5倍、アプリ：2.5倍、カスタム：8倍



（出典：「ソフトウェア輸出入統計調査」電子情報技術産業協会（2003）

日本は米国やインドに比べて
ソフトウェアの生産性も質も高い！？？？

クスマノ教授の新調査

	日本	米国	インド	欧州他
プロジェクト数	27	31	24	22
コード行数	469	270	209	436
バグの数	.020	.400	.263	.225

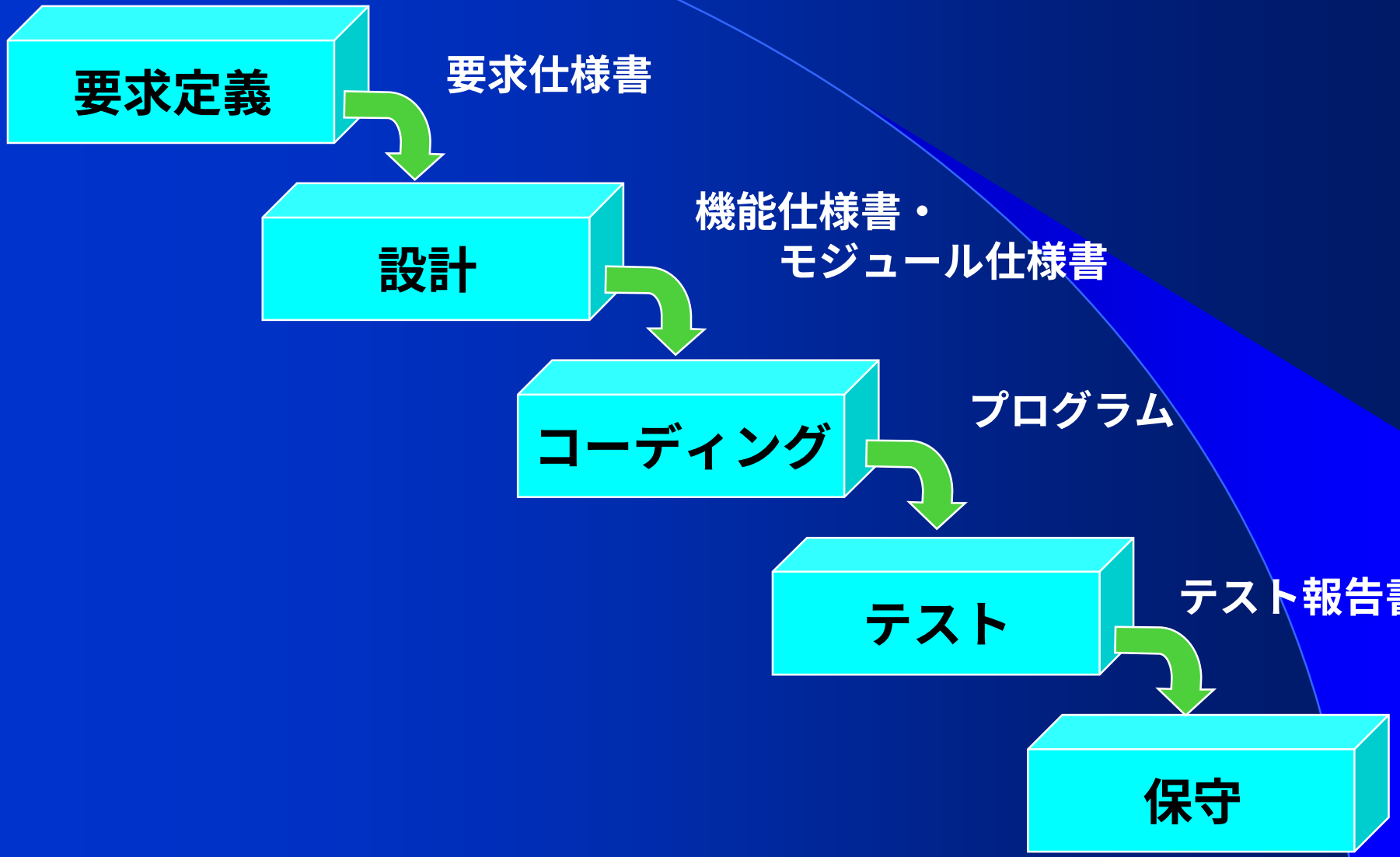
（注）１．コード行数は、一人のプログラマが1ヶ月に書くプログラムの行数
２．バグの数はリリース後12カ月に発見された1000行当たりの数

（出典：Cusumano,M., MacCormack, A., Kemere,C.F., Crandall,B. “Software Development Worldwide: The State of the Practice” IEEE Software Nov/Dec 2003, pp.28-34）

1

ウォーターフォールモデルは間違いだ！

ウォーターフォールモデル



（出典：大場充ほか『ソフトウェアプロセス　改善と組織学習』　ソフト・リサーチ・センター（2003））

「ウォーターフォールモデルは間違いだ！」

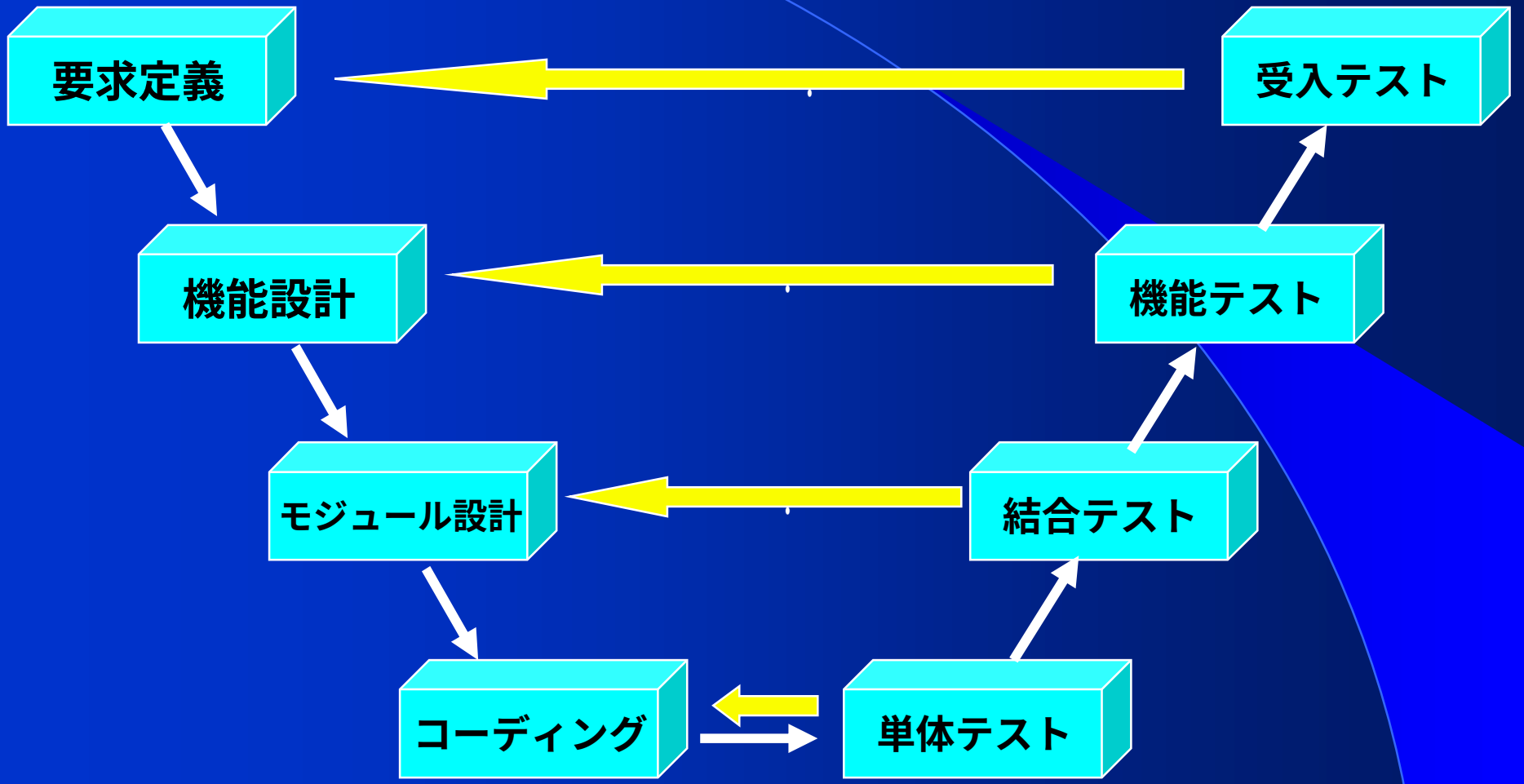
◆「ウォーターフォールモデルは、1975年にたいていの人が抱いていたソフトウェアプロジェクトについての考え方だった。だから、不幸にもDOD-STD-2167というすべての軍事ソフトウェアに関する国防総省の仕様書に記述され祭り上げられてしまった。そのために思慮深い専門家のほとんどが不適切さに気付き捨て去ってからもなお生き延びた。幸いなことに、国防総省もそれ以後ようやく気が付き始めたようだ。」

（フレデリック・P・ブルックス・Jr.『人月の神話』）

◆ すべての要件を把握し、その後にすべての設計、そしてその後にすべてのコーディングを行うことも可能ですが、こういったウォーターフォール・アプローチは、危険かつ問題をはらんだ、企業における風土病とも言えるものなのです。」

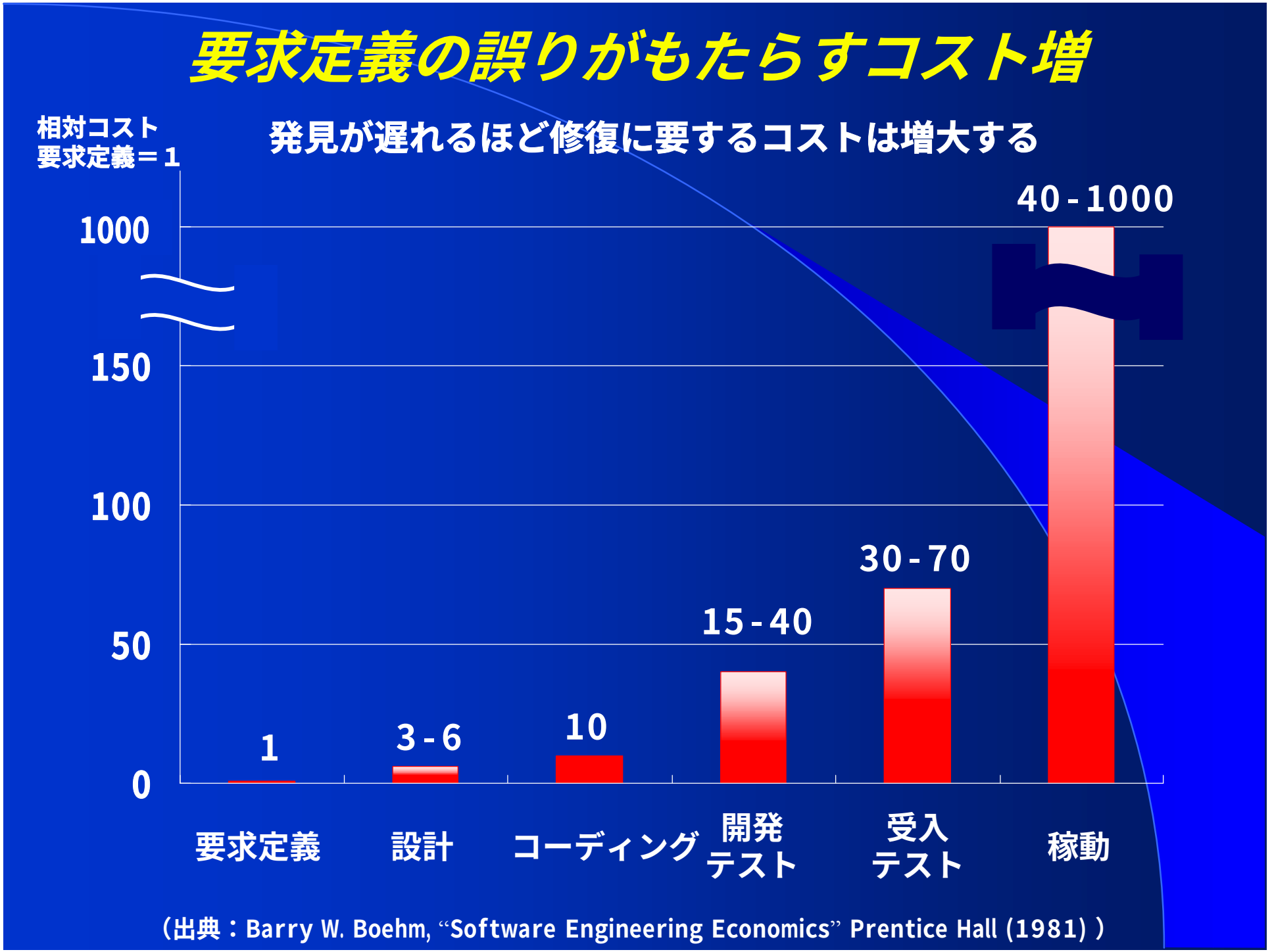
（ピート・マクブリーン『ソフトウェア職人気質』）

ウォーターフォールモデルの問題点



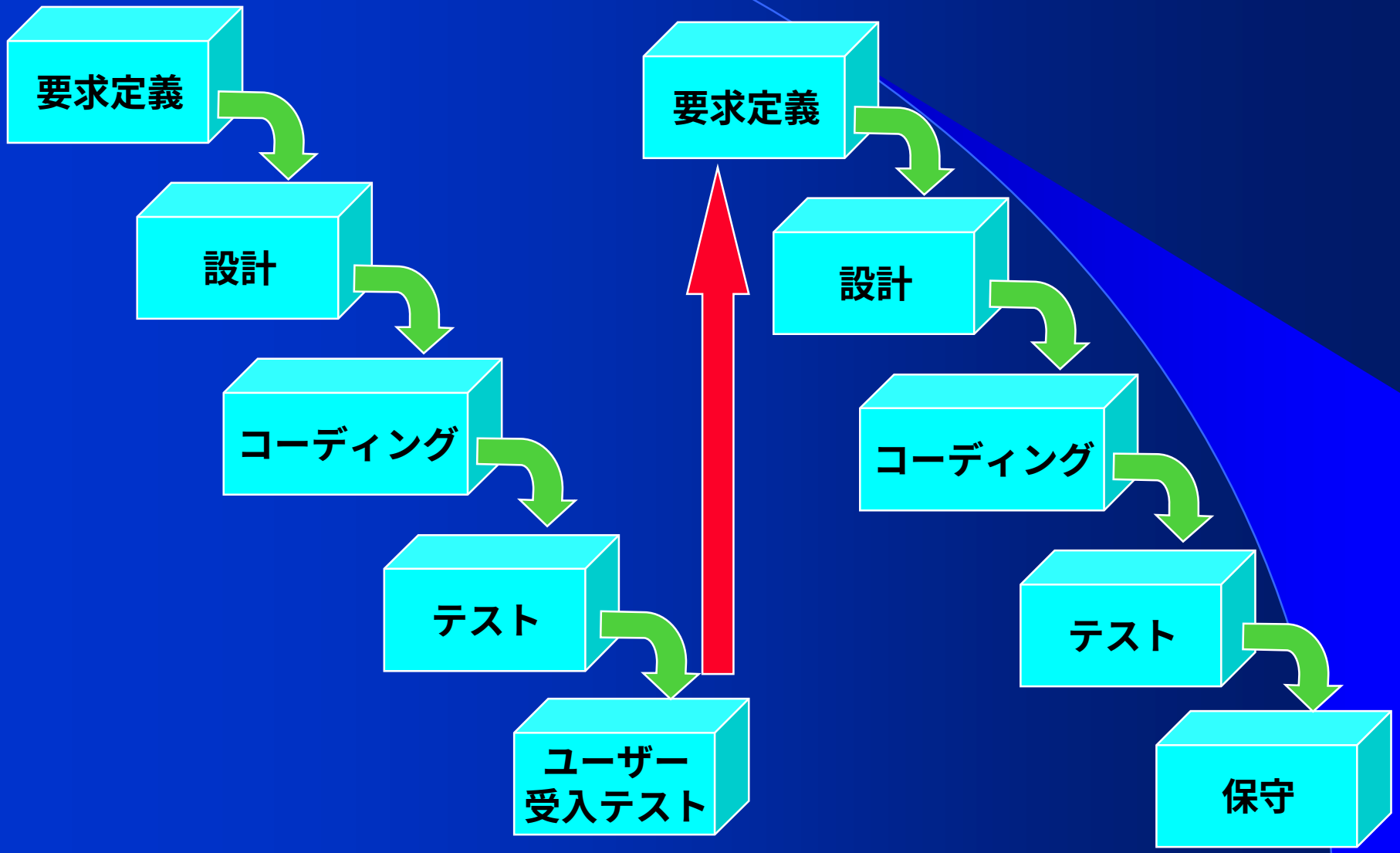
見た目の進捗はつかめても、真の進捗は最後までわからない

要求定義の誤りがもたらすコスト増

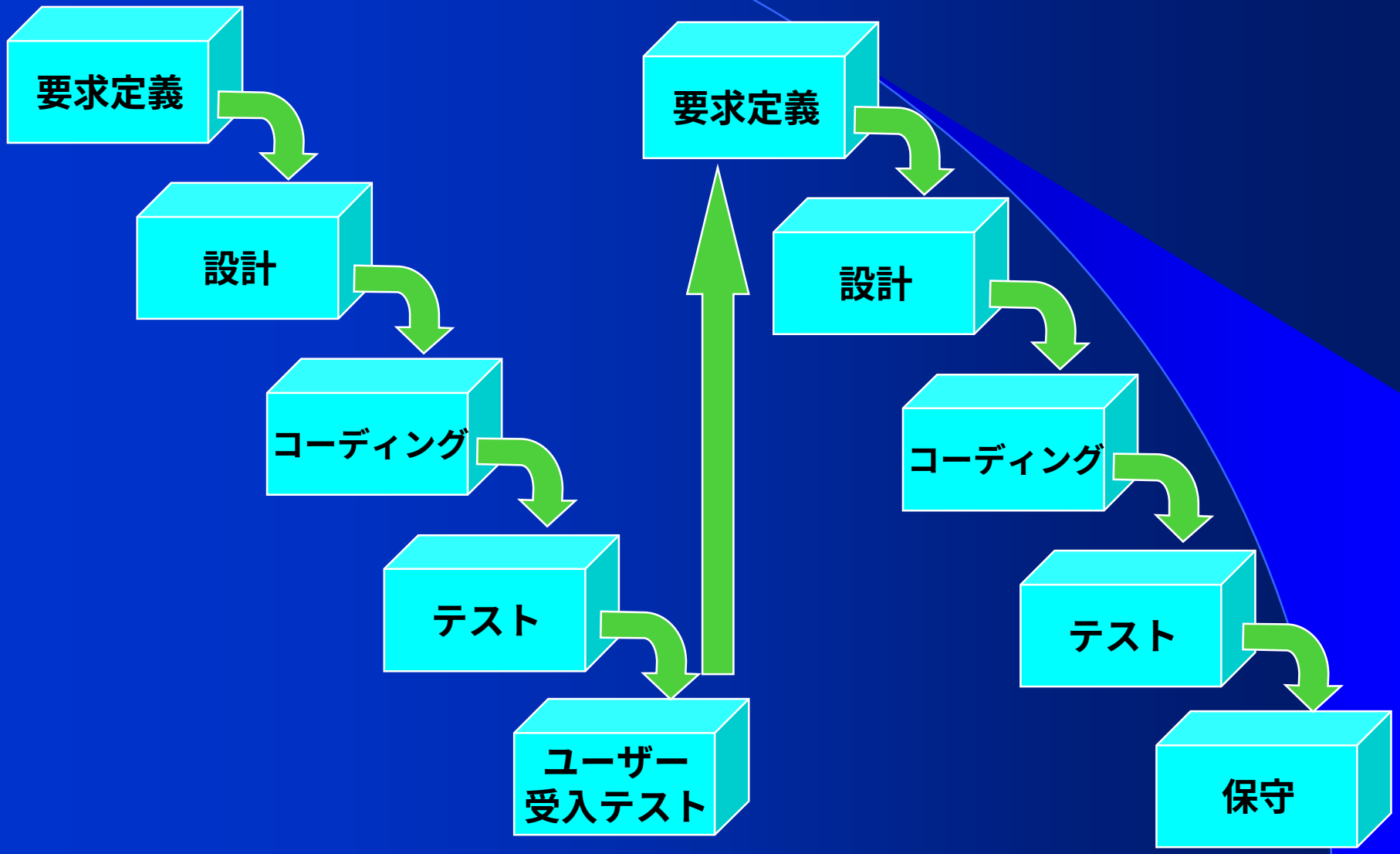


（出典：Barry W. Boehm, “Software Engineering Economics” Prentice Hall (1981) ）

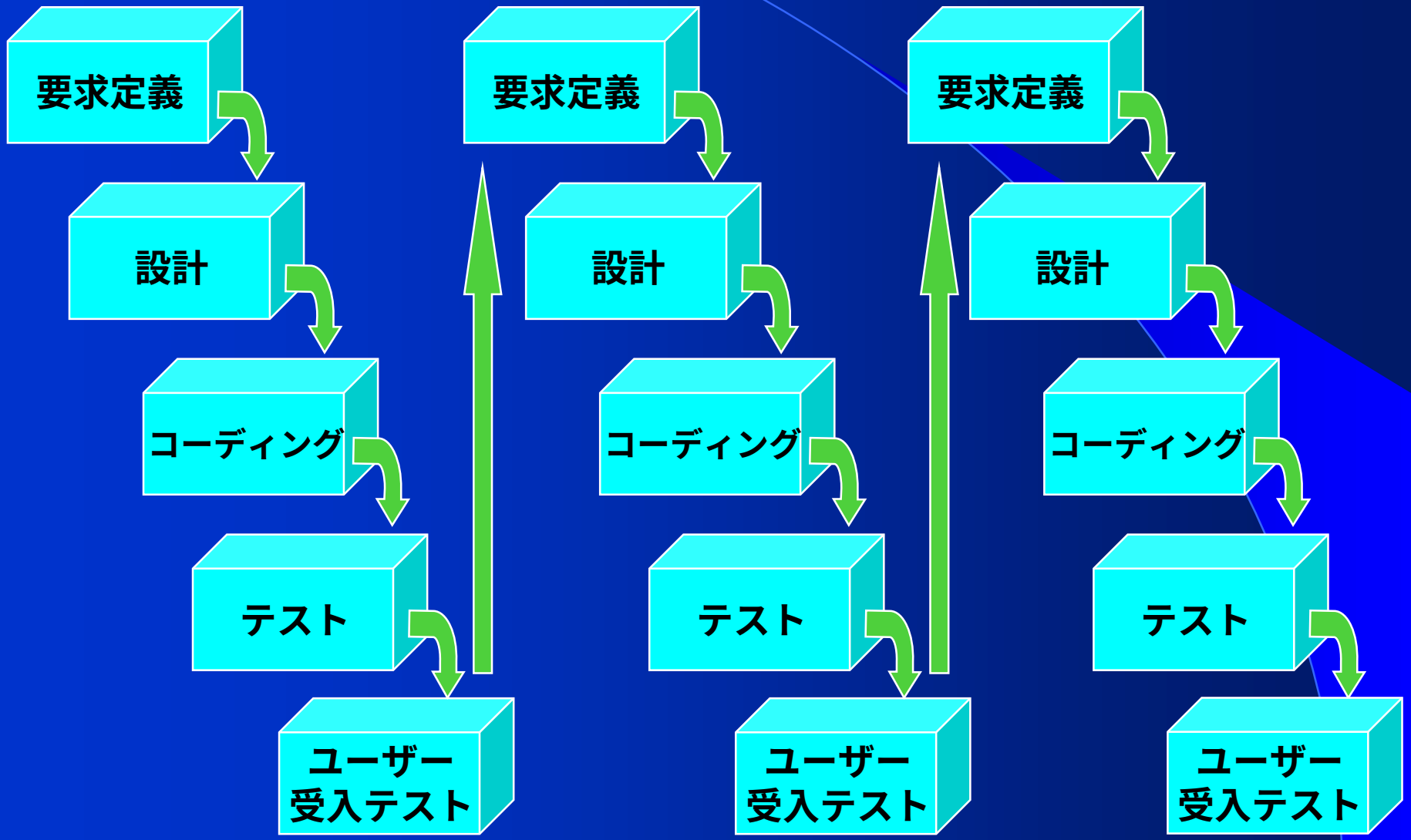
要求定義の欠陥は最悪の手戻りを招く



手戻りを前提にすればよいのでは？
（佐賀市　-サムスン- の例）



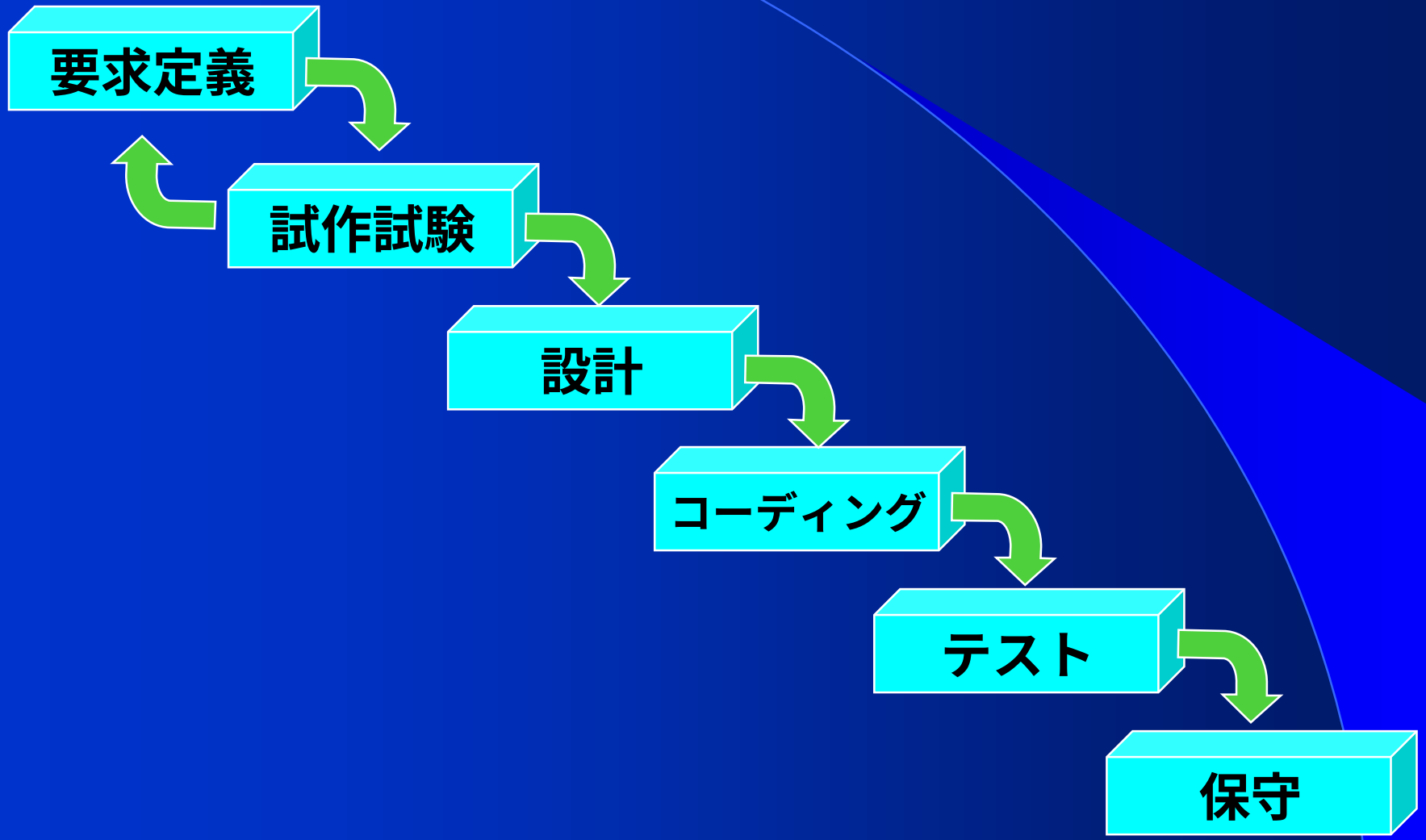
ウォータフォールを何度も繰り返せば...



根底から考え直そう

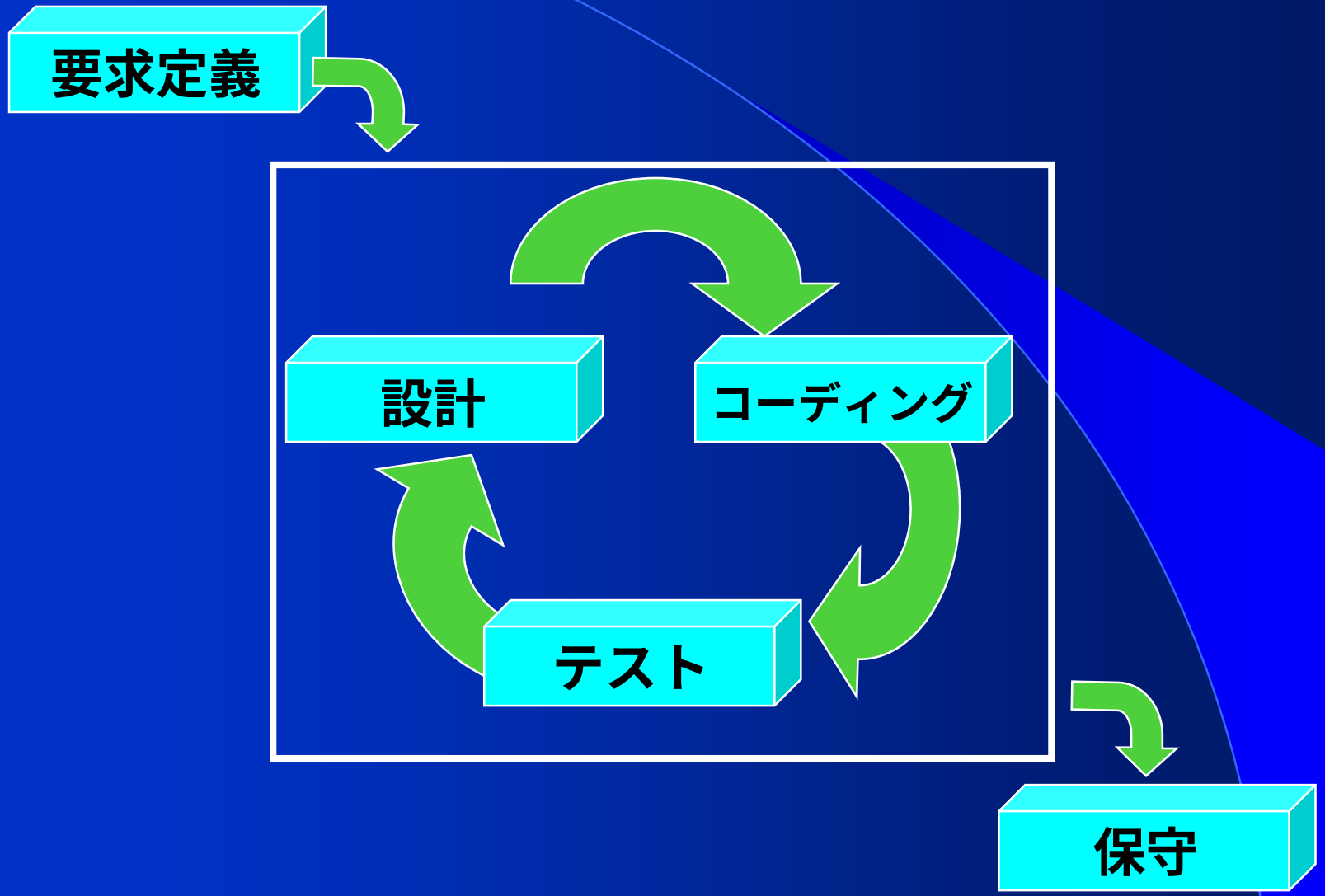
- 最初に完全な仕様書を作れるシステムは少ない
- 手戻りがあるのが当たり前だと考えよう
- リリース直前まで仕様や機能は変更できる
- 仕様書は最後にできればよい
- 「従来のソフトウェア工学では、終盤の設計変更は、悪いことであり、仕様が誤っていたとか不完全であったことを示唆した。だが、ここでの理念は、製品作成の過程でユーザーからのフィードバックに答えることだ」
  （マイケル・A・クスマノ「synchronize and stabilize ～ソフトウェア開発の成功例に学ぶ～」 CSK XPRESS, 2003, 10-11）

プロトタイピング・モデルで解決できるか？



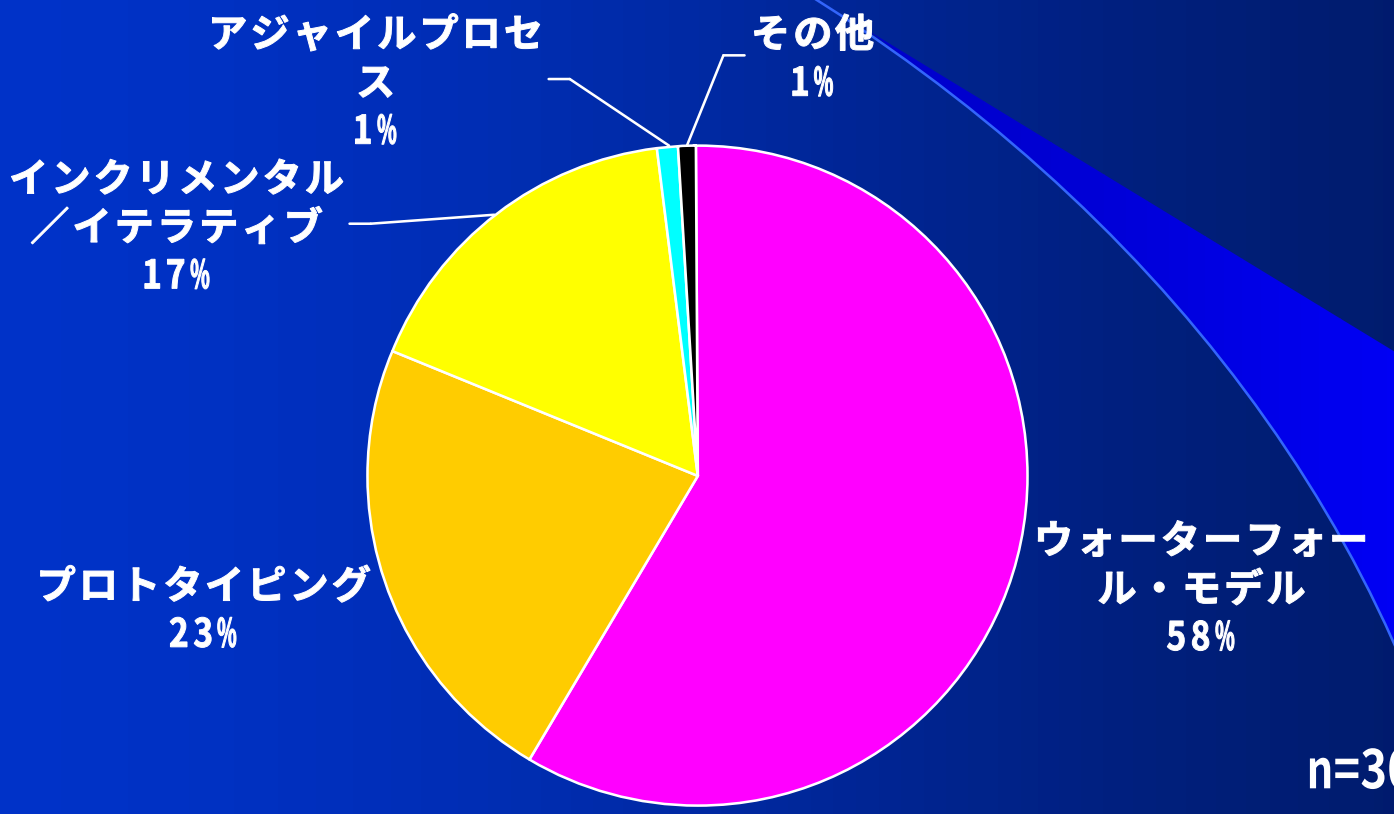
（出典：大場充ほか『ソフトウェアプロセス　改善と組織学習』　ソフト・リサーチ・センター（2003））

あまり良くないインクリメンタル・モデル



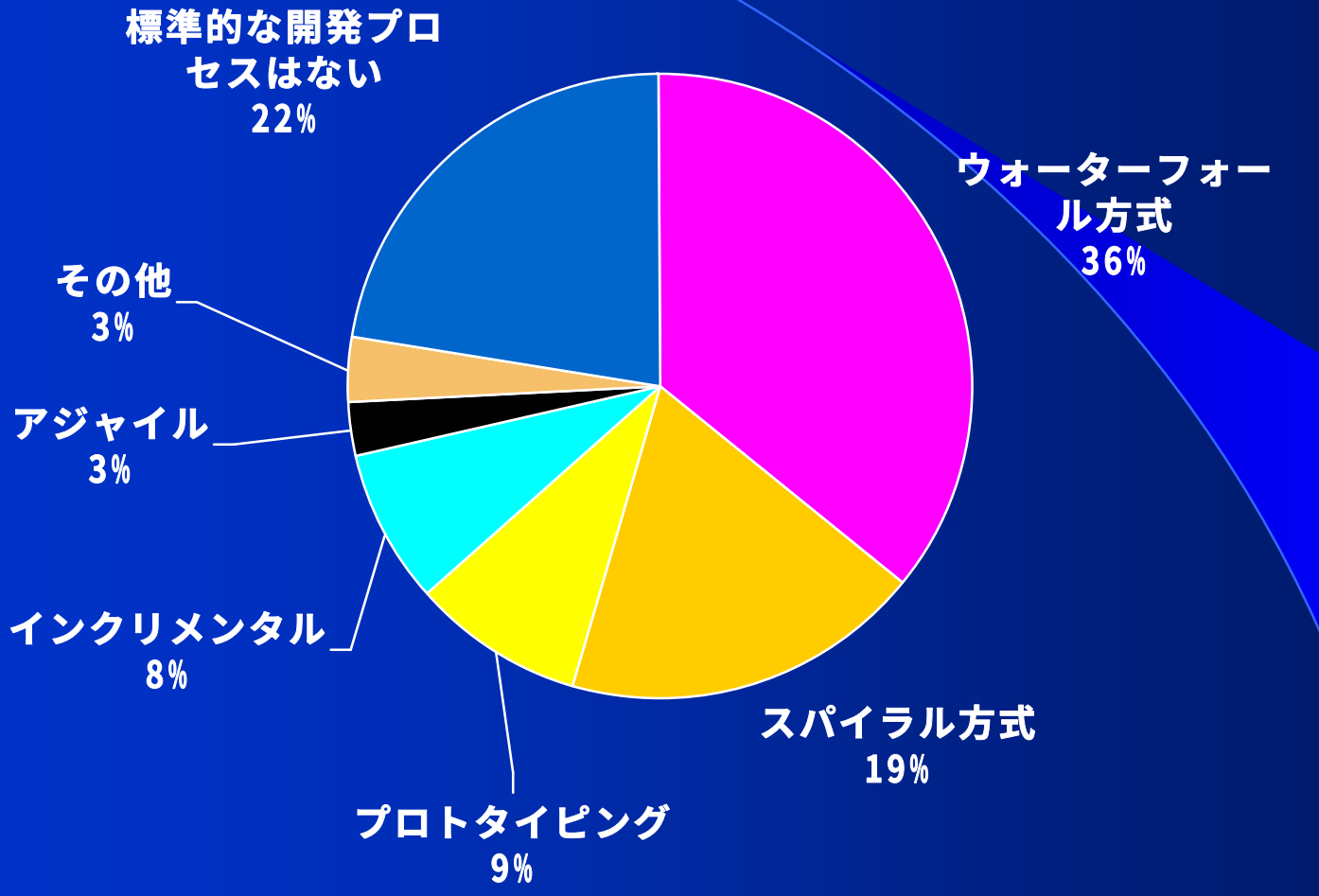
約６割がウォーターフォールモデルを利用

（利用している開発モデルは何か？）



（注）調査対象は、ソフトウェア開発プロジェクトに従事している会社員であり、
この１年間にスケジュールの遅延、予算超過を経験したことのあるもの
調査実施時期：2004年8月

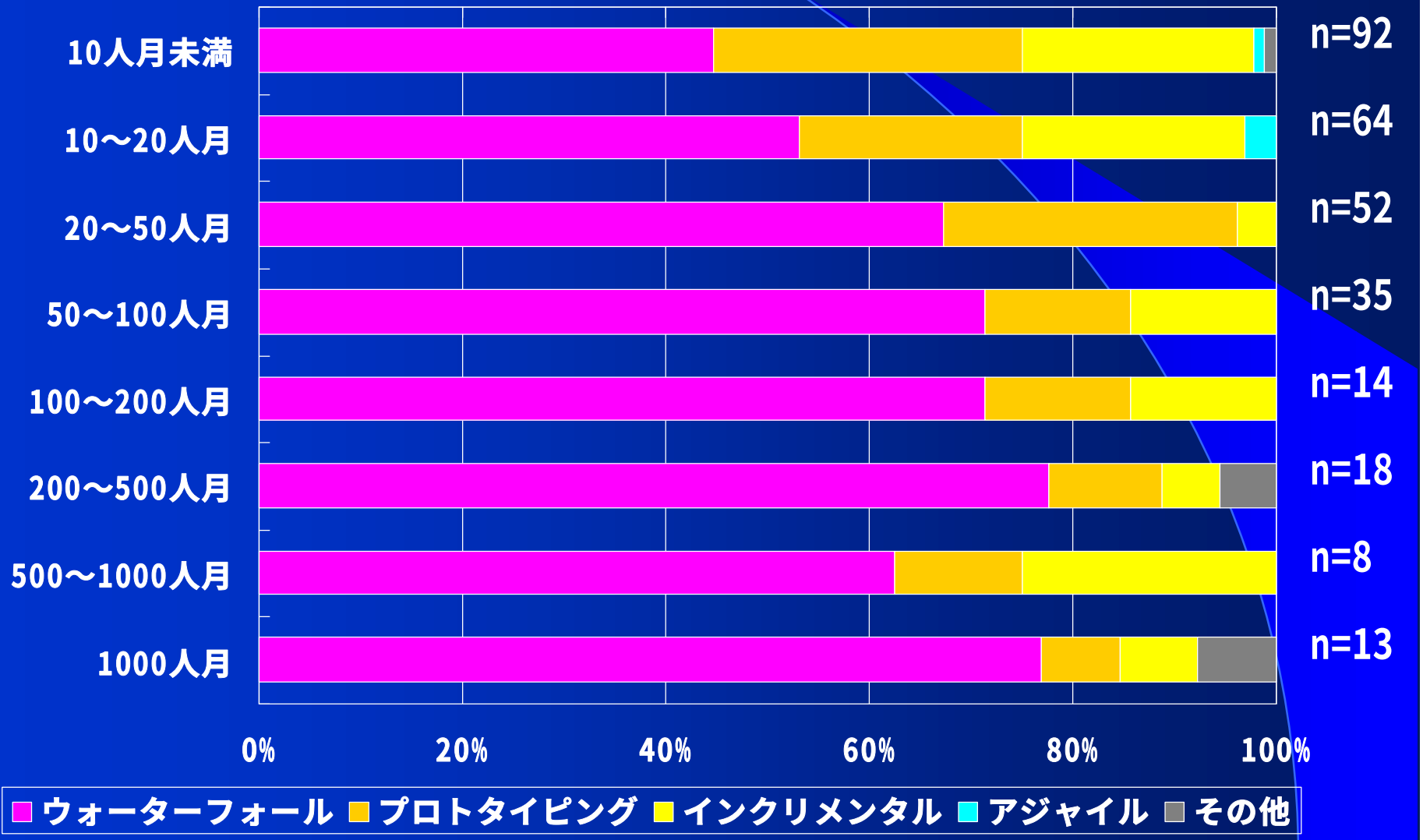
（参考）組込みの場合の開発プロセス



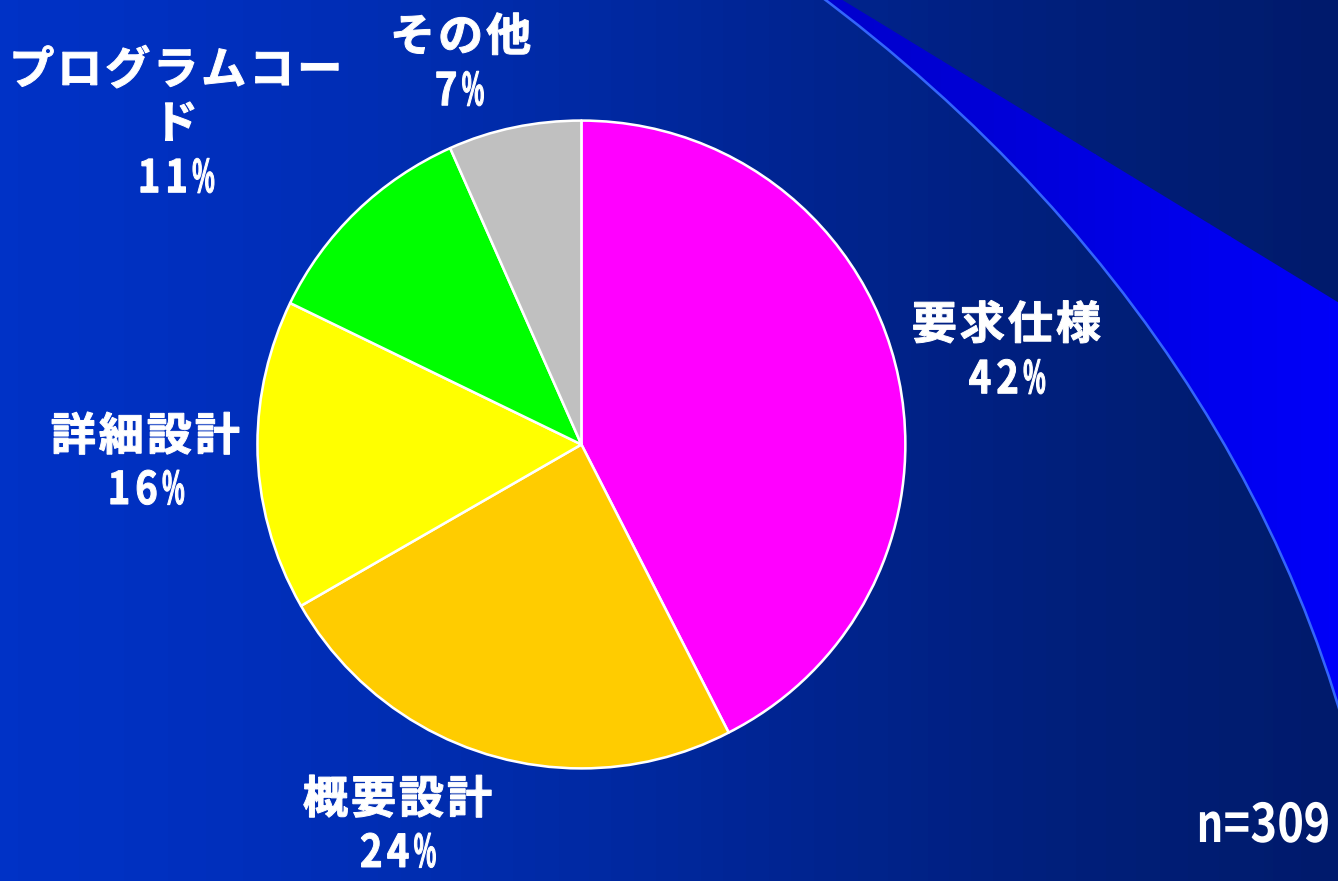
（出典：「2005年版組込みソフトウェア産業実態調査報告書」
経済産業省商務情報政策局（2005）

規模が小さくても ウォーターフォールモデルが主流である

（規模別の開発モデル）

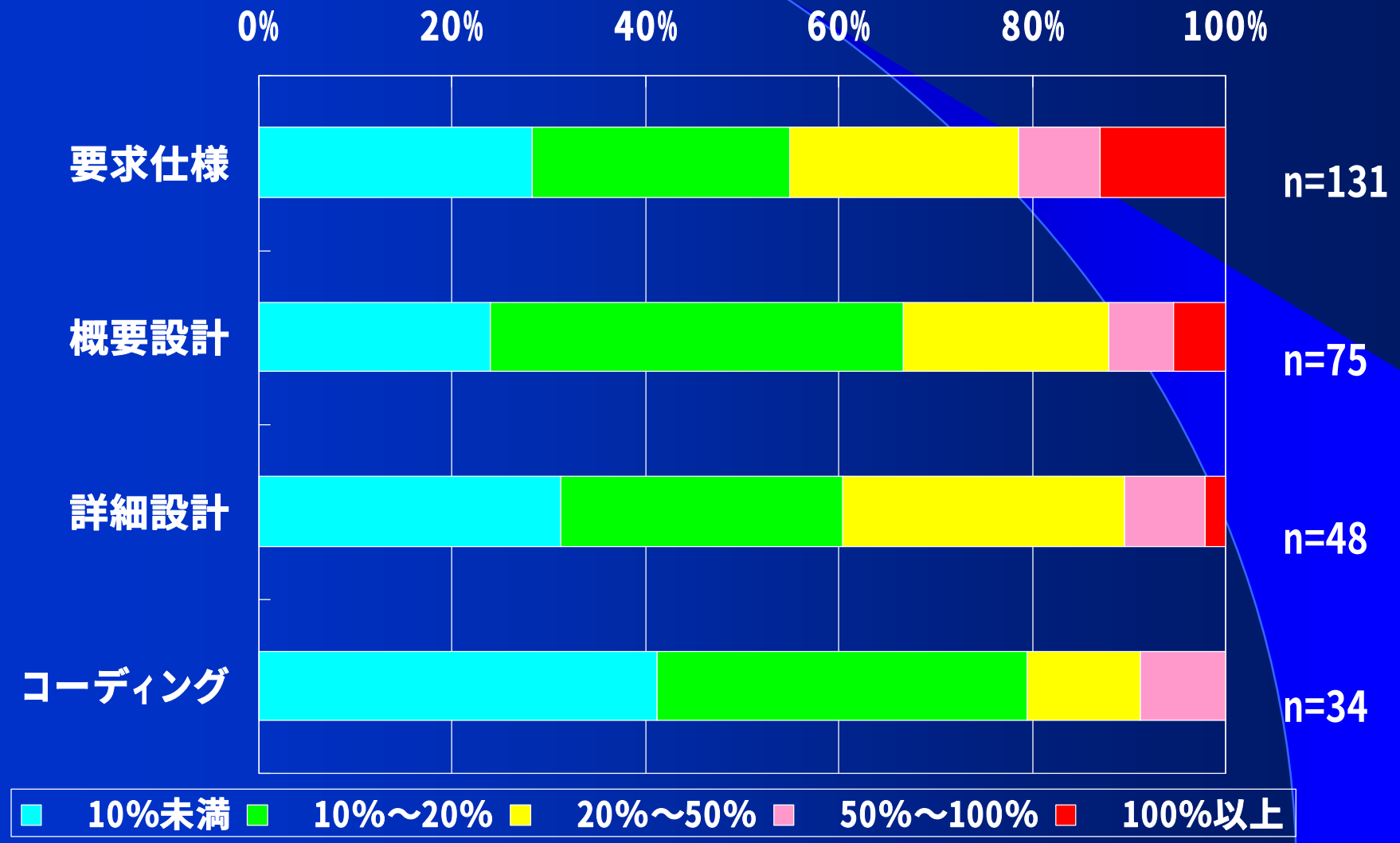


トラブルの原因の4割強が要求仕様
４分の１が概要設計
（トラブルの原因はどこにあったか？）

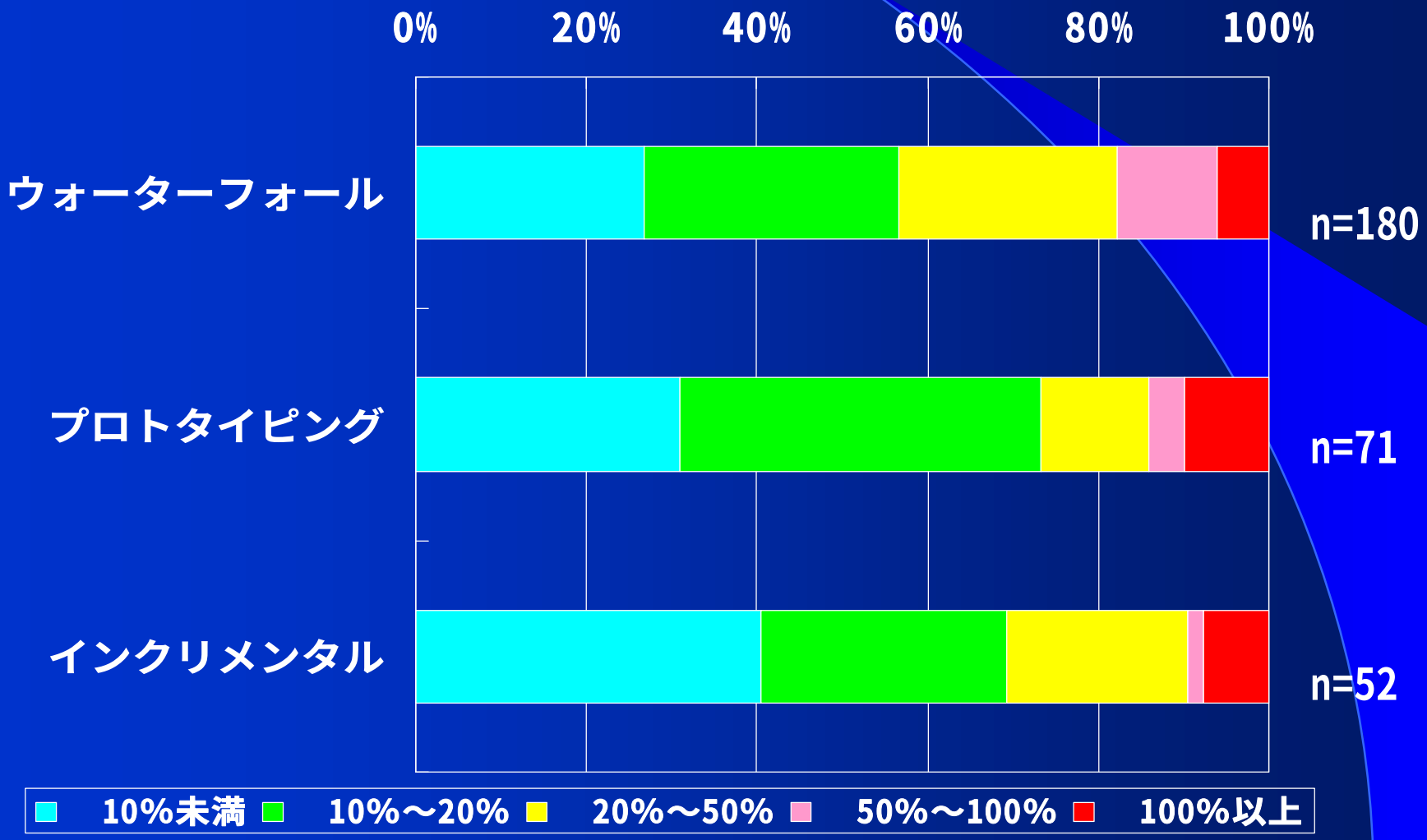


要求仕様に原因がある場合
大幅な予算超過を招くケースが多くなる

（トラブルの要因別の予算超過割合）



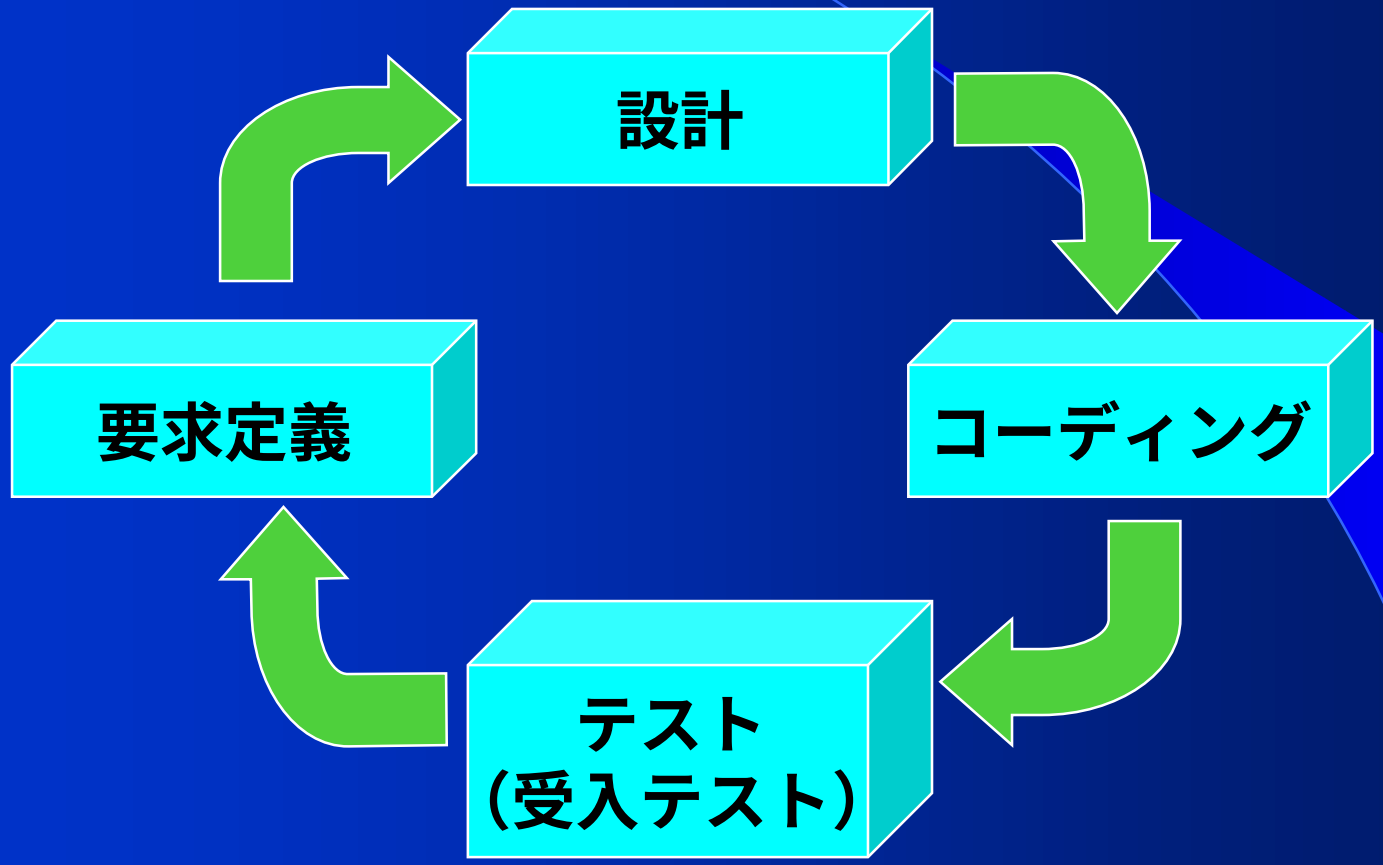
プロトタイピングやインクリメンタルでもトラブルは小さくなっていない



動くソフトウェアが一番重要

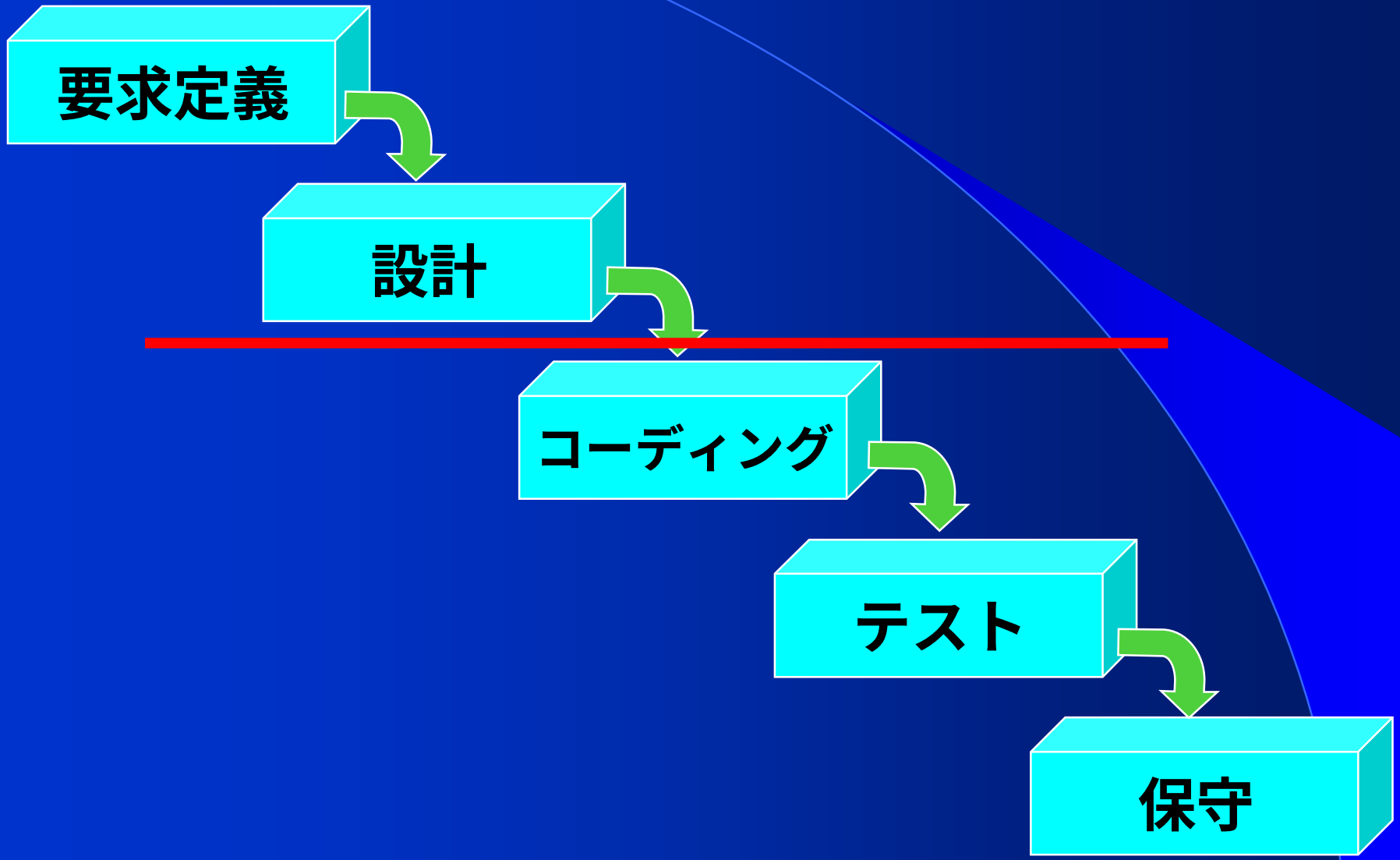
最終段階で仕様書の不備が発覚 完成したつもりでもエンドユーザから不満	⇒	完璧な仕様書は求めない 早く動くソフトを作ってエンドユーザーに見せる
要求仕様を固めるのに長い時間をかけている ユーザーが要求仕様を承認しようとしない	⇒	何をしたいのかをユーザーに書いてもらう 動くソフトウェアでユーザの確認をとる
仕様書や設計書では本当の進捗はつかめない 遅延や予算超過のリスクも大きい	⇒	動くソフトウェアで進捗確認を行う 遅延も予算超過もないプロジェクト管理

理想的なスパイラルモデルの概念図
（ソフトウェア開発におけるPDCAサイクル）

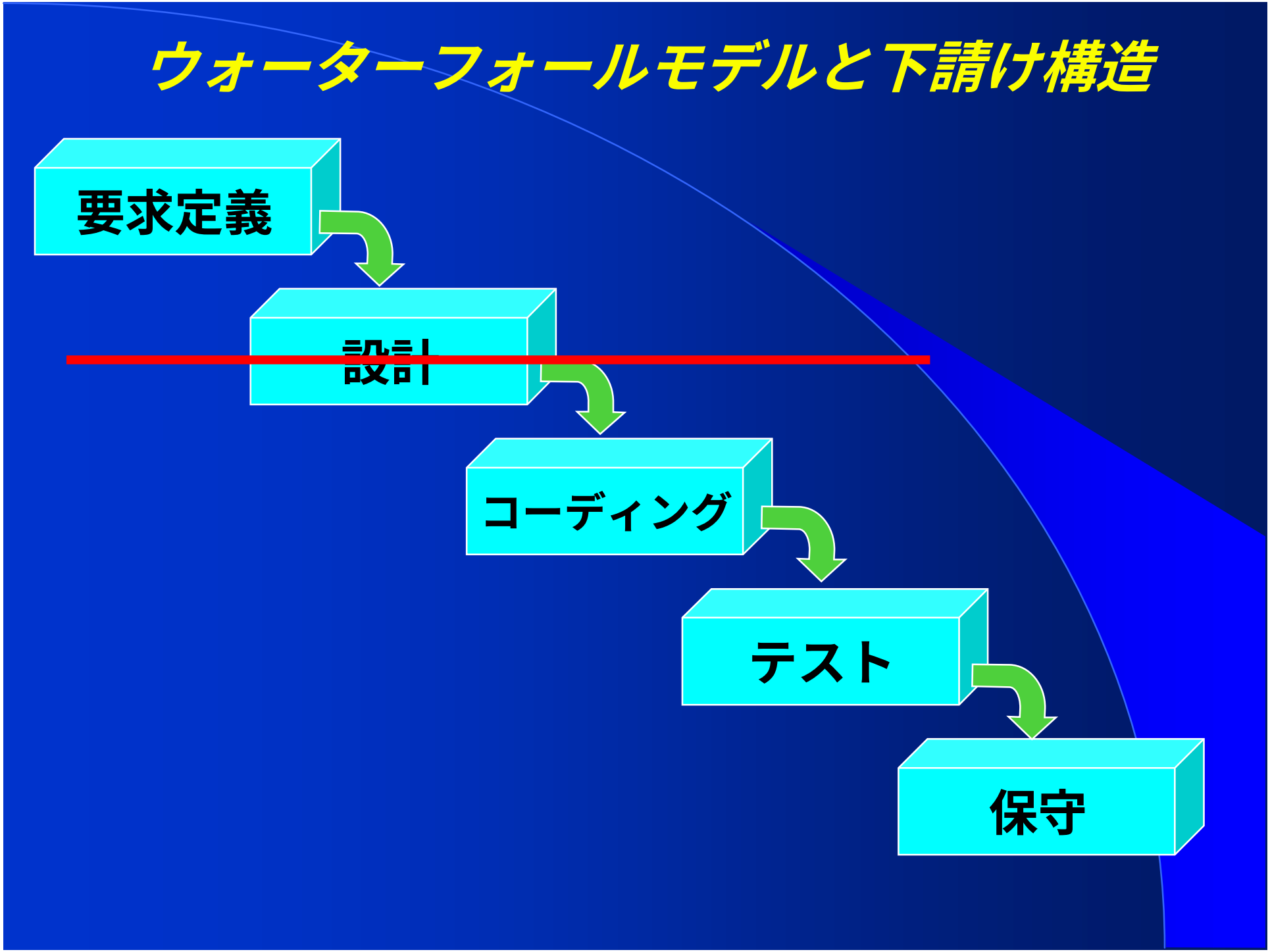


あるいは
エンドユーザーによる実利用

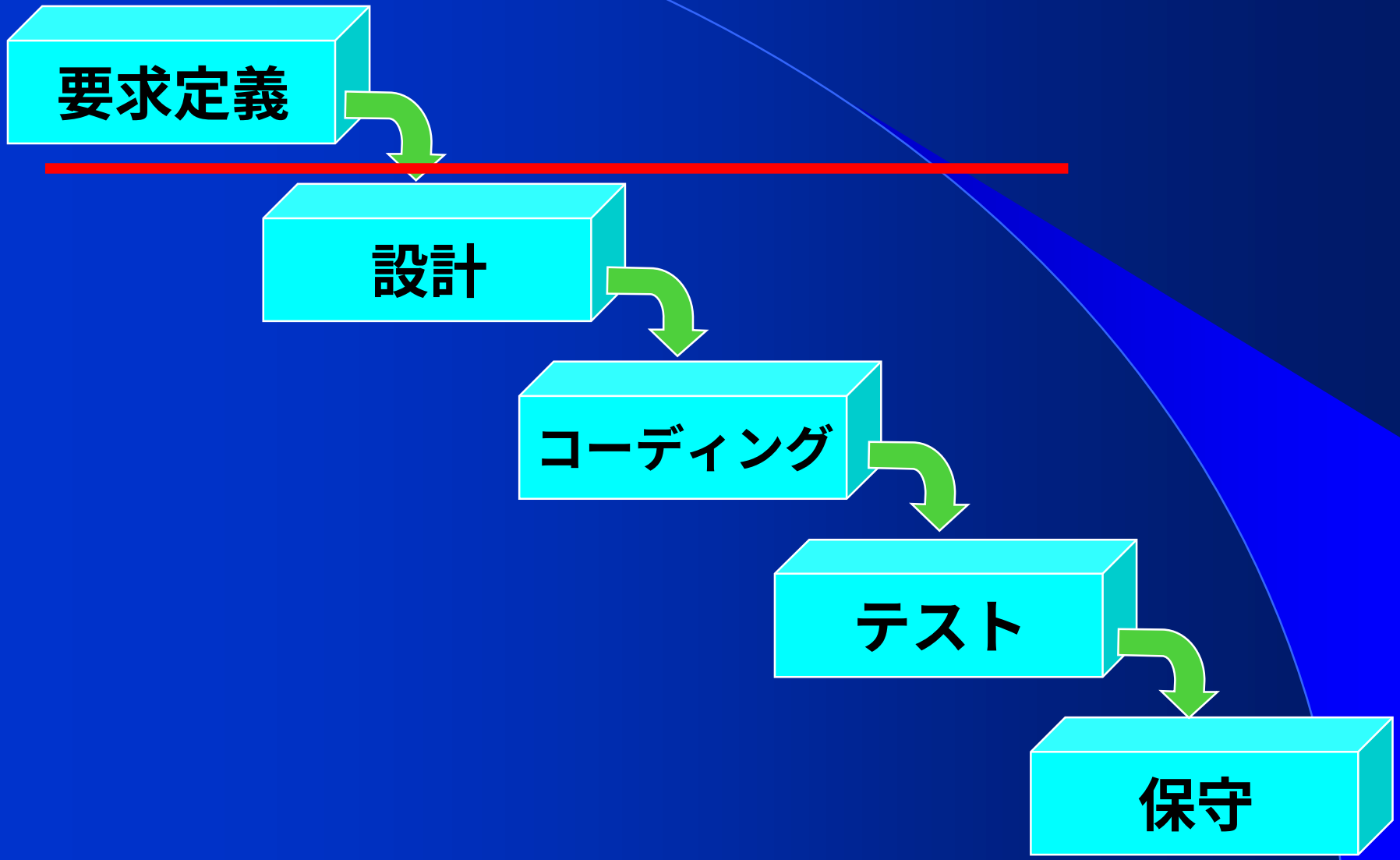
ウォーターフォールモデルと下請け構造



ウォーターフォールモデルと下請け構造



ウォーターフォールモデルと下請け構造



2

個人の能力をより重視すること

優秀な技術者の不足と
恒常的な残業の悪循環を断ち切る

「優れた人が優れたソフトウェアを作る」

- 「優れた人が優れたソフトウェアを作る、というのは昔からの考えだ」
  （エドワード・ヨードン『プログラマーの復権』）

- 「プログラマの世界は、10%の優秀な人と、40%の普通の人と、50%の足を引っ張る人からなっています。プログラムは10%の優秀な人が作ります」
  （新・闘わないプログラマ　No. 17 プログラマという人種）

- 「プログラミングマネージャーは、できるプログラマとできないプログラマの間に、大きな生産性の相違があることに、前々から気がついていた。」
  （フレデリック・P・ブルックス『人月の神話』）

個人の能力差に関する既往の研究

◆プログラミング（デバッグ作業を含む）における個人の能力差は、最大で **28 対 1**

H. Sackman, W. J. Erikson, E. E. Grant, “Exploratory experimental studies comparing online and offline programming performance”, Communications of the ACM Volume 11 , Issue 1 (January 1968)

◆ 個人の生産性の差は **5 対 1**

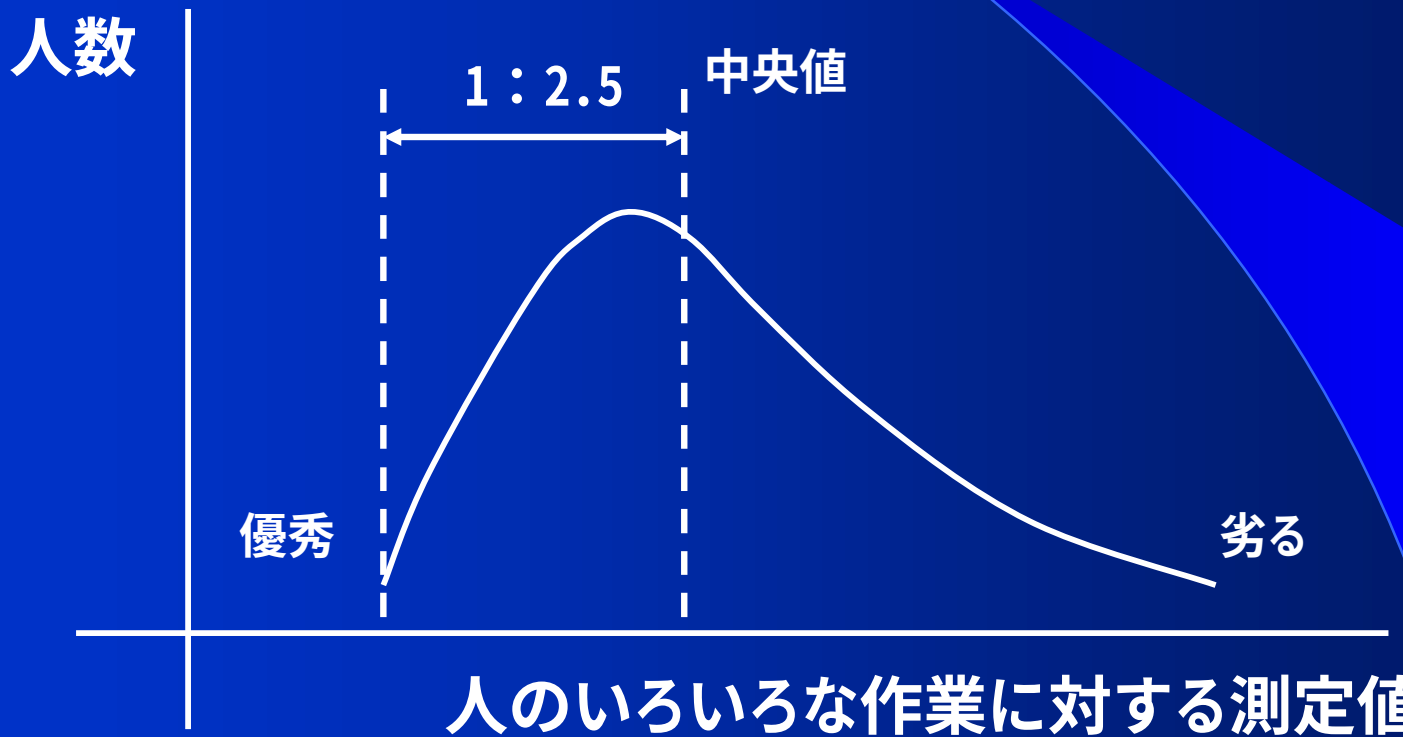
Tom DeMarco , Tim Lister, “Programmer performance and the effects of the workplace”, Proceedings of the 8th international conference on Software engineering, p.268-272, August 28-30, 1985

◆ 個人の能力比は **2 から 7（最大で 7.3：1 ）**

Lutz Prechelt, An empirical study of working speed differences between software engineers for various kind of task”, IEEE software engineering, February 16, 2000

最優秀者の測定値は平均的作業者の約２.５倍である

デマルコとリスターが発見したプログラマの生産性の個人差



（出典：トム・デマルコ、ティモシー・リスター『ピープルウェア』日経ＢＰ（2001）、p.56）

生産性がマイナスのプログラマ

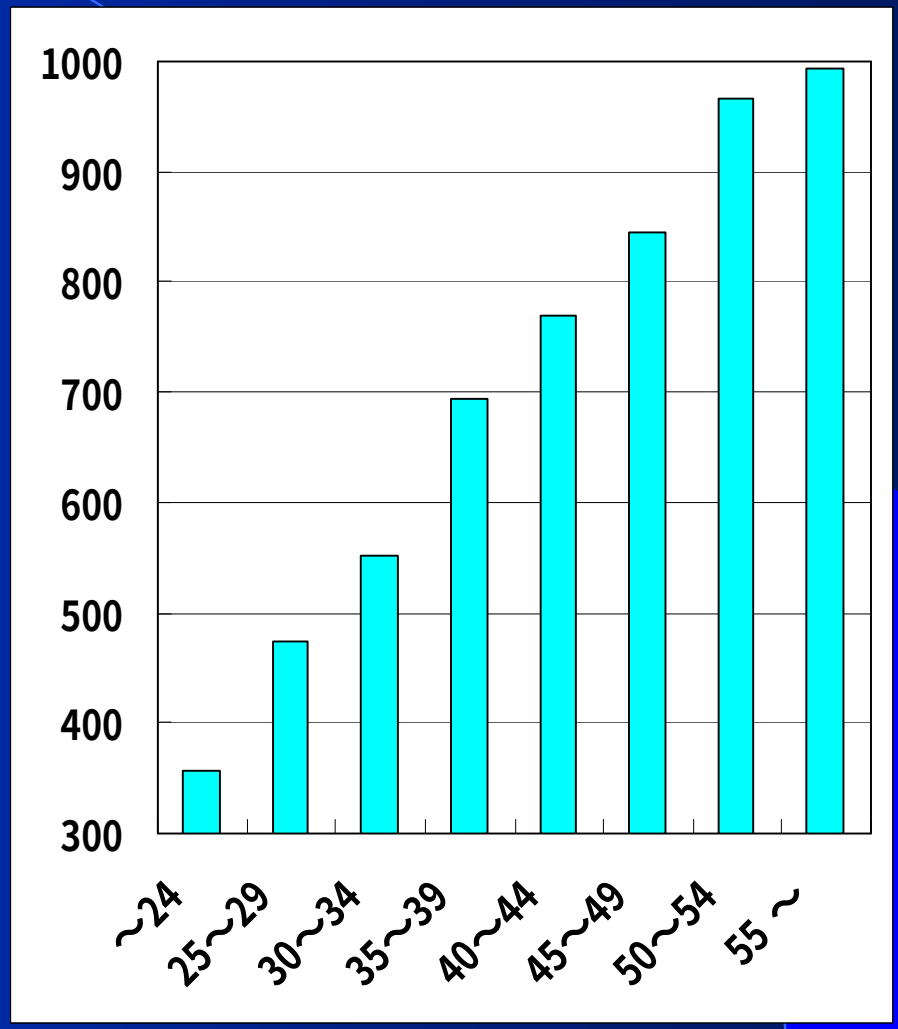
- 生産性に差があるという話は、話の前半にすぎない
- デマルコとリスターの研究では、166人中13人が作業を終えることができなかった。
- カーチス（Bill Curtis）の研究では、60人中6人が単純なデバッグ作業を終えられなかった。
- 誰が「彼ら」の代わりにプログラムを書くのか？
- 誰が「彼ら」の作りこんだバグを取り除くのか？

（参考）スティーブ・マコネル『ソフトウェア開発プロフェッショナル』
日経BP社、2005年1月

ソフトウェア技術者の報酬（年齢別）

年齢が高くなるほど総年収は高くなっている

年齢	年収	標準偏差
～24	357.1	79.0
25～29	473.4	75.7
30～34	550.7	105.4
35～39	693.1	118.5
40～44	769.2	141.1
45～49	844.0	178.4
50～54	965.6	199.4
55 ～	992.9	206.6



（出典：「ITエンジニアのための仕事・市場基準の人材評価システム」平成15年3月、（社）情報サービス産業協会）

ソフトウェア技術者の報酬（重回帰分析）

「年収は年齢や企業規模と非常に密接な関係にある」

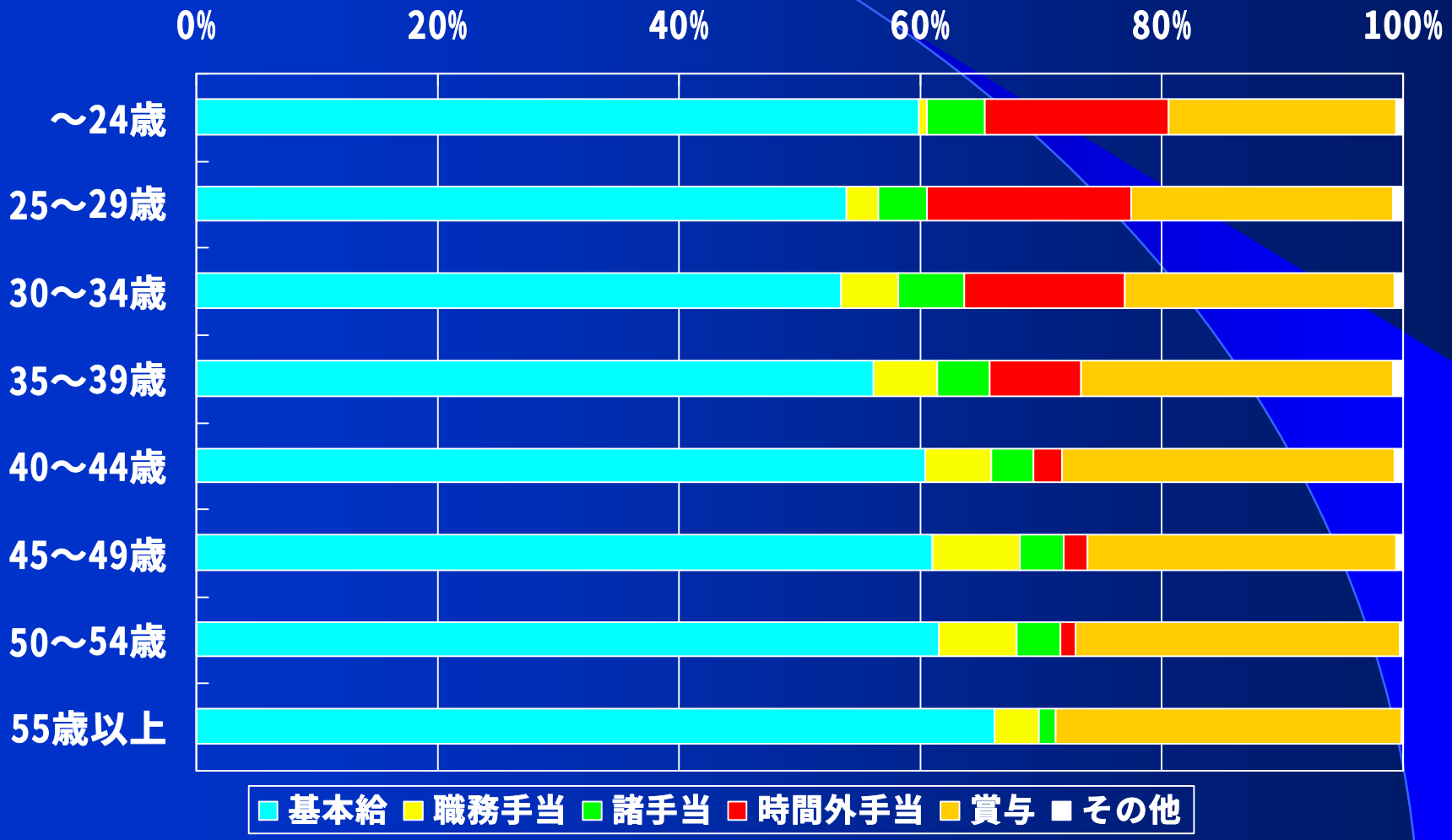
被説明変数は年収、調整済みR２乗値は0.678、職種ダミーは「業務系スペシャリスト」が基準、企業系列ダミーは「独立系」が基準、***は1%水準、 **は5%水準、 *は10%水準で有意

説明変数		係数	t値	有意確率
(定数)		-128154.699	-0.523	0.601
年齢		153756.503	22.785	0.000 ***
職種	プロジェクトマネジャー	638747.264	3.908	0.000 ***
	システムマネジャー	640443.817	2.696	0.007 ***
	コンサルタント	934241.909	3.950	0.000 ***
	エンジニアリング系スペシャリスト	15090.888	0.107	0.915
	運用／ネットワーク系スペシャリスト	-129426.154	-0.903	0.367
企業系列	ユーザ系	461143.412	4.137	0.000 ***
	メーカー系	-574650.332	-2.897	0.004 ***
正社員数		600.028	6.910	0.000 ***
業務スキル	1 流通･生産管理･財務経理系	62548.606	1.776	0.076 *
	2 保険･証券系	73859.816	0.900	0.369
	3 銀行系	-12712.825	-0.148	0.882
	4 行政･大学教育機関系	145240.503	1.975	0.049 **
	5 科学技術･CADCAM系	-296612.867	-2.267	0.024 **

(出典：「ITエンジニアのための仕事･市場基準の人材評価システム」平成15年3月、(社) 情報サービス産業協会)

残業代を払うことは当然なのか？

ソフトウェア技術者の年齢別給与構成比



（出典：「ITエンジニアのための仕事・市場基準の人材評価システム」平成15年3月、（社）情報サービス産業協会）

できないプログラマの方が給与が多くなる

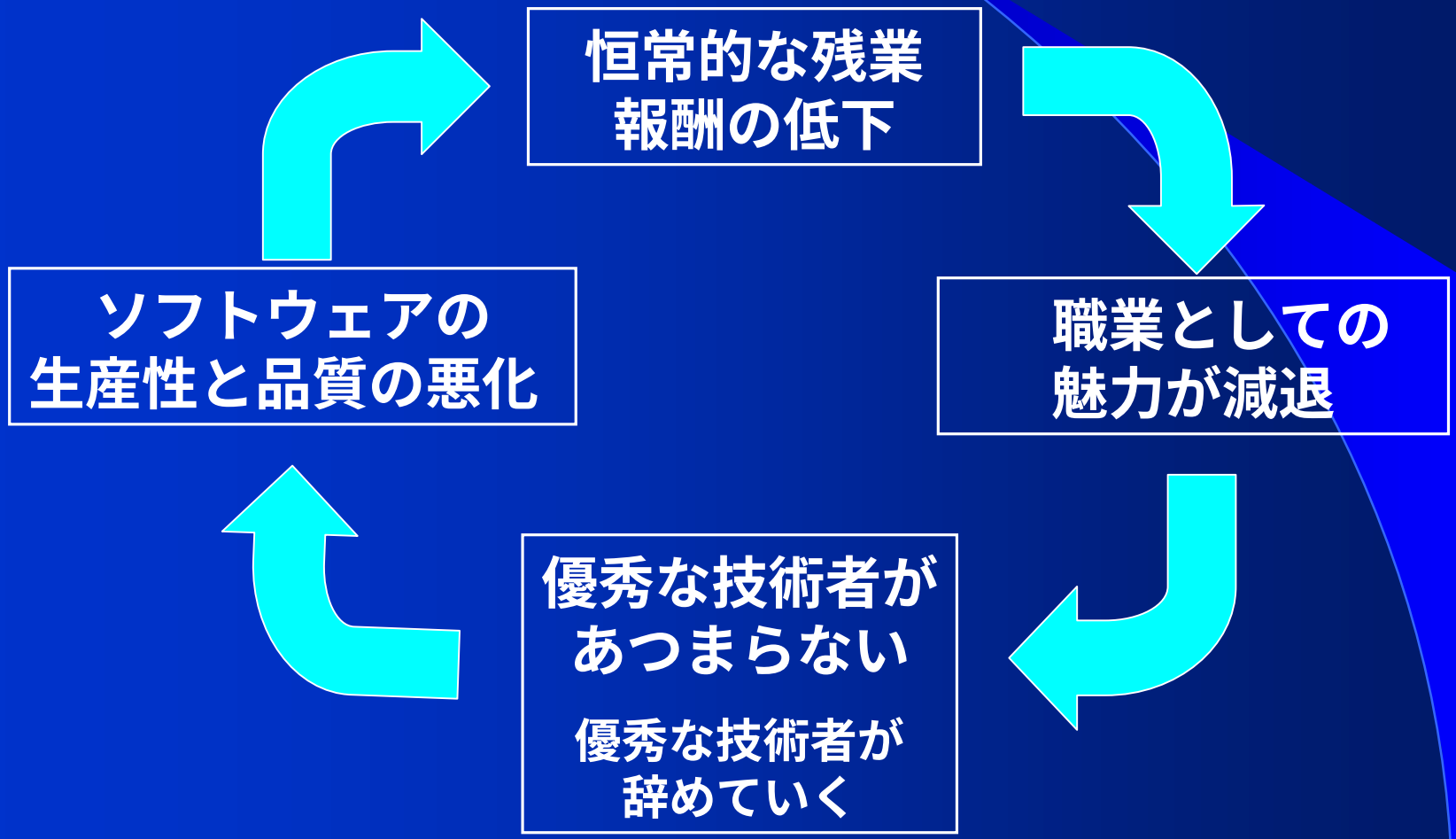
- ITエンジニアの年齢別給与構成比を見ると、残業手当の占める割合が「24歳以下」で15.3%、「25～29歳」で16.9%、「30～34歳」で13.3%、「35～39歳」で7.6%を占める。
  （「ITエンジニアのための仕事・市場基準の人材評価システム」平成15年3月、（社）情報サービス産業協会による）

- 「プログラムにかかる時間は、下手ほど長くなる。そして労働意欲が高いと評価される。」
  （藤原博文『コの業界のオキテ！！』 技術評論社（1995））

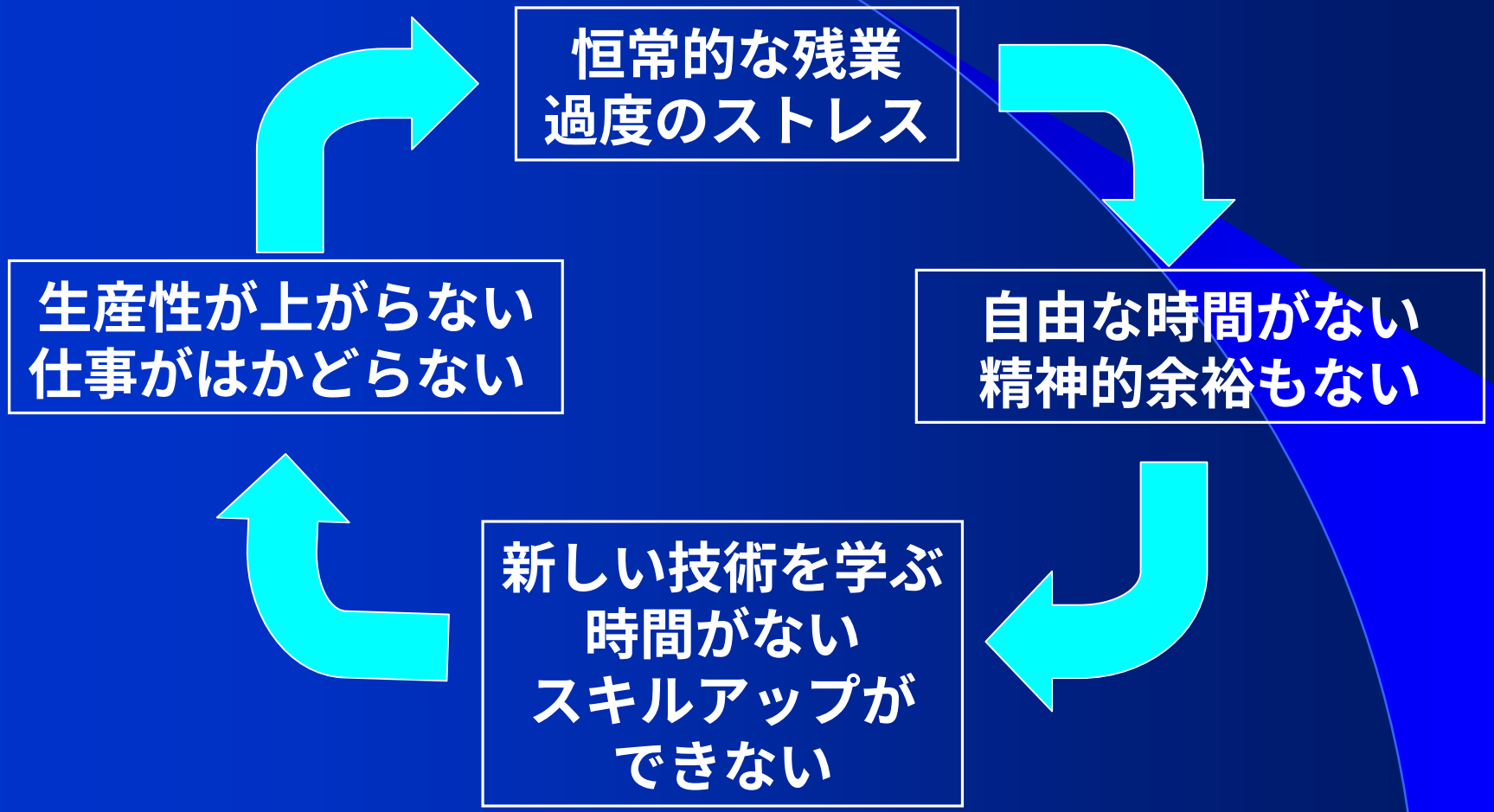
- 「アメリカのソフトウェア業界では、年俸やプロジェクトベースで給料を支払うため、通常は残業手当がつかない」
  （トム・デマルコ＆ティモシー・リスター『ピープルウェア』 日経BP社（2001））

最初に逃げ出すのは優秀なプログラマである

ソフトウェア企業（産業）における悪循環



個人における悪循環


恒常的な残業
過度のストレス
自由な時間がない
精神的余裕もない
新しい技術を学ぶ
時間がない
スキルアップが
できない
生産性が上がらない
仕事がはかどらない

残業時間・労働時間の比較

	所定内労働時間	所定外労働時間 (残業時間)	労働時間の合計
情報処理産業	1892時間	260時間	2152時間
全産業平均	1708時間	120時間	1828時間
差	184時間	140時間	324時間

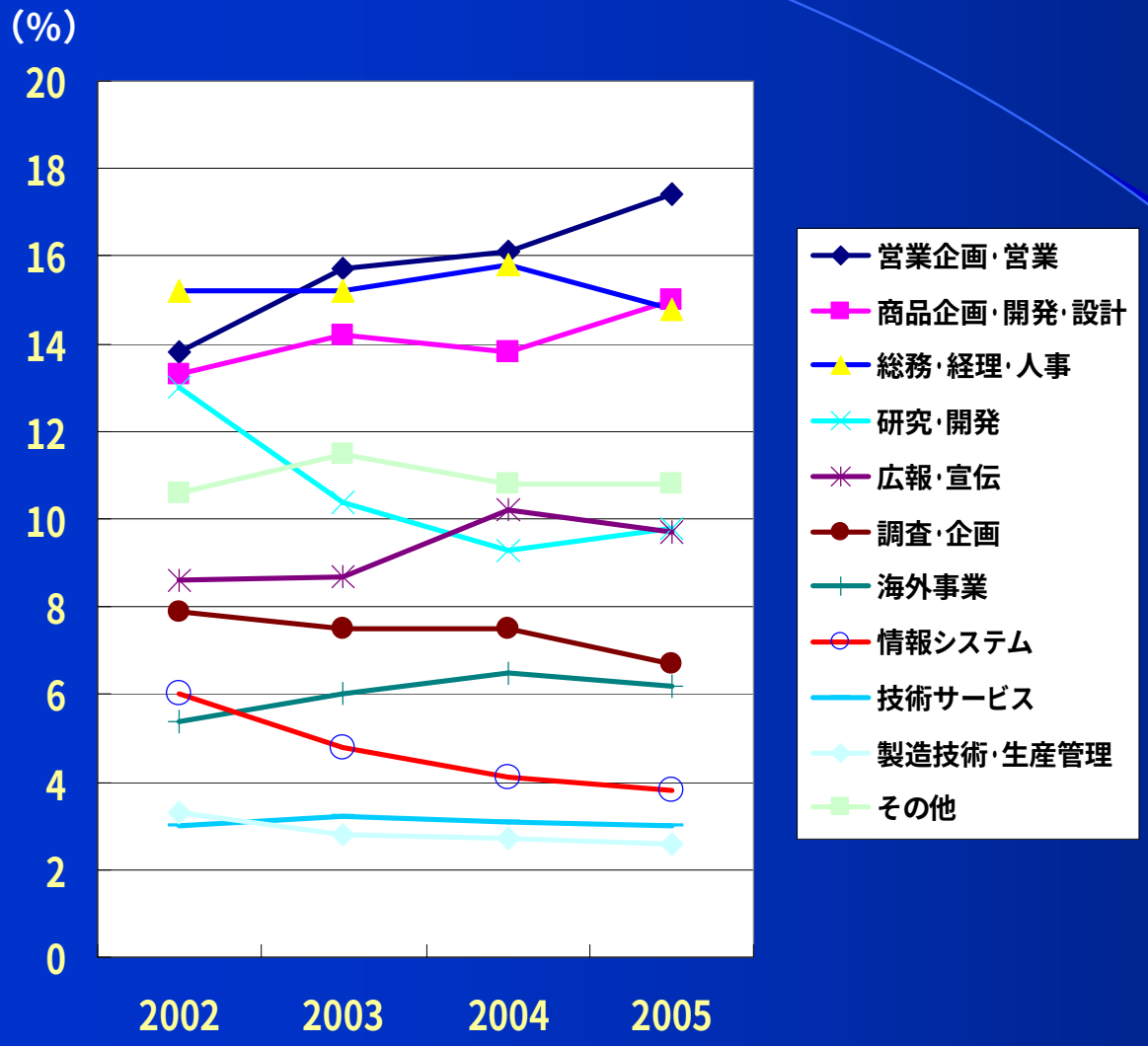
（注）データはいずれも2003年度
（出典）情報処理産業経営実態調査報告（2004年版）、毎月勤労統計調査平成15年分結果確報

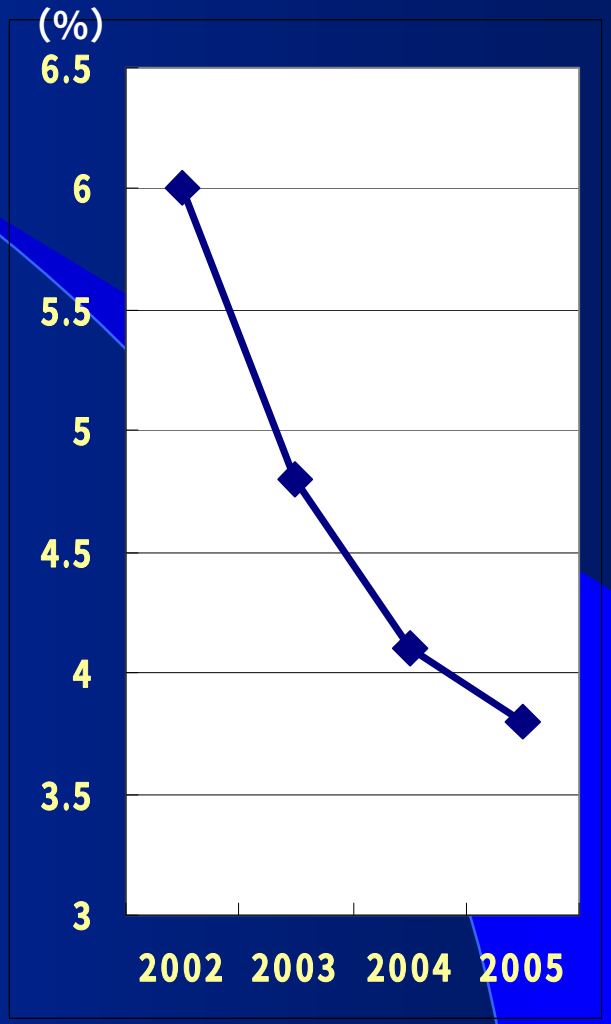
- ITプロフェッショナルの年間残業時間は 570 時間
- ITプロフェッショナルの 51.2 % が転職を希望

（参考：転職を希望する就業者（全産業平均）の割合は 9.7 %）

（出典）日経コンピュータ 2005年12月12日号、
日経BP社がWebサイト「IT Pro」上で実施した労働実態・意識調査

就職を希望する職種

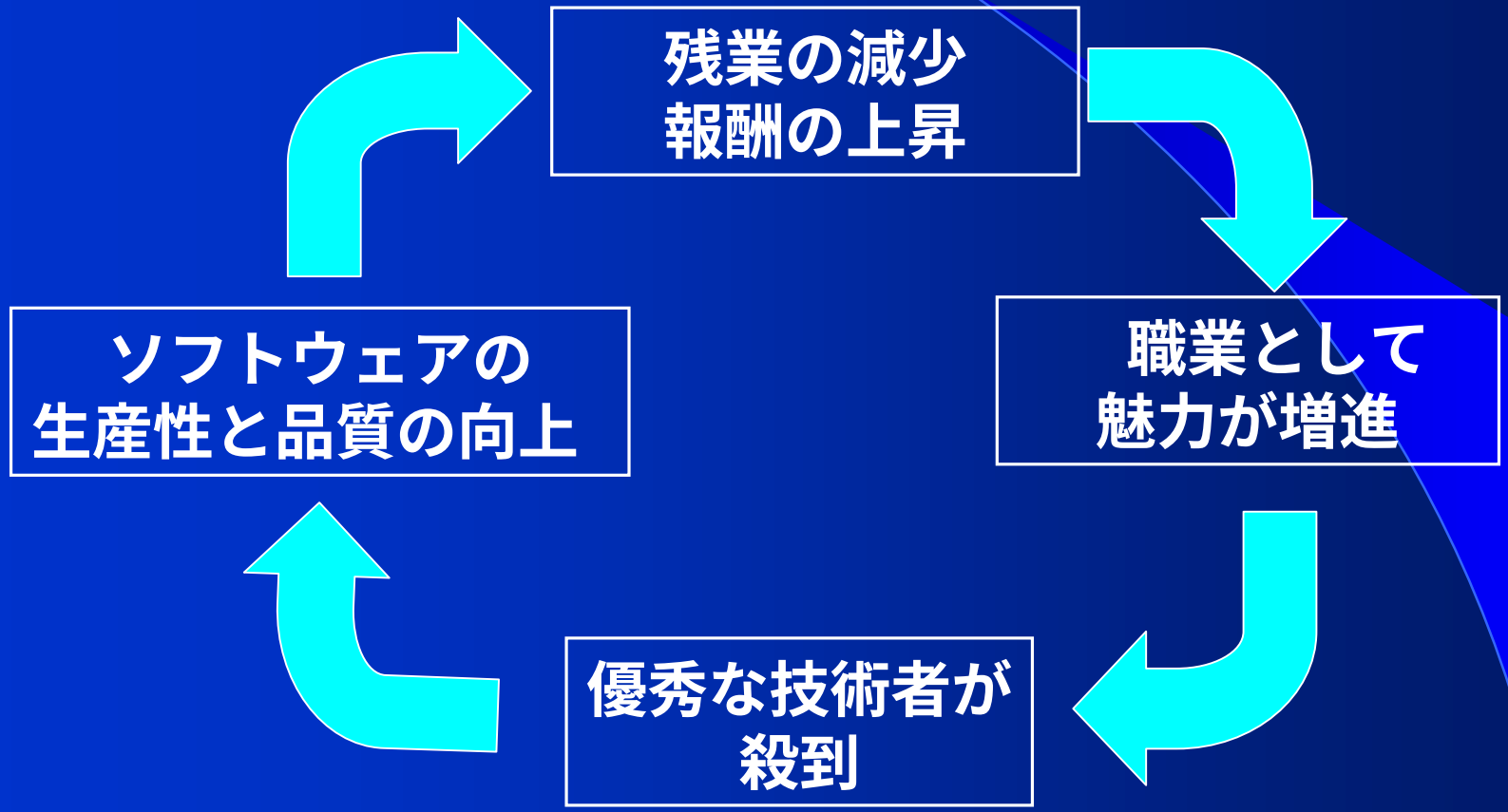




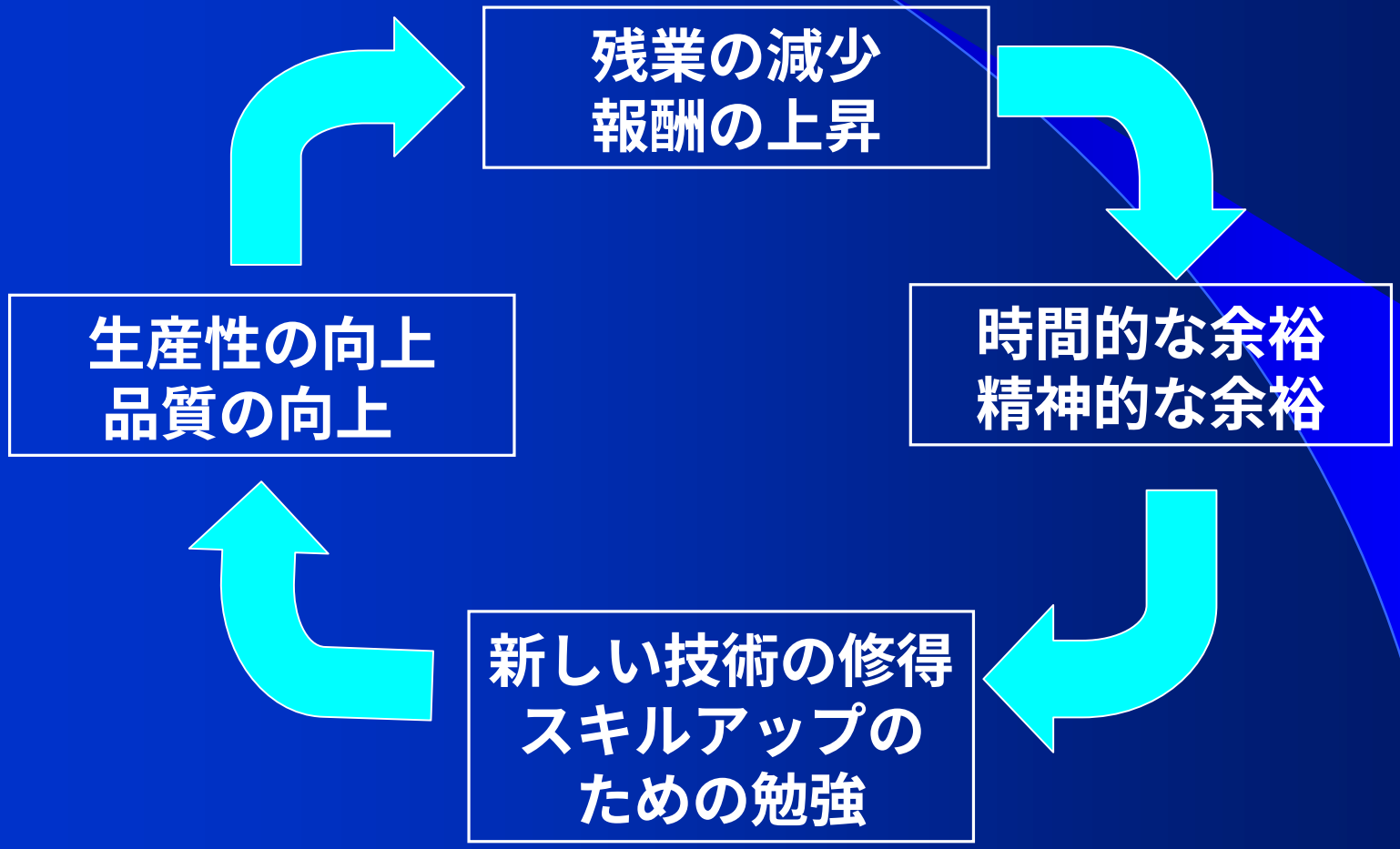
情報システム系の推移

（出典）毎日コミュニケーションズ「大学生の意識調査」（2005）

ソフトウェア企業（産業）が目標とすべき好循環



ソフトウェア技術者が目標とすべき好循環



3

ユーザー側の問題と責任

ソフトウェア産業を育てるのはユーザー

「店が客を育て、客が店を育てる」

もっとも効率のよいソフトウェア開発方法

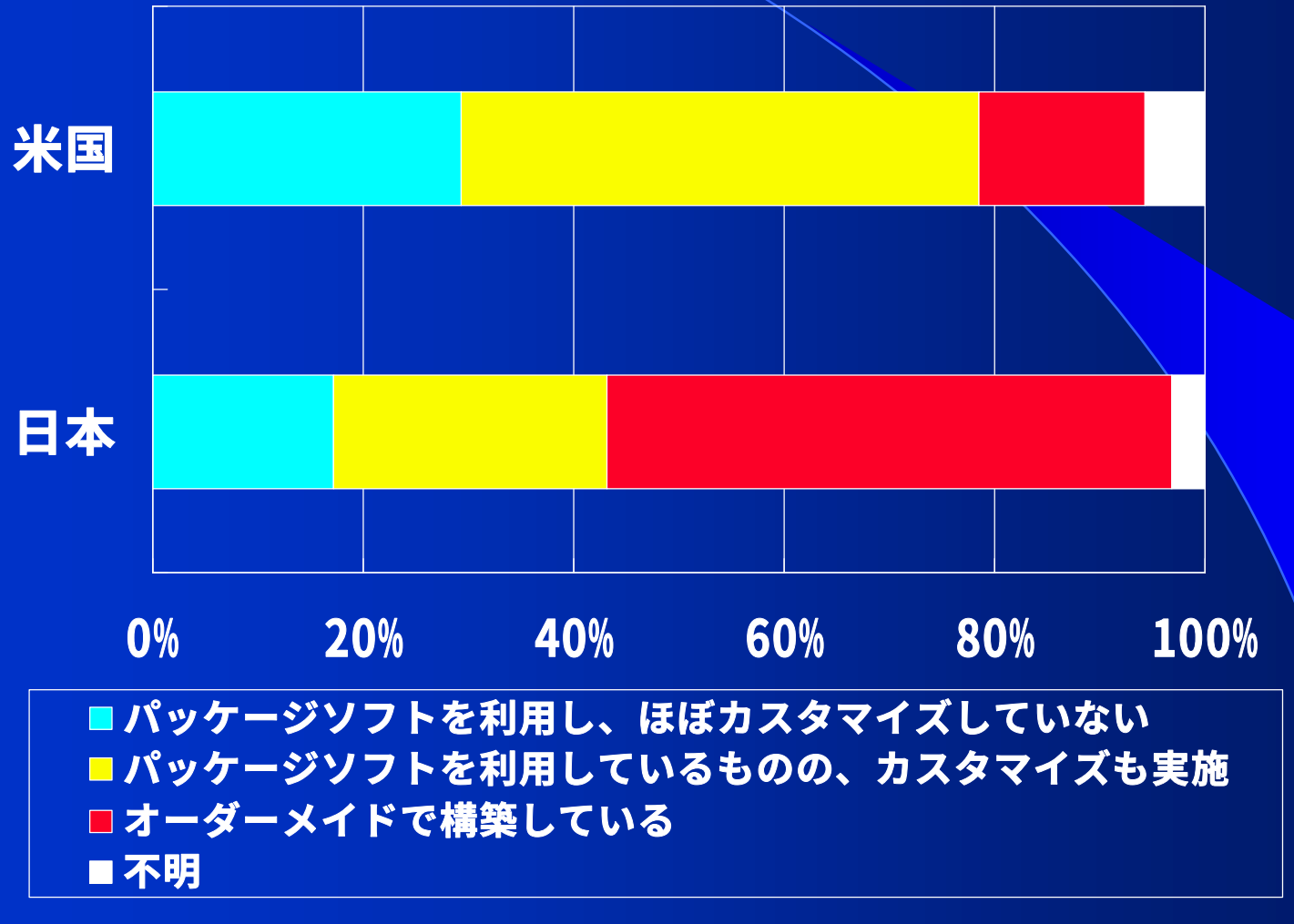
- 「ソフトウェア構築について考えられるもっとも極端な解決策は、まったく自主開発しないことだ」
  （フレデリック・P・ブルックス『人月の神話』p.183）

- 「小規模な再利用（例えば、ライブラリやサブルーチン）は、50年前に始まり、既に解決済みの問題」「大規模な再利用（コンポーネント単位）は、誰もがその重要性を認識しなくてはならないと感じているが、今なお未解決」
  （ロバート・L・グラス『ソフトウェア開発　55の真実と10のウソ』p.71）

- 自分でつくらず、出来合いのものを利用するのが一番

日本企業はパッケージソフトの利用が嫌い？

日米の企業におけるパッケージソフトの利用状況



（出典：「情報通信白書　平成15年度版」　資料編、資料1-2-10、2003年7月）

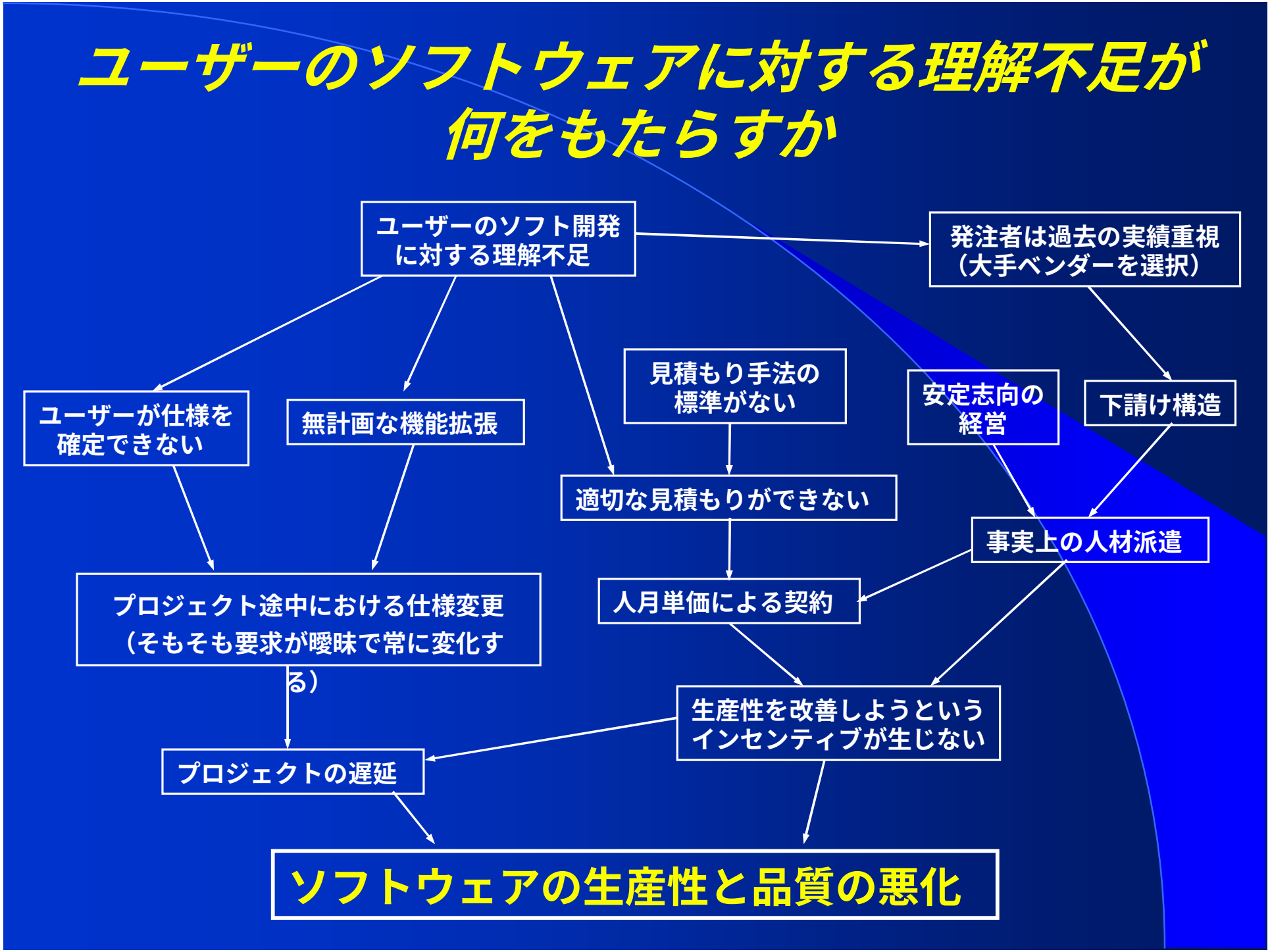
パッケージソフトが普及しなかった理由

1. 日本企業は細部にこだわり、独自性を好む
2. エンドユーザーのリテラシーが低く、パッケージソフトが使えない
3. 日本の情報システムがメーカー毎に細分化されていたために、パッケージソフト市場が小さかった
4. リスクの大きなパッケージソフト開発よりも、受託開発型のビジネスを好む企業が多かった

ユーザー（顧客）の問題点を指摘する声

- プロジェクト開始時点で仕様を確定できない
- プロジェクトの途中で仕様変更を強要してくる
- 仕様の書き方が曖昧で、工数が見積もれない
- 重要な制約条件が仕様書に書かれていなかった
- 見積もった工数を無視した低い金額を提示された

ユーザーのソフトウェアに対する理解不足が何をもたらすか



店が客を育て、客が店を育てる

- 「よいレストランでは、料金に文句を言ってはいけないし、安いレストランでは、味に文句を言ってはいけない！」
- 美味しいレストランがたくさんある街には味のわかる人たちが住んでいる
- よいレストランとそうでないレストランの違いが分からない人たちばかりだと……

4

ソフトウェア産業における
パラダイムシフト

「規律」重視から「アジャイル」重視へ

「ソフトウェア工場」

- Michael Cusumano, “Japan’s Software Factories
  - – A Challenge to U.S. Management ”, Oxford Univ. Press (1991)
- モノの生産管理手法をソフトウェアに適用しようとする試み
  - 欠陥ゼロのソフトウェアを作る。仕事は最初から正しく行った方が、後から修復するより安上がりである。欠陥の発見は早いほどよい
  - 達成基準と実績管理表を使って、一定の品質を実現
  - さまざまなプロジェクトの間で資源を共有する
  - ソフトウェア・モジュールの再利用性を高める
  - プロセスの研究開発を集中化して行う
  - 少人数のグループでの従業員参加を通じて絶え間ない改善に取り組む
  - 製品、品種に逐次改良を加えていく

（参考：ポール・リルランク『ソフトウェア社会進化論』富士通経営研修所（1998）

ソフトウェア工場は「アナロジーの罠」？

- 『アナロジーの罠』　（不適切な比喩は危機をもたらす）
- モノの生産とソフトウェアの生産はまったく異なる

  工場におけるモノの生産は、ソフトウェアではコピー

  ソフトウェアの生産は、常に一品生産
- ソフトウェアの生産とは、企画・設計・開発のプロセス
- ソフトウェアは目に見えず、作業状況の把握・管理が困難
- 「確かにソフトウェア工場というアイディアの持つコンセプトは素晴らしいものかもしれませんが、ここでまたしても肉体労働と知的労働を混乱させてしまっているのです。」

  ピート・マクブリーン『ソフトウェア職人気質』ピアソンエデュケーション（2002）

パラダイムシフトが起きているのではないか

ソフトウェア工場,CMM	アジャイル
ウォータフォール	XP、スクラム
包括的なドキュメント	動くソフトウェア
プロセス重視	結果重視
バグのないソフト	顧客が満足するソフト
顧客は発注者	顧客はチームの一員
綿密なプラン	変化への対応
プロセスとツール	個人とチームワーク
形式知	暗黙知
事前合理性	事後合理性
フォード生産方式	トヨタ生産方式
OSI	インターネット

まとめ

- 「ウォーターフォール・モデルは間違いだ！」
- 個人の能力をより重視すること
  優秀な技術者不足と
  恒常的な残業の悪循環を断ち切る
- ソフトウェア企業（産業）を育てるのはユーザ
  「店が客を育て、客が店を育てる」
- ソフトウェア産業におけるパラダイムシフト
  「規律」重視から「アジャイル」重視へ