Software Design
1. Introduction

Natsuko Noda
nnoda@shibaura-it.ac.jp

Copyright© Natsuko NODA, 2014-2024

ソフトウェア設計論
1. はじめに

野田夏子
nnoda@shibaura-it.ac.jp

Copyright© Natsuko NODA, 2014-2024

At first...

Copyright© Natsuko NODA, 2014-2024

はじめに

Copyright© Natsuko NODA, 2014-2024

About myself, Natsuko NODA

- Major: Software engineering
  - "How can we develop right software efficiently?"
- Bio in short: first, researcher in a private company (NEC), and then faculty member of SIT
- Recently care about… : How can we apply software engineering to solve various social problems?
- Favorite word: Even if the world would go to pieces tomorrow, you would still plant your apple tree.

Copyright© Natsuko NODA, 2014-2024

担当教員自己紹介

- 専門分野：ソフトウェアエ学
  - 「正しいソフトウェアを効率よく開発するには？」
- 経歴(超短縮版)：民間企業(NEC)から本学教員へ
- 気になっていること：どのようにソフトウェアエ学を応用して様々な社会問題を解決することができるか
- 好きな言葉：明日、世界が滅びるとしても
  今日、あなたはリンゴの木を植える

Copyright© Natsuko NODA, 2014-2024

Objectives

- Main objectives
  - To understand the basics of software design.
  - To read correctly documents described in UML (unified modeling languages).
    - and to draw correctly simple diagrams using UML.
  - To understand methods of describing various aspects of software.
- Hidden objective
  - To communicate people from different cultures, in order to become an engineer who works globally!

Copyright© Natsuko NODA, 2014-2024

達成目標

- 主目標
  - 設計を中心に、ソフトウェア開発における基本的な考え方を理解する。
  - オブジェクト指向によるソフトウェア開発成果物の統一表記法であるUML（Unified Modeling Language）を読むことができる。
  - ソフトウェアの様々な側面について適切な記述方法で記述する方法を理解する。
- 隠された目標
  - グローバルなエンジニアになるために、様々な文化背景を持った人たちと交流する

Copyright© Natsuko NODA, 2014-2024

Evaluation

- Quiz after every lecture 36%
  - "Quizzes" on ScombZ
  - Answer the quiz at the beginning of the next class.
  - Do not refer to any materials.
  - Retakes of the quiz will be allowed only in case of network problems.
- minute paper of every class 24%
  - "minute paper" on ScombZ "Assignments"
  - You should submit the minute paper between the end of class and 23:00 (JST) day after next.
  - Download the template ("minutePaper.docx") and use it.
- Final exam 40%
  - At the 14th class

Copyright© Natsuko NODA, 2014-2024

評価方法

- 毎授業後のミニテスト(Quiz) 36%
  - ScombZ上のテスト "Quiz"
  - 翌週のクラスの開始時に受ける
  - 資料は何も見ないこと
  - 再受験はネットワークトラブル時のみ
- 毎授業後に提出するミニットペーパー(minute paper) 24%
  - ScombZ上の課題 minute paper
  - 授業終了時間後から翌々日23時までの間に提出
  - テンプレート ("minutePaper.docx") をダウンロードして利用のこと
- 期末試験 40%
  - 第14回に実施

Copyright© Natsuko NODA, 2014-2024

Software

Copyright© Natsuko NODA, 2014-2024

ソフトウェア

cf. SE-J, 1.1

Copyright© Natsuko NODA, 2014-2024

What is software?

- The entire set of programs, data, and related documentation that is used to make a computer work.
  - Software ≒ Program
    - Very similar, but not equal
    - Software includes programs
  - Software (NG: a software, softwares !)
    - uncountable noun
  - In contrast with hardware

Copyright© Natsuko NODA, 2014-2024

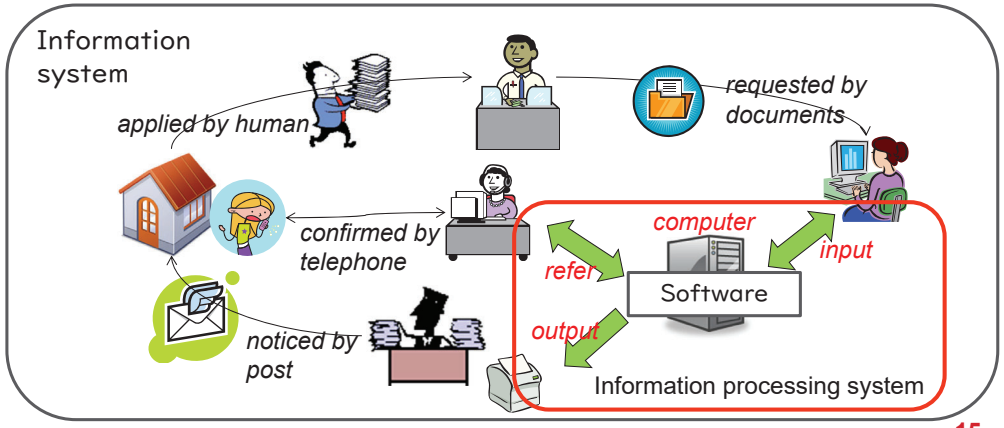
ソフトウェアとは何か

- コンピュータに仕事をさせるために必要なプログラム、データ、関連するドキュメント等
  - ソフトウェア≒プログラム
    - しかし全くの同値ではない
    - ソフトウェアにはプログラムも含まれる
  - Software
    - 不可算名詞
  - ハードウェアに対して、ソフトウェア

Copyright© Natsuko NODA, 2014-2024

Information system and software

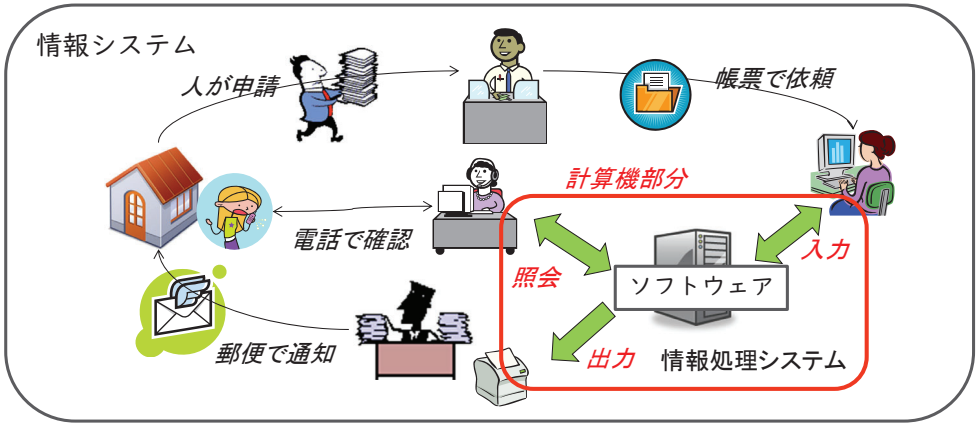
- Information system: mechanism to process, store, and exchange information, which is constructed by humans, machines, and computers.



Copyright© Natsuko NODA, 2014-2024

情報システムとソフトウェア

- 情報システム：人、機械、コンピュータなどから構成される情報の処理、蓄積、交換をする仕組み



Copyright© Natsuko NODA, 2014-2024

Various aspects of software

- Where is software used?
- How big is software?
  - How big? How do we measure the software?
- How is software developed?
  - How many people are involved?
  - How long does it take to develop software?
  - …

Copyright© Natsuko NODA, 2014-2024

ソフトウェアの様々な側面

※ No Japanese page. Simple English!
(易しい英語のため、該当する日本語ページはありません)

Copyright© Natsuko NODA, 2014-2024

Characteristics of software

- Software is present everywhere!
  - e.g.; in bank systems, e-learning systems, electrical rice cookers, mobile phones, power plants, …
- Variety of software
  - Various target domain: business, communication, factory automation, …
  - Various running environment: PC, workstation, home electronics, car, …
  - Various development style: order made, packaged software for mass customers, end-user development, …
  - Various users: general users, engineers, equipment (in cases of embedded software), …

Copyright© Natsuko NODA, 2014-2024

ソフトウェアの特徴・性格

- あらゆるところに存在
  - 例：銀行システム、eラーニングシステム、炊飯器、ケータイ、発電所、… （未分類）
- 様々な種類
  - 対象とする分野：ビジネス分野、通信分野、工場自動化、…
  - 稼働環境：パソコン、ワークステーション、家電、自動車、…
  - 開発の仕方：受注型、企画型、エンドユーザ開発、…
  - ユーザ：一般ユーザ、技術者、機器(組込みソフトウェアの場合)、…

Copyright© Natsuko NODA, 2014-2024

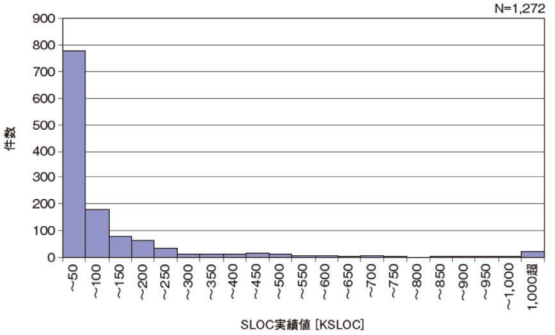
Scale of software

- Scale, or size of software is often described by lines of code.
- Roughly speaking,
  - 100 lines: exercise of programming
  - 1,000 lines: very compact
  - 10,000 lines: small
  - 100,000 lines: middle size
  - 1,000,000 lines-: large. Difficult to develop.
    - Mobile phone, bank system, space shuttle, ...

Copyright© Natsuko NODA, 2014-2024

Scale of software

- Software scale by lines of code



basic statistics

[KSLOC]

	Minimum		Middle		Maximum	Average	standard deviation
N	最小	P25	中央	P75	最大	平均	標準偏差
1,272	0.0	11.5	32.5	98.9	6,600.0	134.6	416.4

N = 1,272 (未回答 : 203 件)

By IT Knowledge Center on emerging tech trends, Information-technology Promotion Agency, Japan (IPA) 2018

Copyright© Natsuko NODA, 2014-2024

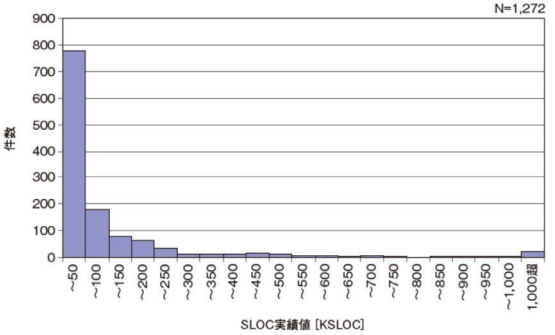
ソフトウェアの規模

- 規模は行数で表すことが多い
- おおざっぱな感覚
  - 100行：プログラムの練習
  - 1,000行：かなりコンパクト
  - 10,000行：小規模
  - 100,000行：中規模
  - 1,000,000行：大規模。相当な覚悟が必要
    - 携帯電話
    - 銀行のオンラインシステム
    - スペースシャトル

Copyright© Natsuko NODA, 2014-2024

ソフトウェアの規模

- コード行数で表したソフトウェアの規模



基本統計量

[KSLOC]

N	最小	P25	中央	P75	最大	平均	標準偏差
1,272	0.0	11.5	32.5	98.9	6,600.0	134.6	416.4

N = 1,272 (未回答 : 203 件)

出典：「ソフトウェア開発データ白書2018-2019」独立行政法人情報処理推進機構社会基盤センター(2018)

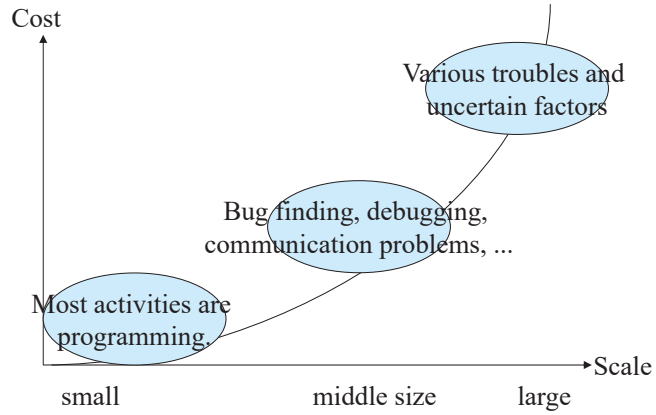
Copyright© Natsuko NODA, 2014-2024

Development cost: not proportional to size

- The larger in scale, the higher the cost.
  - It exponentially increases.

Cost = Time (development time duration) and Humans (developers).

Image:

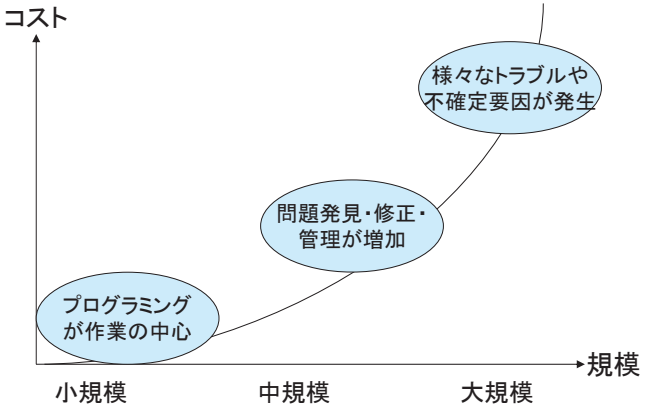


Copyright© Natsuko NODA, 2014-2024

開発コストは規模に比例しない

- 大規模になればなるほどコストが膨らんでいく
  - コスト＝開発期間と必要な人間

イメージ:



Copyright© Natsuko NODA, 2014-2024

Social impact

- Software has a serious impact on our daily life.
  - In a good way, and in a bad way
  - Software becomes essential to modern life.
  - At the same time, software sometimes raises serious problems and we get confused.
    - Do you know examples?
      - System trouble of Tokyo stock market, ATM system trouble, ...

Copyright© Natsuko NODA, 2014-2024

社会的な影響

- ソフトウェアは私たちの日常に大きな影響を及ぼしている...
  - 良い意味でも、悪い意味でも
  - ソフトウェアは現代社会にとって欠くことができない存在
  - 一方で、非常に深刻な問題を引き起こすことがある
    - どんな事例を知っていますか？
      - 東京証券取引所のシステムトラブル、ATMのシステムトラブル、...

Copyright© Natsuko NODA, 2014-2024

Software engineering

Copyright© Natsuko NODA, 2014-2024

ソフトウェアエ学

cf. SE-J, 1.2-1.3

Copyright© Natsuko NODA, 2014-2024

Software engineering

- (Casual understanding) The study about how to develop software that is indispensable in this modern society.
- Standard definition
  - (1) the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
    (2) The study of approaches as in (1).
    (IEEE Std 610-1990)

Copyright© Natsuko NODA, 2014-2024

ソフトウェアエ学

- 現代社会において重要な存在であるソフトウェアをどう作るのかを探求する学問
- 標準的な定義
  - (1) ソフトウェアの開発、運用、保守に対する、系統的で統制され定量化可能な方法。すなわちソフトウェアへの工学の適用。
    (2) (1) のような方法の研究。 (IEEE Std 610-1990)
  - ソフトウェアの作成と利用に関連した概念を科学的に抽出体系化し，正しいソフトウェアを計画的かつ効率的に作成・利用するための理論と，実践的技術 (岩波情報科学辞典)

Copyright© Natsuko NODA, 2014-2024

The birth of software engineering

- 1968 NATO Software Engineering Conference
  - October 7-11, 1968, Garmisch, Germany
  - Sponsor: NATO Science Committee
  - The phrase "software engineering" made its first appearance.

The phrase 'software engineering' was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines, that are traditional in the established branches of engineering.

(from the conference report)

Copyright© Natsuko NODA, 2014-2024

ソフトウェアエ学の誕生

- 1968年　NATOソフトウェアエ学会議
  - October 7-11, 1968, Garmisch, Germany
  - 主催: NATO Science Committee
  - はじめて「ソフトウェアエ学」という名称が使われる

The phrase 'software engineering' was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines, that are traditional in the established branches of engineering.

(会議レポートより)

Copyright© Natsuko NODA, 2014-2024

Background of the origin

- Software crisis in the late 1960s.
  - Many problems occurred regarding software productivity and quality.
    - (Software crisis in narrow meaning) Needed numbers of software was increasing. → It was difficult to hire enough qualified programmers.
  - In the course of expansion of software scale, we had huge issues.
    - We did not have enough productivity.
    - And then serious quality problems occurred.
  - The change of scale brought the change of quality.
    - Other processes before programming had impact on productivity and quality.
      - "Just program" did not work well.
    - Organization's power became more important than individual skills.
    - Management techniques became more important.

Copyright© Natsuko NODA, 2014-2024

誕生の背景

- ソフトウェア危機 (1960年代後半)
  - ソフトウェアの生産性・品質の問題が頻出
    - 狭義のソフトウェア危機：多くのソフトウェアが必要→質の良いプログラマを雇用することができない
  - 規模の拡大過程で大きな問題を経験
    - 生産性が追いつかない
    - 深刻な品質問題を引き起こす
  - 量の変化が質の変化をもたらす
  - 上流工程の生産性・品質へのインパクト
  - 個人スキルから組織力へ
  - 管理技術の重要性

Copyright© Natsuko NODA, 2014-2024

Brooks's law

- "Adding people to a late project only makes it more delayed."
  - Number of people and development time are not interchangeable.
- Brooks
  - Father of IBM system/360
  - "The Mythical Man Month"
    - Brooks, F.P.: Mythical Man-Month: Essays on Software Engineering, 1975.

Copyright© Natsuko NODA, 2014-2024

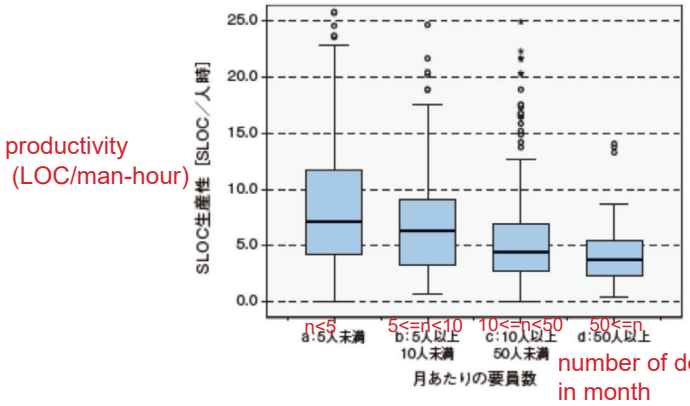
ブルックスの法則

- 遅れている開発プロジェクトでは、開発要員の増員はかえって遅れをひどくする
  - 人数と時間には互換性があるという誤った認識
- ブルックス
  - IBM system/360 の父
  - 「人月の神話」
    - Brooks, F.P.: Mythical Man-Month: Essays on Software Engineering, 1975.
      邦訳：「人月の神話-狼人間を撃つ銀の弾丸はない」滝沢他訳

Copyright© Natsuko NODA, 2014-2024

Developers/month and productivity

- The higher the number of developers per month, the lower the productivity.

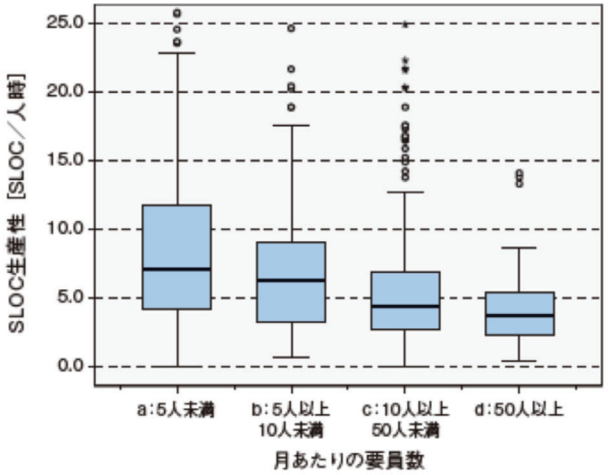


Source：「ソフトウェア開発データ白書2018-2019」 INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN (2018)

Copyright© Natsuko NODA, 2014-2024

月あたりの要員数と生産性

- 月あたりの要員数が多いほど、生産性が低い傾向



出典：「ソフトウェア開発データ白書2018-2019」独立行政法人情報処理推進機構社会基盤センター(2018)

Copyright© Natsuko NODA, 2014-2024

How to develop software

- Lifecycle of software
  - Requirements analysis： to understand problems, and acquire and order users' requirement
  - Specification： to decide software to be built
  - Design： to decide the structure of software
  - Coding： to write programs
  - Test： to check that the software follows the design, the specification, and the requirements
  - Operation & Maintenance： to use the software

※Classifications and names are various.

Copyright© Natsuko NODA, 2014-2024

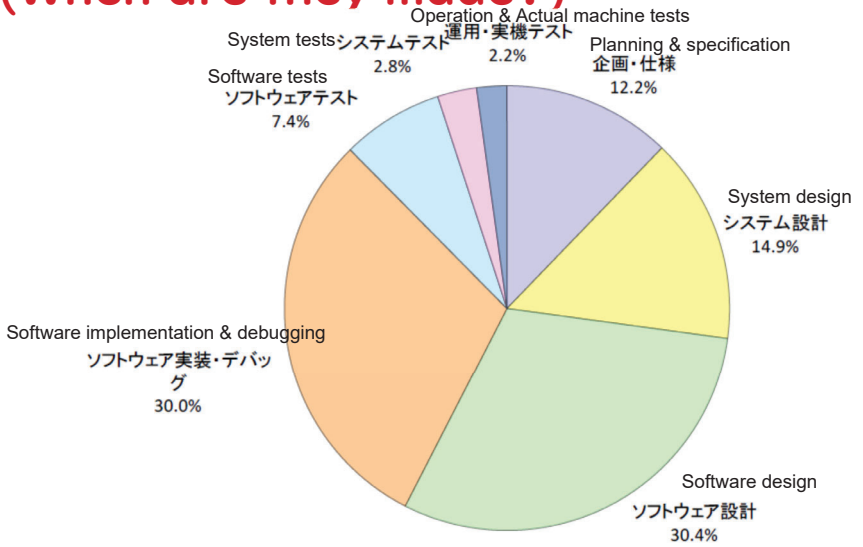
ソフトウェアの開発の仕方

- ソフトウェアのライフサイクル
  - 要求分析： 問題を理解し、要求事項を整理する
  - 仕様： どんなソフトウェアを作ればよいか決める
  - 設計： ソフトウェアの方式や構造を決める
  - コーディング： プログラムを書く
  - テスト： 設計、仕様、要求事項どおりかどうか確認する
  - 運用・保守： 実際に利用する

※いろいろな名称や分類がある

Copyright© Natsuko NODA, 2014-2024

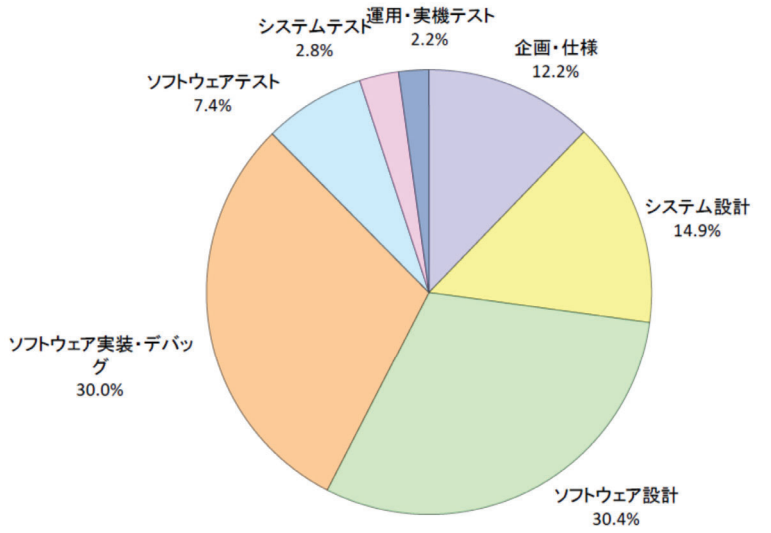
Where are causes of errors? (When are they made?)



出典：2012年度「ソフトウェア産業の実態把握に関する調査」調査報告書
独立行政法人情報処理推進機構(2013)
Source： INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN (2013)

Copyright© Natsuko NODA, 2014-2024

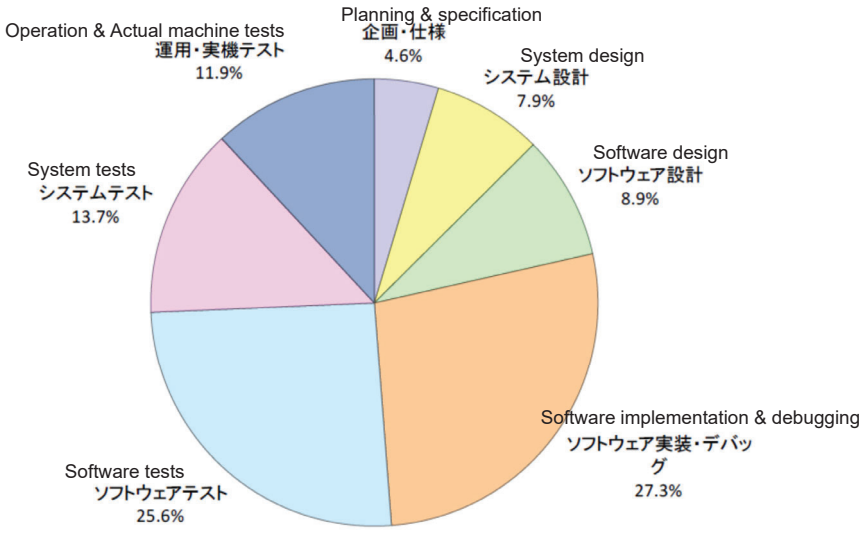
不具合の原因工程



出典：2012年度「ソフトウェア産業の実態把握に関する調査」調査報告書
独立行政法人情報処理推進機構(2013)

Copyright© Natsuko NODA, 2014-2024

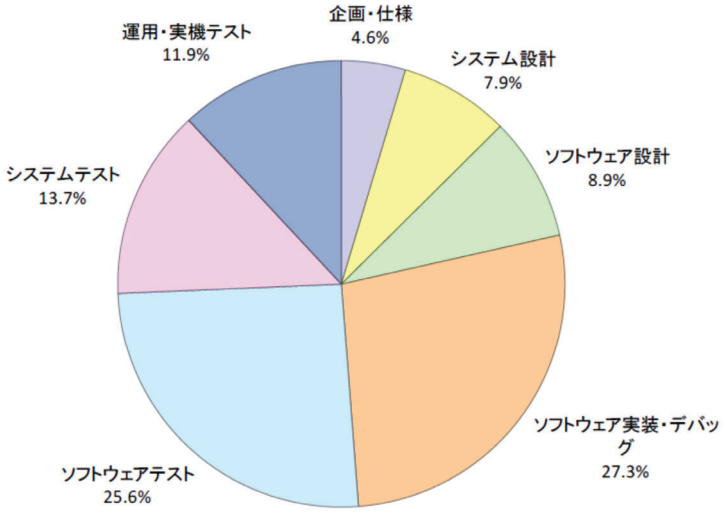
When are causes of errors corrected?



出典：2012年度「ソフトウェア産業の実態把握に関する調査」調査報告書
独立行政法人情報処理推進機構(2013)　Source： INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN (2013)

Copyright© Natsuko NODA, 2014-2024

不具合の発見工程



出典：2012年度「ソフトウェア産業の実態把握に関する調査」調査報告書
独立行政法人情報処理推進機構(2013)

Copyright© Natsuko NODA, 2014-2024

Software design is the key!

- Software design is getting more and more important.
  - Because the size and complexity of software is increasing.
  - We cannot make such complicated software without a design process.
    - Just jump to implementation (programming), can easily lead to many problems.

Copyright© Natsuko NODA, 2014-2024

ソフトウェア設計がカギ！

- ソフトウェア設計はますます重要になっている.
  - なぜなら、規模と複雑性が増している
  - 設計なくしてこのような複雑なソフトウェアを作ることはできない
    - いきなり実装(プログラミング)を始めれば、簡単に多くの問題を起こし得る

Copyright© Natsuko NODA, 2014-2024

Reference books

- No specific textbook
- Reference books
  - Software Engineering (10th edition)
    - by Ian Sommerville. Pearson
  - UML Distilled: A Brief Guide to the Standard Object Modeling Language
    - by Martin Fowler. Addison-Wesley
  - Mythical Man-Month
    - by Frederick P. Brooks. Addison-Wesley
  - Showstopper! the Breakneck Race to Create Windows NT and the Next Generation at Microsoft
    - by G. Pascal Zachary. E-Rights/E-Reads Ltd

Copyright© Natsuko NODA, 2014-2024

参考文献

- 教科書はない
- 参考書
  - Software Engineering (10th edition) 和訳なし
  - UML Distilled: A Brief Guide to the Standard Object Modeling Language
    和訳：UMLモデリングのエッセンス 第3版 (翔泳社)
  - ソフトウェア工学
    - 岸知二、野田夏子著、近代科学社
      - 日本語ページに該当の部分を“cf. SE-J, pp. xx-yy”で示す
  - Mythical Man-Month
    和訳：人月の神話 (ピアソン・エデュケーション)
    - ソフトウェア工学の古典的名著である、エッセイ・論文集。大規模ソフトウェア開発における様々な問題点を分析。時代背景が古いが本質を捉えている。
  - Showstopper! …
    和訳：闘うプログラマー ビル・ゲイツの野望を担った男達 (日経BP出版センター)
    - マイクロソフトのOS”WindowsNT”の開発物語。ソフトウェア開発とはどんなものであるのか、その一端を知ることができる。
  - ソフトウェアアーキテクチャ
    - 岸 知二、野田 夏子、深澤 良彰 著 共立出版
      - ソフトウェア工学の一分野であるソフトウェアアーキテクチャについての教科書。関連するソフトウェア工学の他の話題にも言及、背景理解に役立つと思われる。

Copyright© Natsuko NODA, 2014-2024

After class research

- Search an example of actual system troubles that was caused by its software.
  - What was that?
  - Why did it happen? What was the detail of the cause?

Copyright© Natsuko NODA, 2014-2024

授業後発展学習

- ソフトウェアが原因で起こった実際のシステムトラブルの例を調べてみよう
  - どんなトラブルか？
  - なぜ起こったのか？原因の詳細は？

Copyright© Natsuko NODA, 2014-2024