ソフトウェアテスト
[2] ソフトウェア工学の概要

Software Testing
[2] Overview of Software Engineering

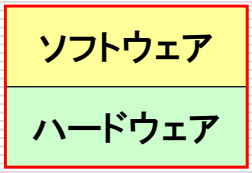
あまん　ひろひさ
阿萬 裕久（AMAN Hirohisa）
aman@ehime-u.ac.jp

ソフトウェアテスト：Software Testing
ソフトウェア工学（こうがく）の概要（がいよう）：Overview of Software Engineering

ソフトウェアの役割

- コンピュータシステムは
  ハードウェアとソフトウェアの組合せ

コンピュータシステム



ハードウェアを使いこなすのが
ソフトウェアの役目である

ハードウェアをどのように
活かすかはソフトウェア次第

ゲームは典型的な例

ソフトウェアの役割（やくわり）: role of software
コンピュータシステム : computer system
ハードウェア : hardware
典型的な例（てんけいてきなれい）: typical example

（参考）2000年までに流通した主な家庭用ゲーム機（専用コンピュータ）

名称	CPU のビット数	メインメモリ	発売年
ファミリーコンピュータ	8ビット	2kバイト	1983年
スーパーファミコン	16ビット	128kバイト	1990年
PlayStation	32ビット	2Mバイト	1994年
セガサターン	32ビット	2Mバイト	1994年
NINTENDO64	64ビット	4.5Mバイト	1996年
ドリームキャスト	32ビット	16Mバイト	1998年
PlayStation2	128ビット	32Mバイト	2000年

ソフトウェアに要求されること

高品質
&
高信頼性
&
低価格

特にテストが重要

- ソフトウェアはハードウェアの能力を引き出して**魅了的なゲーム**となるよう工夫されていた
- 一方で販売後の**不具合の修正は極めて困難**（カートリッジまたはCD/DVDで販売のため）

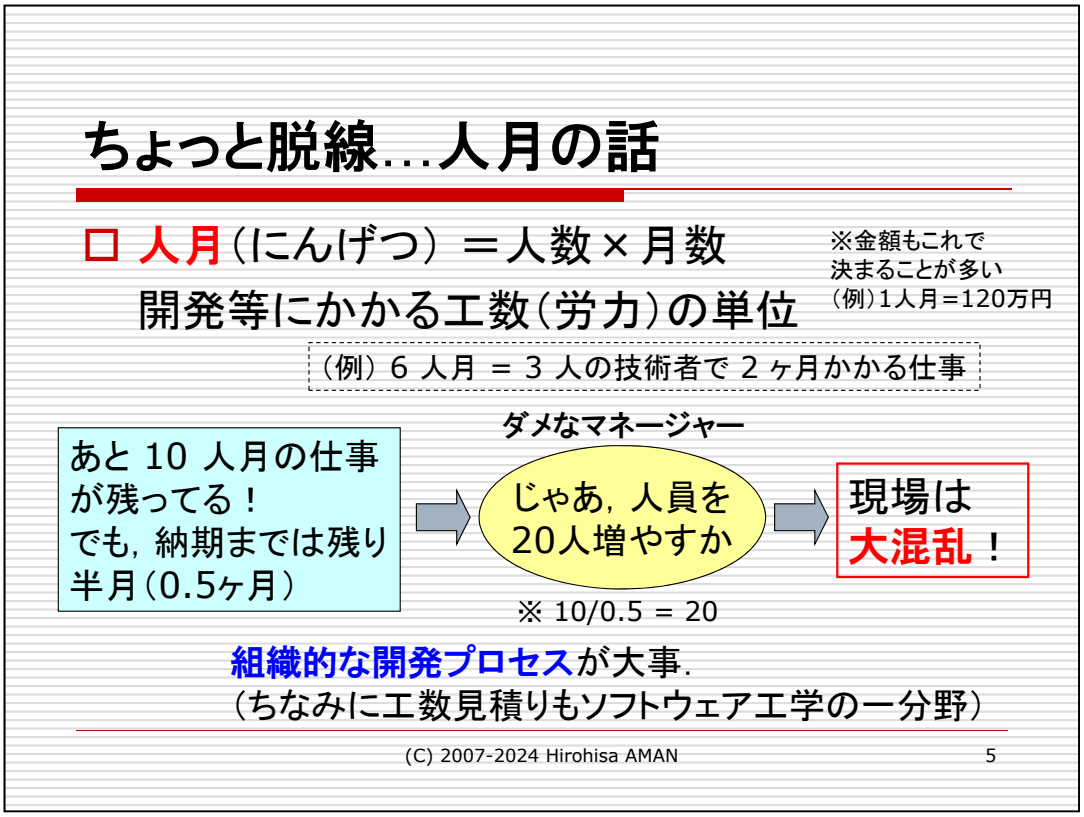
家庭用ゲーム機（かていようゲームき）: video game console, home gaming system
不具合（ふぐあい）: defect, bug, fault
高品質（こうひんしつ）: high quality
高信頼性（こうしんらいせい）: high reliability
低価格（ていかかく）: low price

ソフトウェア工学(Software Engineering)

- **良質のソフトウェアを効率的に開発するための学問分野**
  - 「一部の人たちが**頑張ったらどうにかなる**」という考え方では**ない** ※いわゆるスーパープログラマは別格だが，そういう人はめったにいない
  - 開発をうまくいくようにするための学問

要求分析，設計，プログラミング，テスト等を効率的かつ高品質に行うための理論・技術 ＋ 開発活動をうまく運営するための理論・技術（**開発のマネジメント**）

良質（りょうしつ）：high quality
学問分野（がくもんぶんや）：discipline
効率的（こうりつてき）に開発（かいはつ）する：develop effectively
要求分析（ようきゅうぶんせき）：requirements analysis
設計（せっけい）：design
理論（りろん）・技術（ぎじゅつ）：theory and technique
開発（かいはつ）のマネジメント：management of development



人月（にんげつ）：man-month

大混乱（だいこんらん）：chaos

組織的（そしきてき）な開発（かいはつ）プロセス：well-organized development process

工数見積り（こうすうみつもり）：effort estimation

ソフトウェアの責任は重大

- 日常生活における安心・安全の鍵

ソフトウェアの信頼性

- ソフトウェアの不具合は，深刻な障害をもたらす
  - 電気，ガス，水道，交通，金融すべてに影響の可能性
    （例）証券取引所がストップ → 大損害
  - 事故を引き起こす可能性
    （例）エレベータが誤作動 → 人身事故

さまざまな状況についてテストが必要

深刻（しんこく）な障害（しょうがい）：serious failures
証券取引所（しょうけんとりひきじょ）：stock market
大損害（だいそんがい）：heavy damage
人身事故（じんしんじこ）：fatal accident
さまざまな状況（じょうきょう）：various situations

ソフトウェアの特徴（1／4）

□ 実態を把握しにくい

- ソフトウェアは物理的ではなく論理的な存在
  つまり，**実際に「モノ」としての存在ではない**

- **開発がどの段階まで進んでいるのか，**
  仕様（目的の規格）通りに作られているのか，
  などを把握するのは難しい

これと対照的なのが建設で，
工事の進み具合いは外から
見てもおおよそ把握できる

特徴（とくちょう）：characteristics
実態（じったい）を把握（はあく）：understand the situation (development progress)
物理的（ぶつりてき）：physical
論理的（ろんりてき）：logical
段階（だんかい）：stage
仕様（しよう）：specification

ソフトウェアの特徴（2／4）

□ 開発工程に作業が集中

- ハードウェアの場合，製品を開発し，それを工場等で製造するという流れになる
- ソフトウェアの場合，製造は単なるコピー作業

- 品質もコストも開発工程が中心
- ✓ ハードウェア：製造工程で不良品を作る心配
- ✓ ソフトウェア：不良品＝「最初から全部が欠陥品」
  → **開発で失敗しないことがすべて**

冒頭でふれたゲーム等はその典型的な例

開発（かいはつ）：development
製造（せいぞう）：manufacture
不良品（ふりょうひん），欠陥品（けっかんひん）：defective product

ソフトウェアの特徴（3／4）

□ 運用・保守の期間が長い

作るのにかかった時間よりも，ソフトウェアの

- **運用**（実際にユーザが使う）
- **保守**（機能の修正や改良，拡張を行う）

を行う期間の方が**はるかに長い**

- 物理的な消耗は無く，いったん使われるようになるとその後は「長いつきあい」になることも

保守（メンテナンス）のしやすさがポイントになる

運用（うんよう）：operation
保守（ほしゅ）：maintenance
物理的（ぶつりてき）な消耗（しょうもう）：consumption

保守のしやすさ(保守性)の重要性

- 例えば，運用・保守が 20 年に及ぶとする

  作った人たちがずっと面倒を見るわけではない

  → やがて別の人たちが保守していくことになる

  ゆえに，保守性の高いソフトウェアにしておくべき

「レガシーシステム」といって，20年・30年前から使われているシステムも世の中には多くある．
「いま動いているものは下手にいじりたくない」という考えが現場には根強い．

(例)COBOL で書かれた金融システム

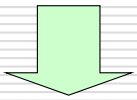
面倒(めんどう)を見る : care

レガシーシステム : legacy system

ソフトウェアの特徴（4／4）

□ 再利用が少ない

- ✓ ハードウェア： 既存の部品を再利用（流用）していくことが一般的
- ✓ ソフトウェア： 部品を組み合わせるだけで完成させることは難しい（いろんなカスタマイズが必要）



- ■ ライブラリのかたちで再利用可能なソフトウェアはあるが，プログラミング無しで使えるわけではない

再利用（さいりよう）：reuse
カスタマイズ：customize
ライブラリ：library

再利用

- ソフトウェアの再利用は大きく分けて2種類
  - **ブラックボックス的**
    中身を気にする必要がなく，部品感覚で使う．
  - **ホワイトボックス的**
    プログラムを「コピー＆ペースト」し，必要に応じて書き換える．

ブラックボックス的（ブラックボックスてき）：black box like
部品（ぶひん）：parts
ホワイトボックス的（ホワイトボックスてき）：white box like
コピー＆ペースト：copy and paste

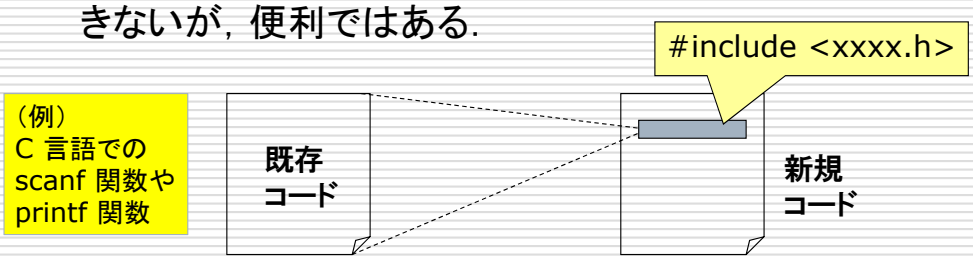
ブラックボックス的な再利用

- ライブラリのかたち

  例えば，既存の関数をそのまま利用して，

  ソースコードそのものの内容は問題にしない．

  あらかじめ決められた使い方（マニュアル通り）しかできないが，便利ではある．



関数（かんすう）：function

そのまま：as is

ホワイトボックス的な再利用

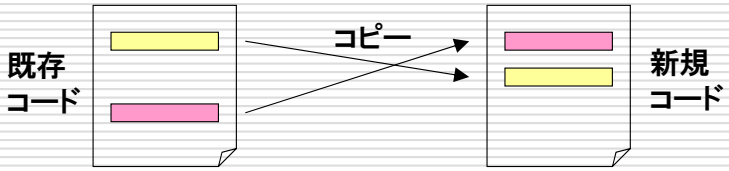
- ソースコードを「コピー＆ペースト」するかたち

  例えば，既存のソースコードの一部を流用．

  本や Web ページに掲載されているコードを転記．

  「良いコード」をお手本としてコードを作成する場合は適切だが，逆にミス（バグ）をコピーしたり，誤用の恐れもある．

  （例）自分が理解できていないプログラムをコピペしても混乱するだけ



既存（きぞん）：existing

お手本（おてほん）：good example

誤用（ごよう）：wrong use, misuse

【演習1】
コードクローンの問題点を考察せよ

- ソースコードをそのままコピー＆ペーストしたもの，あるいは，一部を変更して流用したものを「**コードクローン**（Code Clone）」という
- ソフトウェア産業において，コードクローンの多いソフトウェアは問題視されている
  それは何故だろうか？

問題点（もんだいてん）： problem, issue
考察（こうさつ）： consider
一部（いちぶ）を変更（へんこう）して流用（りゅうよう）： modify and reuse a part of …
ソフトウェア産業（さんぎょう）： software industry
問題視（もんだいし）される：be considered as a problem
何故（なぜ）: why

【演習1】
（解説： 修正作業のコストとリスクの観点から）

あるコードの一部（数行程度）が**50ヶ所にコピー＆ペースト**されていたとする.

運用の途中で問題が発生したり，機能向上の要望が出てきたりする関係で，**コードの修正が必要なことは多い**.

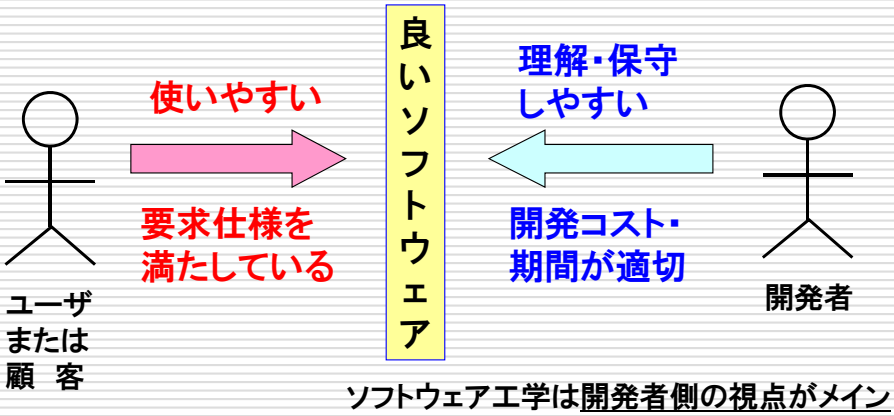
- ✓ もしも，修正箇所がコードクローンであったとすると，**他の50ヶ所も全部修正しなければならない**（**見落としは許されない**）

※最初から，そこを「関数」にしておけば修正は1ヶ所で済むのに. . .

解説（かいせつ）：expository
修正作業（しゅうせいさぎょう）: modification activity
観点（かんてん）: point of view
数行程度（すうぎょうていど）: about a few lines
運用（うんよう）の途中（とちゅう）: during the operation
問題が発生（もんだいがはっせい）: a problem occurs
機能向上（きのうこうじょう）: functionality enhancement
要望（ようぼう）: demand
コードの修正 : modification of source code
修正箇所（しゅうせいかしょ）: changed part
見落とし : miss, overlooking

良いソフトウェアとは

- 視点によって概念が異なる点に注意



ソフトウェア工学は開発者側の視点がメイン

視点（してん）: viewpoint
概念（がいねん）: concept
異なる（ことなる）: differ
顧客（こきゃく）: customer
使いやすい: easy to use
要求仕様（ようきゅうしよう）: requirements specification
満たす（みたす）: satisfy
開発者（かいはつしゃ）: developer
理解（りかい）: understanding, comprehension
保守（ほしゅ）: maintenance
開発コスト（かいはつコスト）: development cost
期間（きかん）: development period
適切（てきせつ）: appropriate

ユーザ側の視点(1) 要求仕様の満足

- 顧客(依頼者)からの**要求仕様を満足**していること

  要求仕様は 2 種類に大別される:

  - **機能仕様**: **実現すべき機能**

    (例)ユーザ認証ができること ※「どのように」は別として

  - **非機能仕様**: 要求を**「どのように」実現**すべきか

    (例)500ユーザの認証が 3 秒以内に完了すること

  ※機能そのものについてではなく, 性能やセキュリティといった実現の仕方

顧客(こきゃく): customer, client
大別(たいべつ)される: be classified roughly
機能仕様(きのうしよう) : functional requirements
実現(じつげん)すべき機能(きのう): the functionality you should implement
どのように: how
別として:apart from
非機能仕様(ひきのうしよう) : non-functional requirements
認証(しょうにん) : authentication
性能(せいのう): performance
セキュリティ: security

ユーザ側の視点(2) 操作性

- ひとことで言えば「**使いやすさ**」
  - 操作がやりやすい
  - 操作が分かりやすい
  - 間接的には，動作速度やメモリ使用量も関係
    (いわゆる「重い」システムは使いたくない)

操作性 : usability

開発者側の視点（1） コストと期間

- 開発コストは低い方が望ましい
- 開発期間は期限内（納期まで）に
  - ゆっくり時間をかけて開発できることは珍しい
  - 限られた人材と時間で期限内に仕上げる必要

安価で開発できて期限内に納入でき，なおかつ要求仕様を満たしたソフトウェアの開発が目標

人材（じんざい）：staff
開発期間（かいはつきかん）：development period
納期（のうき）：delivery period
安価（あんか）：low price
限られた（かぎられた）：limited
仕上げる（しあげる）：finish
要求仕様（ようきゅうしよう）：requirements
目標（もくひょう）：goal

開発者側の視点(2) 保守性

- いったん完成し，運用に入ると，**必ずと言ってよいほど要求仕様は変化**する
  - **細かい修正**の要求「ここはこう変えて欲しい」
  - **機能の追加・拡張**の要求「もっとこんな機能も」
- **不具合(バグ)が見つかる**こともある

保守しやすいソフトウェアは「良い」ソフトウェア
別人が保守することもあるので「分かりやすさ」も大事

保守性(ほしゅせい) : maintainability
必ず(かならず)と言ってよい:almost always
変化(へんか): change
細かい修正(こまかいしゅうせい): minor modification
追加(ついか)・拡張(かくちょう): addition and expansion
保守(ほしゅ)しやすい: easy to maintain
別人(べつじん): other people (people other than the developer)
分かりやすさ: understandability, comprehensibility

開発現場（かいはつげんば）: development field
製造業（せいぞうぎょう）: manufacture
勘（かん Kan）: feeling, intuition
経験（けいけん Keiken）: experience
度胸（どきょう Dokyo）: courage, guts

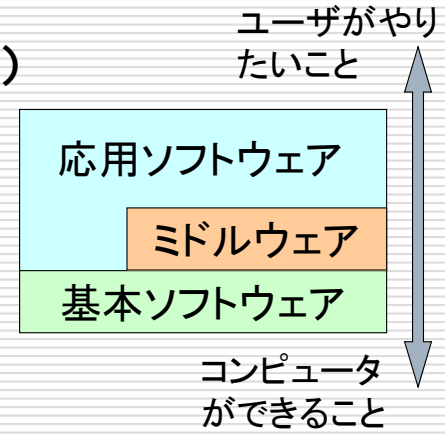
10-minutes rest break

10分休憩

ソフトウェアの分類

- 基本ソフトウェア
- 応用(アプリケーション)ソフトウェア
- ミドルウェア
- 組込みソフトウェア

これら3層がハードウェアに組込まれたイメージ



分類(ぶんるい): classification
基本(きほん)ソフトウェア : basic software
応用(おうよう)ソフトウェア : application software
ミドルウェア : middleware
組込み(くみこみ)ソフトウェア : embedded software

基本ソフトウェア

□ コンピュータの利用やソフトウェアの実行を行う上でその基礎となるソフトウェア

- OS(オペレーティングシステム)

単に「基本ソフトウェア」と言うとこれを意味していることが多い

- コンパイラ/インタプリタ
- サービスプログラム(ユーティリティ)

(例)ファイル編集・検索

基本ソフトウェアが**何か特定の業務を行えるわけではない**が，それらが無いと**話が始まらない**

(例)OS (Windows等)がインストールされていれば何でもできるというわけではない．
しかし，そもそもインストールされていないと何も動かない．

利用(りよう) : use
実行(じっこう) : execution, run
ユーティリティ : utility
ファイル編集(ファイルへんしゅう) : file edition
ファイル検索(ファイルけんさく) : file search
特定(とくてい)の業務(ぎょうむ): a specific task
話(はなし)が始まらない(はじまらない): we can do nothing

つまり「アプリ」

応用(アプリケーション)ソフトウェア

- 特定の用途をもったソフトウェア
  - 例えば,
    - 証券取引管理 (業種別ソフトウェア)
    - 予算管理 (業務別ソフトウェア)
    - 表計算, プレゼンテーション (共通応用ソフトウェア)
  - ソフトウェアがある種の仕事や目的を果たすための道具になっている

特定(とくてい)の用途(ようと): a specific use
証券取引管理(しょうけんとりひきかんり) : stock transaction management
業種(ぎょうしゅ) : business type
予算管理(よさんかんり) : budget management
業務(ぎょうむ) : business assignment, operation
表計算(ひょうけいさん) : spreadsheet
ある種の(あるしゅの): a kind of
仕事(しごと): task
目的(もくてき): goal
果たす(はたす): achieve
道具(どうぐ): tool

ミドルウェア

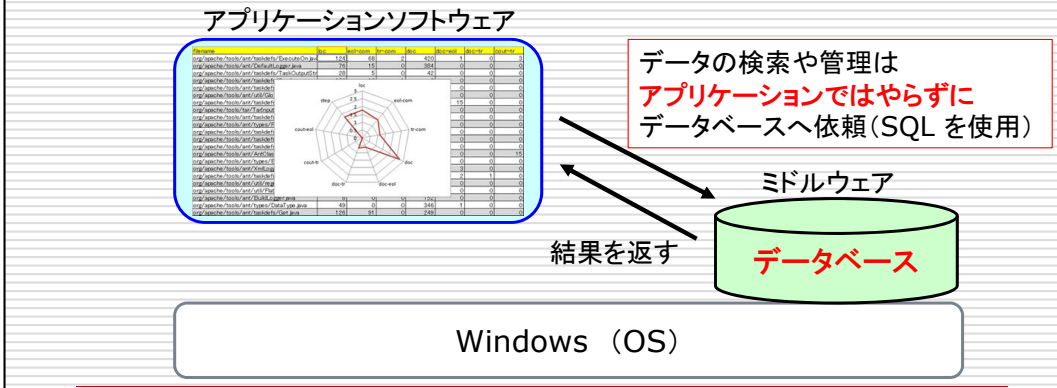
- 基本ソフトウェアと応用ソフトウェアの中間
  - 比較的新しい概念であり，**応用ソフトウェアの基盤**となるようなソフトウェアのこと
  - 例えば，
    - データベース管理システム
    - Webサーバ
    - アプリケーションサーバ
  - OSの機能を直接使う（システムコール等）のではなく，その上位の概念で利用：例えば SQL で

ユーザが直接使うソフトウェアというよりも「アプリのための」ソフトウェア

ミドルウェア：middleware
比較的（ひかくてき）新しい（あたらしい）概念（がいねん）：a relatively new concept
基盤（きばん）：base, platform
データベース管理（かんり）システム：database management system
機能（きのう）：functionality
直接使う（ちょくせつつかう）：use it directly
等（など）：etc.
上位（じょうい）の概念（がいねん）：higher level concept
アプリのためのソフトウェア: software for an application software

ミドルウェアの位置付け

- 例えば Windows 上で大量のデータを処理するアプリケーションがあるとする



位置付け（いちづけ）: positioning
大量（たいりょう）のデータを処理（しょり）: processing a massive amount of data
検索（けんさく）: search
管理（かんり）: management
データベース : database
依頼（いらい）: ask, request

組込みソフトウェア

- **特定の機能を実現する電子機器において，主に機器制御を行うためのソフトウェア**

（※汎用的なコンピュータ上で動作するのではなく，**特定の機器に組込まれている**）

- 例えば，
  電化製品，**カーナビ**，**エレベーター**等
- **ハードウェアと一体**になって供給されている
  - 手軽に修正版をインストールとはいかない
  - 動作環境は多様で，かつ，**高い信頼性**が求められる

機器制御（ききせいぎょ）：machine control
汎用的（はんようてき）：general
特定の（とくていの）：specific
電化製品（でんかせいひん）：home electronics
カーナビ：car navigation system
供給（きょうきゅう）：supply
手軽に（てがるに）：easy
修正版（しゅうせいばん）：fixed version
動作環境（どうさかんきょう）：platform
多様（たよう）：various
高い（たかい）信頼性（しんらいせい）：high relaiability

【演習2】組込みソフトウェアの開発や保守における課題を挙げよ

- エレベーター制御の組込みソフトウェアを対象として考える
- このソフトウェアを開発する場合，一般のソフトウェアに比べて何が難しいか？
- また，保守する場合はどうか？

エレベーター : elevator
制御（せいぎょ） : control
開発（かいはつ）: development
一般の（いっぱんの）: general
保守（ほしゅ）: maintenance

【演習2】(解答例)

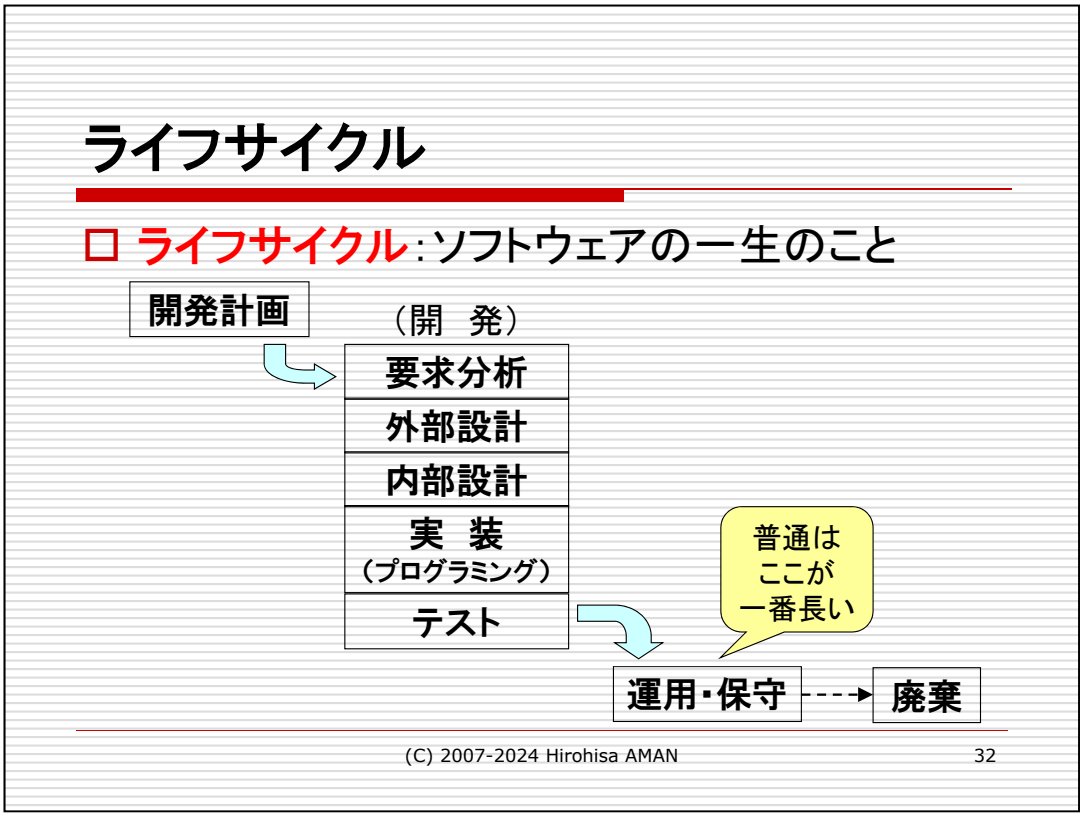
【開発での課題】

- **信頼性**を極限まで高める必要がある
  - 暴走は許されない
  - 安全で安定した動作が常に求められる
- **ハードウェアや環境の影響**が大きい
  - 夏場は高温・多湿に
  - 摩耗や物理故障もある

【保守での課題】

- **修正が容易でない**
  - エレベータを止めなければならない
  - ハードウェアの入れ替えは難しい(例えば,CPUやメモリの追加は簡単にはできない)
- **運用期間が長い**
  - 10年20年は当たり前

課題(かだい): problem, issue, challenge
信頼性(しんらいせい) : reliability
暴走(ぼうそう) : runaway
環境(かんきょう) : environment
高温(こうおん) : high temperature
多湿(たしつ) : high humidity
摩耗(まもう) : wear, attrition
物理故障(ぶつりこしょう) : physical failure
修正(しゅうせい) : fix, repair
容易(よういい)でない: not easy
運用期間(うんようきかん) : operation period
当たり前(あたりまえ): ordinary



ライフサイクル：lifecycle
開発計画（かいはつけいかく）：development plan
要求分析（ようきゅうぶんせき）：requirements analysis
外部設計（がいぶせっけい）：external design
内部設計（ないぶせっけい）：internal design
実装（じっそう）：implementation
テスト：testing
運用（うんよう）：operation
保守（ほしゅ）：maintenance
廃棄（はいき）：discard

ソフトウェア危機（Software Crisis）

- 1968 年に提唱された危機感のこと

ソフトウェア開発がニーズに追いつかず，
コンピュータシステムの発展を妨げてしまう

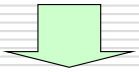
- ソフトウェア開発が間に合わず，
  コンピュータの進展を妨げる
- 巨大化 → **バグ多発** → **社会的問題に発展**
- **開発コストの増大**

要は，**ソフトウェアがネック**になる！

危機感（ききかん）: sense of crisis
発展（はってん），進展（しんてん）: evolution, progress
巨大化（きょだいか）: getting larger
多発（たはつ）: rash
社会的問題（しゃかいてきもんだい）: social concern
開発コスト（かいはつコスト）: cost of development
ネック: bottleneck

(1)コンピュータの進展を妨げる

- いろんなことを「**コンピュータにやらせよう**」という方向になってきた(当時の話)



- **ハードウェアは汎用的なものを安く大量生産**させ，**細かい対応はソフトウェア**にやらせよう

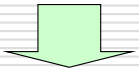
  でもソフトウェアは基本的に「**手作り**」なので

  - 開発が間に合わない！
  - 技術者が足りない！

汎用的(はんようてき) : general
大量生産(たいりょうせいさん) : mass production
細かい対応(こまかいたいおう): detailed tuning
手作り(てづくり) : hand-made, manual
間に合わない(まにあわない): miss the delivery deadline
技術者(ぎじゅつしゃ): engineers
足りない(たりない): a lack of …

(2)社会的問題へ発展の恐れ

- システムへのニーズが高まり，ソフトウェアの巨大化・複雑化が進む



- 当然，人為的な誤り（いわゆるバグ）が生じるリスクも高まる

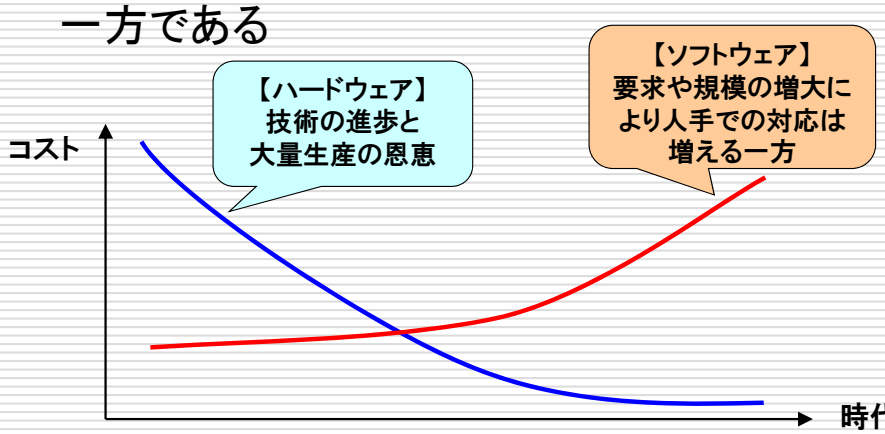
システムとバグの種類によっては，社会生活に**深刻なダメージ**を及ぼすことも考えられる

電気，ガス，水道，交通，金融等に影響する可能性

社会的問題（しゃかいてきもんだい）：social problem
ニーズ：need
巨大化（きょだいか）：getting larger
複雑化（ふくざつか）：getting more complex
人為的な誤り（じんいてきなあやまり）：human error
深刻（しんこく）なダメージ：serious damage
電気（でんき）：electricity
ガス：gas
水道（すいどう）：running water
交通（こうつう）：transportation
金融（きんゆう）：finance

(3)コストの増大

- ソフトウェア開発・保守にかかるコストは増える一方である



コスト : cost
進歩(しんぽ) : progress, improvement
大量生産(たいりょうせいさん) : mass production
要求(ようきゅう) : requirements
規模(きぼ) : size, scale
時代(じだい) : period, year
人手(ひとで): man power

「ソフトウェア工学」の誕生

- そういった問題の解決に向けた対策が必要

良質なソフトウェアを効率的に開発するための理論や技法を研究（それまではどちらかと言えば「職人技」に近い）

- ソフトウェア開発に関する理論・技法を確立し，「工学」（ものつくりの学問）として体系化

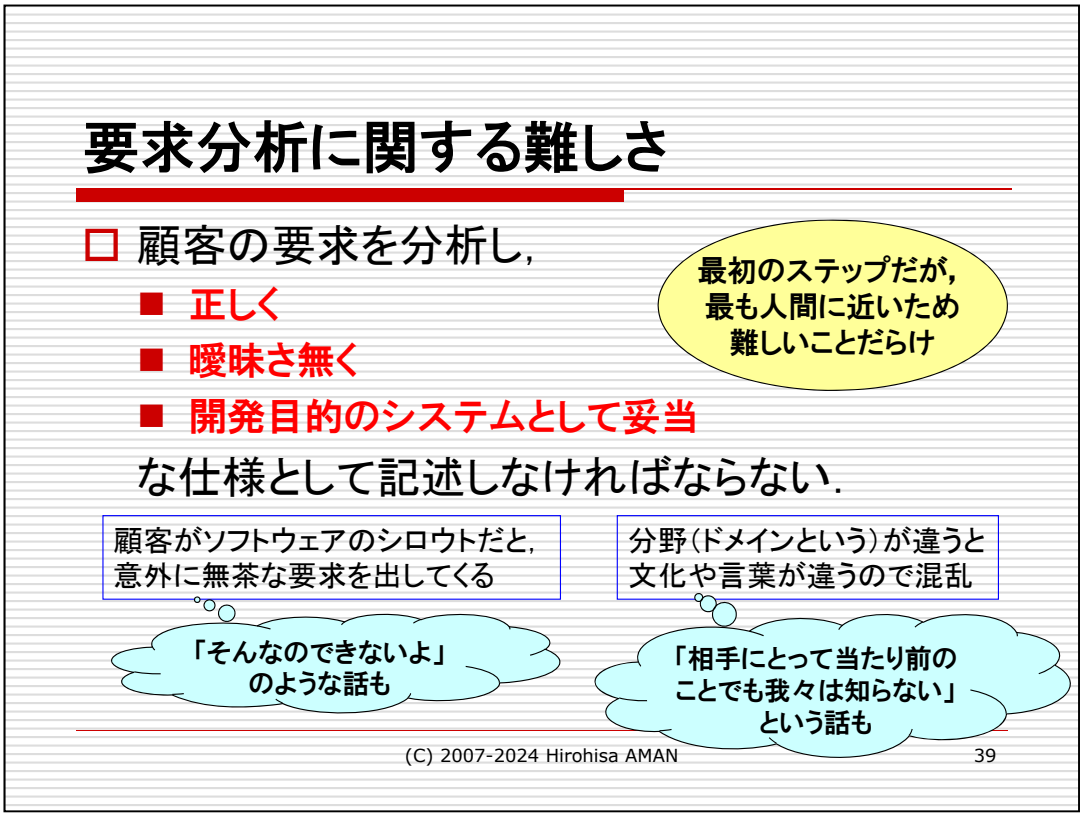
高い生産性と高品質を！

ソフトウェア工学（こうがく）: software engineering
誕生（たんじょう）: birth of …
解決（かいけつ）: resolve
対策（たいさく）: countermeasure
良質（りょうしつ）: high quality
効率的（こうりつてき）: efficiently
理論（りろん）: theory
技法（ぎほう）: technique
職人技（しょくにんわざ）: craftmanship
確立（かくりつ）: establish
体系化（たいけいか）: schematization
生産性（せいさんせい）: productivity

現在・将来の課題

- いろんな研究や技術革新を経て，ソフトウエア産業は発展してきているが，依然として難しい課題は残っている
  - **要求分析**に関する難しさ
  - **再利用**の難しさ
  - **プロジェクト管理**の難しさ
  - **見積り**の難しさ

現在（げんざい）: current
将来（しょうらい）: future
課題（かだい）: issue
技術革新（ぎじゅつかくしん）: technical innovation
依然（いぜん）として: still



要求分析（ようきゅう分析）: requirements analysis
顧客（こきゃく）: customer
曖昧さ（あいまいさ）: ambiguity
妥当（だとう）: valid
仕様（しよう）: specification
シロウト: amateur
無茶な要求（むちゃなようきゅう）: reckless requirement
ドメイン: domain
文化（ぶんか）: culture, custom
言葉（ことば）: language, terminology
当たり前（あたりまえ）のこと: common knowledge

再利用の難しさ

- ソースコードの再利用
  - 比較的実践しやすいが，**安易なコピー＆ペースト**はバグの原因にも → **コードクローン**
  - **どのコードを使えばよいのか？**
- 設計やアーキテクチャの再利用
  - コードよりも抽象的なレベルでの**「知識・ノウハウ」の再利用**であり，ハードルはそれなりに高い
  - **デザインパターン**，**フレームワーク**の知識が必要

再利用（さいりよう）：reuse
比較的（ひかくてき）実践（じっせん）しやすい：easier to practice
安易な（あんいな）：easy, flimsy
抽象的（ちゅうしょうてき）：abstract
知識（ちしき）：knowledge
ノウハウ：know-how
ハードルはそれなりに高い：relatively high hurdle

プロジェクト管理の難しさ

- 開発プロジェクトの管理では
  - **進捗(進み具合)管理**
  - **品質管理**
  - **人員配置, コミュニケーション**

  といった項目がプロジェクトマネージャの課題

  例えば，結局は一部の人たちが右往左往しているだけであったりして，全体で効率的な開発ができていなかったりする

プロジェクト管理 : project management
進捗管理 : progress management
品質管理 : quality control, quality management
人員配置 : staff assignment
右往左往する : run about in confusion

見積りの難しさ

- 現実には，まだまだ属人性が大きい
- つまり，個々のエンジニアの経験や能力に依存するところが大きい
- そこから工数や期間，コストを見積ることは，言うまでもなく困難　→　**遅延・コスト超過**
  - **工数見積りモデル**（COCOMO）
  - 開発組織の**成熟度モデル**（CMM，CMMI）
  - **統計**や**ニューラルネット**等による予測

見積り : estimation
属人性 : personality
個々のエンジニアの経験や能力 : each engineer’s experience and skill
工数 : effort
期間 : period
コスト : cost
遅延 : delay
コスト超過 : cost overrun
工数見積りモデル : effort estimation model
成熟度 : maturity level
統計 : statistics
ニューラルネット : neural network
予測 : prediction

まとめ

- ソフトウェア工学とは，ソフトウェアの作り方だけでなく，**開発プロジェクトを成功へと導く**ための**管理（マネジメント）の学問**でもある
- **ソフトウェア危機**をきっかけとして生まれた（その意味で**産業界との結びつき**が強い）
- ソフトウェア産業は発展してきているが，まだ解決すべき課題は多い：**要求分析**，**再利用**，**プロジェクト管理**，**見積り**等

まとめ：summary
開発（かいはつ）プロジェクト：development project
成功（せいこう）へと導く（みちびく）：lead to success
産業界（さんぎょうかい）：industry
発展（はってん）：developing, evolving
解決（かいけつ）すべき課題（かだい）：challenge
要求分析（ようきゅうぶんせき）：requirements analysis
再利用（さいりよう）：reuse
プロジェクト管理（かんり）：project management
見積り（みつもり）：estimation

宿題(homework)

“[02] quiz”に答えなさい（今週の金曜日まで）

Answer “[02] quiz”
by this Friday 23:59

注意：quiz のスコアは成績の一部となります
（Note: Your quiz score will be a part of your final evaluation）