An Introduction to Software Engineering 2

Asst. Prof. Reginald Neil C. Recario
rcrecario@up.edu.ph
rncrecario@gmail.com
Institute of Computer Science
University of the Philippines Los Baños

Software Engineering

- At the end of today's session, the student is expected to:
  - Enumerate some of the important historical highlights in the development of software engineering as a discipline and its main contributors
  - Types of application and system software
  - Identify and enumerate software stakeholders
  - The three P's in software engineering
  - Identify desirable software capabilities
  - Code of ethics in software engineering

What a software is

- First coined by **John Tukey** in 1958.
- Theoretical concept of a computer was first established by **Alan Turing** in the 1930's.
- The concept of algorithm was introduced by **Muhammad Al-Khawarezmi**, a 9th century mathematician.

History of Software Development

- An algorithm became concrete when it was programmed by **Ada Lovelace**, the first computer programmer.

History of Software Development

- **Computer program**
  - Instructions that perform certain tasks on computer hardware.
  - can be written at different levels of closeness to the hardware (Low to High Level PL).

History of Software Development

- **Documentation**
  - Plays a crucial role in the success of software.
  - of interest to the stakeholders
  - User manuals, installation procedures, and operating manuals are written mainly for the software users.

The Software Crisis (1967)

- Characterized by the inability of existing techniques, tools and processes to deal with the increasing complexity of the needed software.

The Software Crisis (1967)

- **Main reasons:**
  - Software complexity
  - Changing and misunderstanding of requirements
  - Lack of tools
  - Lack of skilled professionals

The Software Crisis (1967)

- Produced software - low quality, hardly maintainable, and not meeting the stakeholder’s requirements.
- Software projects were most of the time running over budget and overtime, and many never delivered a functioning product.

The Crisis persists….

- There are many concerns about the quality and reliability of the software we use. Existing software is plagued with millions of defects.

The Crisis persists….

- Some of these defects are known and have already been detected, others are yet to be uncovered. These defects have caused many disasters leading to financial losses, physical harm to humans and life threatening situations..

The Crisis persists….

- It was also reported that most software practitioners do not hold degrees in software engineering.
- Currently, most people working as software engineers hold either a degree in computer science or computer engineering.

QUIZ!!!

- Quiz!
- On a clean ¼ sheet of paper, name one defect of our web portal for CMSC 128.

Assignment!!!

- **Assignment** (for next meeting)
  - Name one website /web application that has a defect
  - Identify the defect
  - Goal: No student should have the same website to be named

Assignment!!! (Part 1)

- **Assignment** (for next meeting)
  - Create a javascript file named <**lastname_firstname**>**.js**. Example: recario_reginald.js
  - Write a function named **printIAmGroot** with no argument that prints on the console the sentence “I am Groot”.

Assignment!!! (Part 2)

- **Assignment** (for next meeting)
  - Create a function that accepts a string input (e.g. “one hundred”) and returns the numerical counterpart.
  - Create a function that accepts a number and returns the number in words.

Software Engineering

- Software engineering is a term that was coined during the NATO Software Engineering conference held in Garmisch, Germany, in October 1968.
- The term was introduced by the conference chairman Friedrich Bauer.

Software Engineering

- ‘the application of a disciplined approach for the development and maintenance of computer software’

Software Engineering

- ‘deals with the establishment and use of sound engineering principles to economically obtain software that is reliable and works efficiently on real machines’ (by IEEE)

Software Engineering

- ‘encompasses the use of tools, techniques and methods that are useful during the execution of the steps needed for the development of the software and its future maintenance’

Types of Software

- **System software**
  - operating systems, language compilers, assemblers, device drivers, debuggers, and networking tools and software

Types of Software

- Application software or end-user software

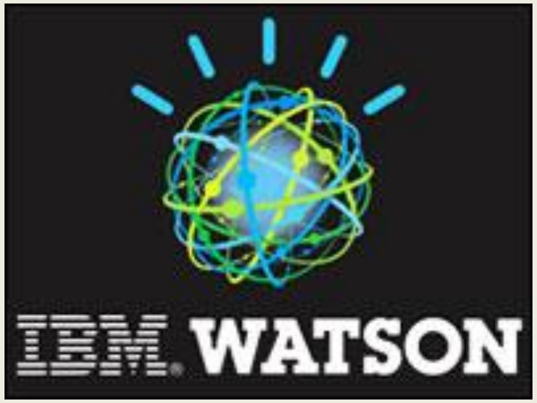
Types of application software

- **Games and entertainment software:**
  - games for handheld devices including mobile phones, PC or stand alone games, and distributed collaborative games.

Reference: http://upload.wikimedia.org/wikipedia/en/5/59/Clash_of_Clans_Logo.png

Types of application software

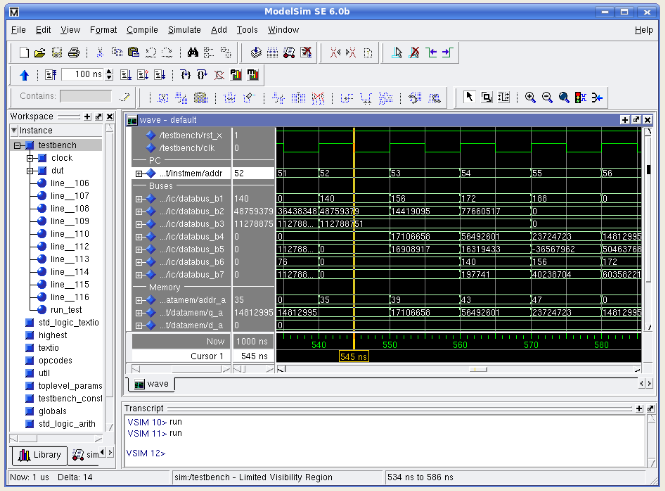
- **Intelligent software:**
  - specialized domain specific expert systems, mobile agent systems, learning systems, robot vision software, business decision and intelligence software, and mining software.



Reference: https://www.ibm.com/developerworks/mydeveloperworks/blogs/5things/resource/BLOGS_UPLOADED_IMAGES/ibm-watson.jpg

Types of application software

- **Modeling and simulation software: domain**
  - specific modeling and simulation packages for military, financial, medical, educational and training uses.



```
VSIM 10> run
VSIM 11> run
VSIM 12>
```

```
Now: 1 us   Delta: 14        sim:/testbench - Limited Visibility Region        534 ns to 586 ns
```

Reference: http://tce.cs.tut.fi/screenshots/simulating_generated_vhdl_in_modelsim.png

Types of application software

- **Real-time software:**
  - industrial plant monitoring and control systems, missile control systems, air traffic control systems, telephony software, and network security software like firewalls and intrusion detection systems.

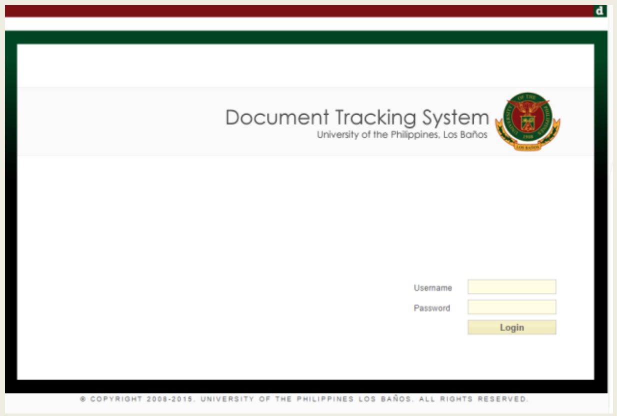
Reference: http://www.affordablecebu.com/pictures/articles/banking_3/Enroll-Credit-Card-to-Metrobank-Online.jpg

Types of application software

- **Embedded software:**
  - home appliance controllers, mobile phone software, and vehicle controllers.

Types of application software

- **Productivity (information worker) software:**
  - Tools that implement proven methods and techniques to help specific types of users doing their tasks with ease and high productivity.



© COPYRIGHT 2008-2015. UNIVERSITY OF THE PHILIPPINES LOS BAÑOS. ALL RIGHTS RESERVED.

Reference: https://docutrack.uplb.edu.ph/

Types of application software

- **Enterprise software:**
  - business workflow management software, customer relation management software, and supply chain management software.

Reference: https://www.rosehosting.com/blog/wp-content/uploads/2014/10/joomla.png

Types of application software

- **Web-based software:**
  - content management software, web publishing software, electronic commerce software, web services, web portal software, and web browsers.

Reference: https://lh5.ggpht.com/_XklY7dK6yGsYt53X15RIp7-tbdjWMwQwl_iUAcW-uSq_8zAprmywdn5DQfBHvxZurs1=w300

Types of application software

- **Educational software:**
  - school and university management software, online and distance learning software, training management software, and educational software for children.

Reference: https://lh6.ggpht.com/si8wcV053QI5Th_HMffHxTOH9qTB-tEot7FfQSgDOwOXT_Vk2gc2fyCyonfigWzRN88=w300

Types of application software

- **Multimedia software:**
  - video, image and sound editing and management software, 3D and animation software, and virtual reality software.

Reference: http://www.google.com.ph/imgres?imgurl=https://lh3.googleusercontent.com/-RIEDPq1W3VY/AAAAAAAAAAI/AAAAAAAAABI/CDt9xKQ4EXg/photo.jpg&imgrefurl=http://www.youtube.com/watch?v%3DYLKg-Ez5lpI&h=512&w=512&tbnid=rwuvER2b1ErIyM:&zoom=1&docid=5pYeJRQJ1AM5uM&ei=YS7vVLK8DcXMmAXy5IHYCQ&tbm=isch&ved=0CEIQMygTMBM

Types of application software

- **Domain-specific software:**
  - banking, finance and stocks, accounting, medical, airline reservation, hospital management, and human resource management software.

CEBU PACIFIC

Reference: http://t2.gstatic.com/images?q=tbn:ANd9GcTNuHLDYm04tdaace-AUlVmk5gHLL4hatEBgHDOA0_Pmtom1xov0A

3 Generic stages in software development and maintenance

- Structured, disciplined approach
- Aiming at enhancing quality and dealing with complexity

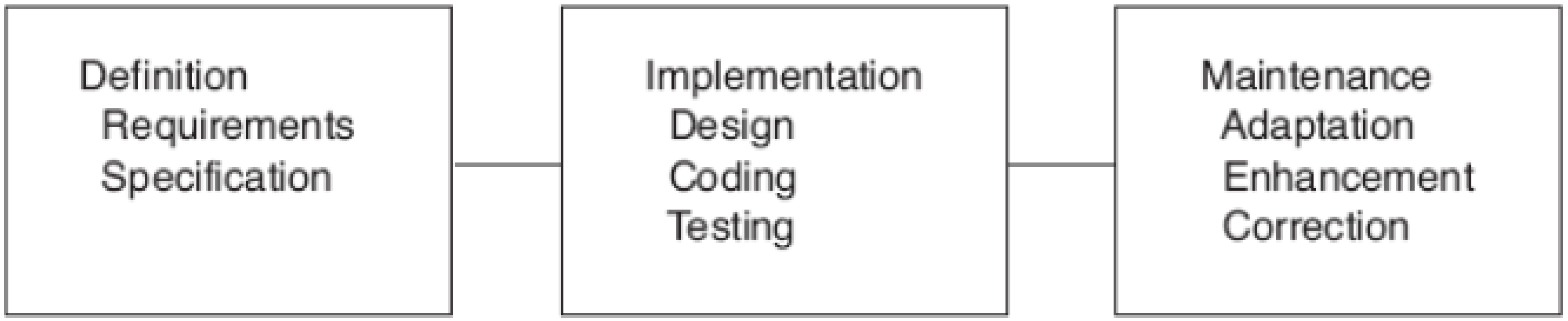


Figure 1.2 The three generic stages for software development and maintenance

Software errors

- Discovered and others are yet to be uncovered
- 25% are definition errors (requirements &
- specification)
- 25% design errors, 10% coding errors
- It costs more to fix a definition error in the maintenance phase – better discover them early!

The software triad

- Structured, disciplined approach
- Aiming at enhancing quality and dealing with complexity

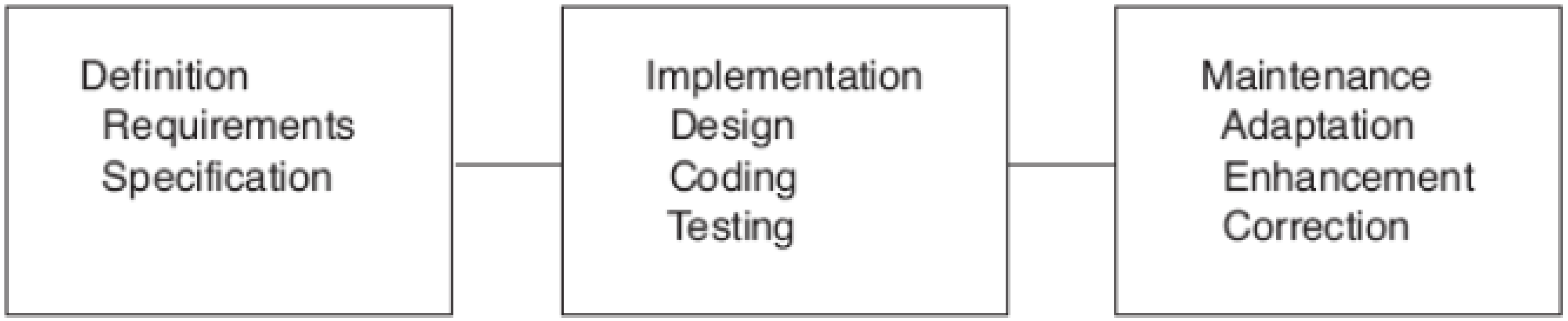


Figure 1.2 The three generic stages for software development and maintenance

The software triad

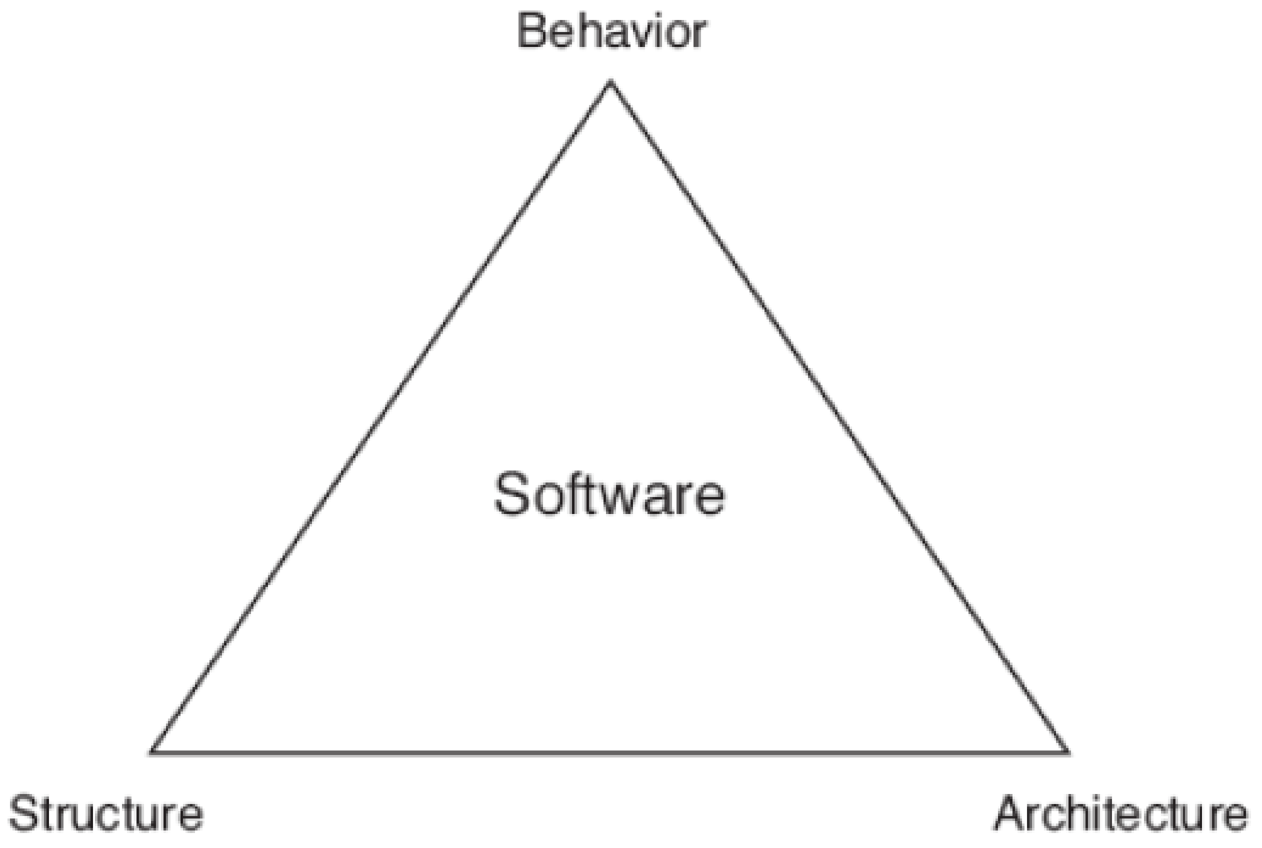


Figure 1.3 The software triad

The Stakeholders

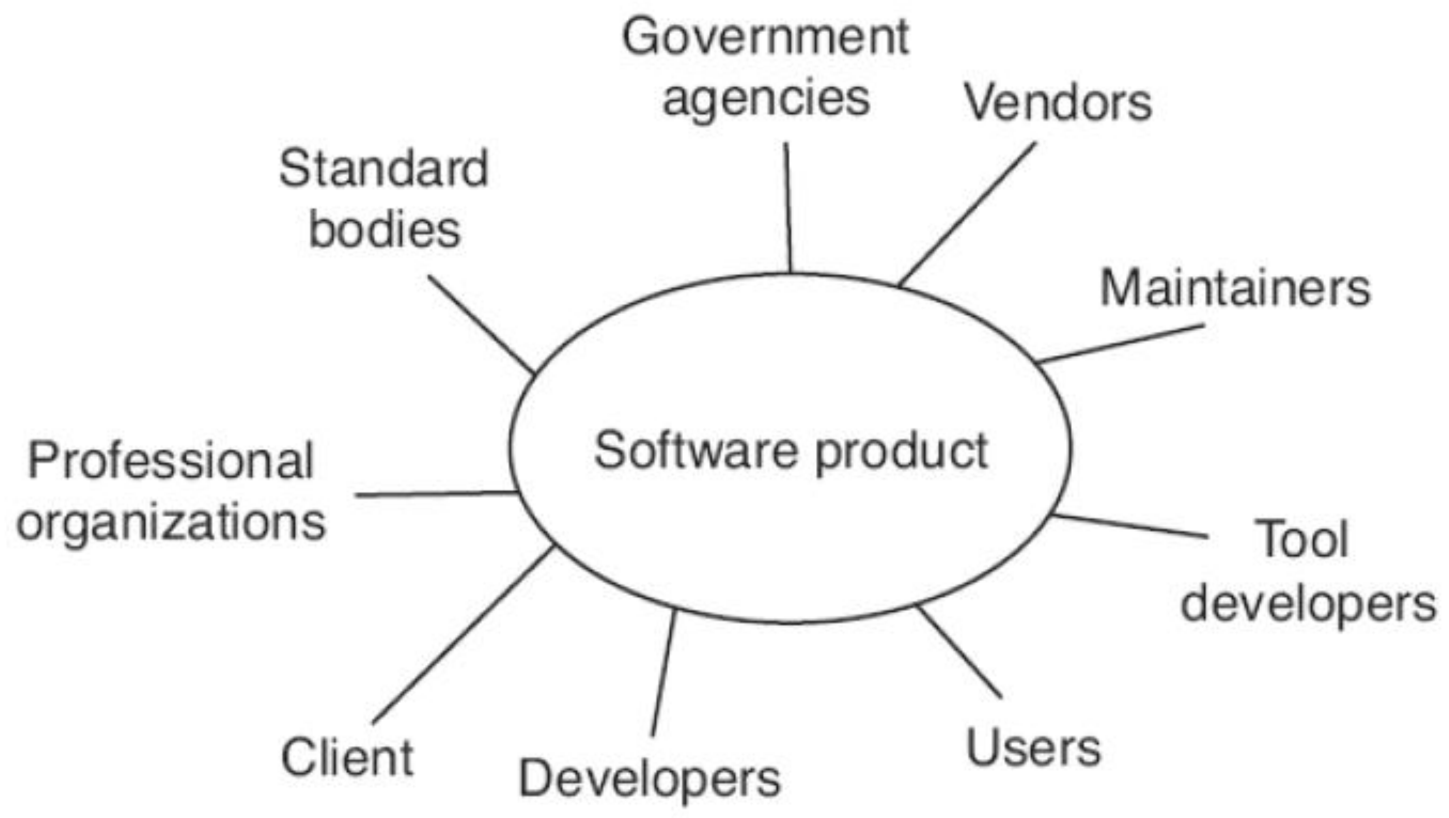


Figure 1.4 Stakeholders in a software product

The Stakeholders

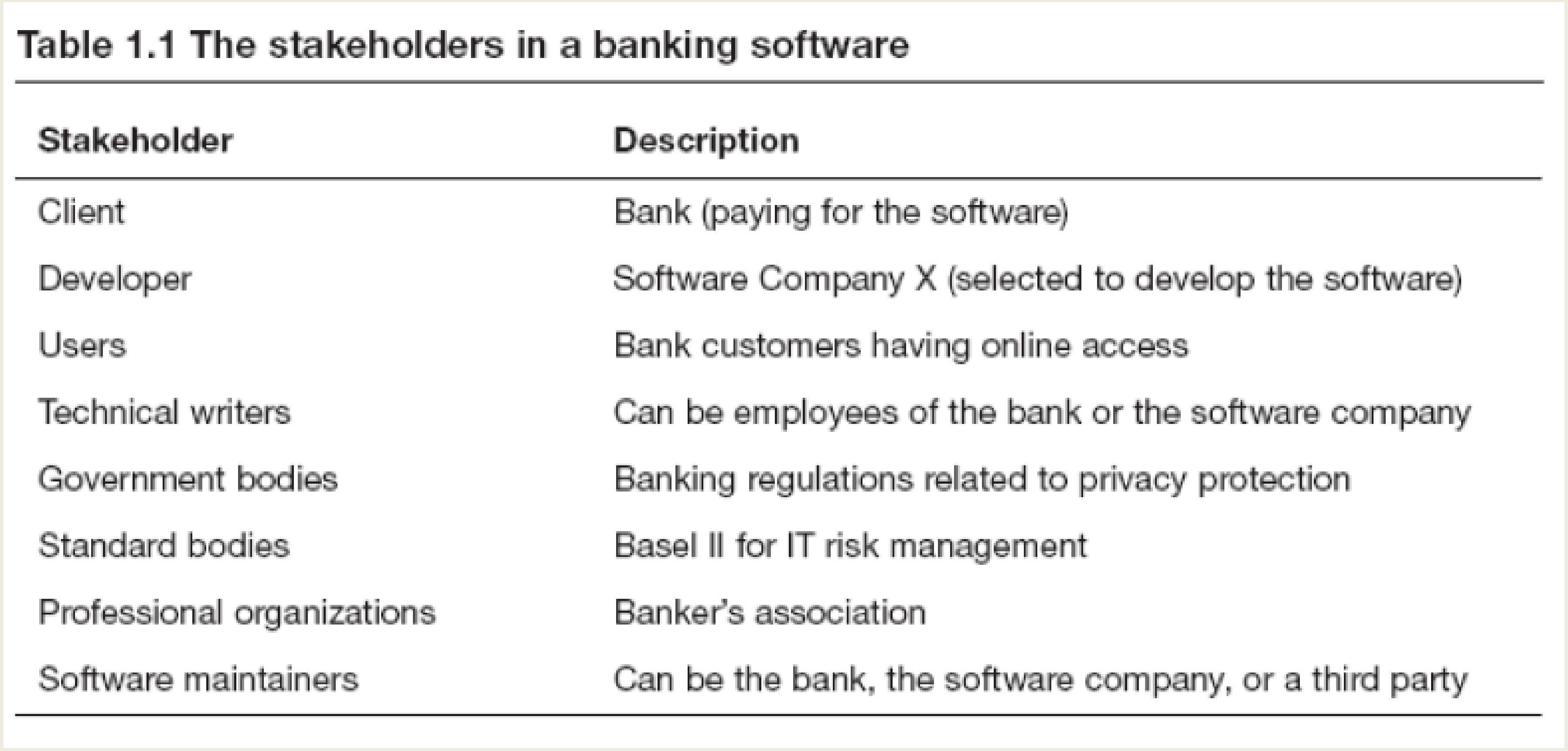


Table 1.1 The stakeholders in a banking software

Stakeholder	Description
Client	Bank (paying for the software)
Developer	Software Company X (selected to develop the software)
Users	Bank customers having online access
Technical writers	Can be employees of the bank or the software company
Government bodies	Banking regulations related to privacy protection
Standard bodies	Basel II for IT risk management
Professional organizations	Banker's association
Software maintainers	Can be the bank, the software company, or a third party

Code of ethics and professional practice for software engineers

Public:

- Software engineers shall act consistently with the public interest.
- Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.

Code of ethics and professional practice for software engineers

Product:

- Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

Code of ethics and professional practice for software engineers

Profession:

- Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

Code of ethics and professional practice for software engineers

- **Peers and self:**
- Software engineers shall maintain integrity and independence in their professional judgment.
- Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

Code of ethics and professional practice for software engineers

- **Peers and self:**
- Software engineers shall be fair to and supportive of their colleagues.
- Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Three Cornerstones of a Software Product

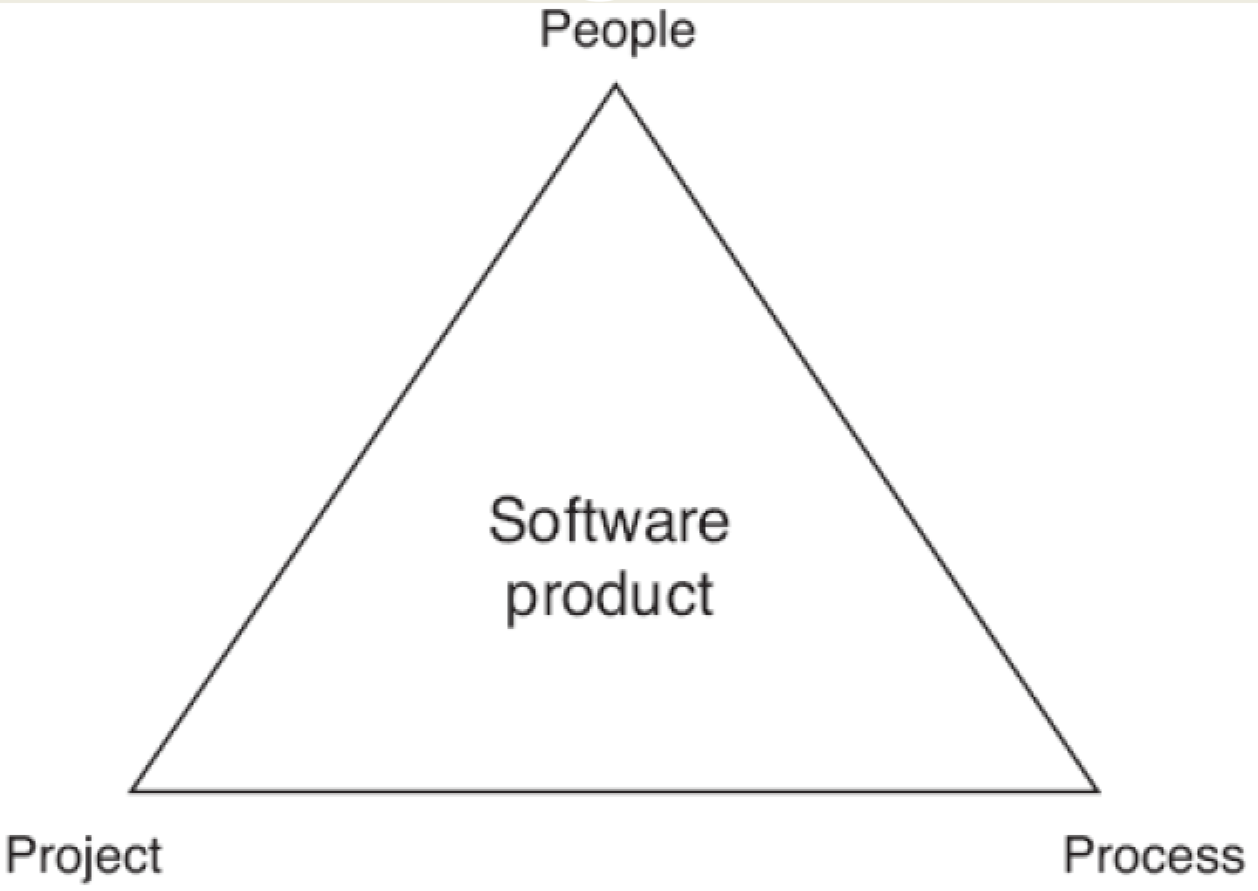


Figure 1.6 The three cornerstones of a software product

Desirable software abilities – user-centered

- **Availability** is the degree to which the software system is available when its services are required. It can be quantified as the ratio of the actual availability of the software services to the expected availability during the same period.

Desirable software abilities – user-centered

- **Correctness** is the degree to which the software meets its requirements specifications. Correctness is affected positively by the completeness, consistency, and traceability of the software. Accuracy is a qualitative assessment of correctness.

Desirable software abilities – user-centered

- **Efficiency** is the degree to which the software system performs its functions using the least amount of computational and memory resources.

Desirable software abilities – user-centered

- **Integrity** is the degree to which the software system prevents unauthorized access to information or programs. Both integrity and availability contribute to the security of the software system.

Desirable software abilities – user-centered

- **Reliability** is the ability of a software system to perform its function under stated conditions and for a specified period of time. Reliability can be quantified using the mean time between failures. Reliability is positively affected by error recoverability and fault tolerance, modularity and simplicity, and other quality factors.

Desirable software abilities – user-centered

- **Scalability** is the ability of the software system to handle a growing number of user requests up to a certain specified limit.

Desirable software abilities – user-centered

- **Usability** is the degree of ease with which the software system can be used. Usability is positively affected by learnability, readability, understandability, and other quality factors.

Desirable software abilities – developer-centered

- **Flexibility** is the measure of how easily the software system can be extended or expanded. Flexibility is positively affected by the simplicity, generality, and modularity of the software, and other quality factors.

Desirable software abilities – developer-centered

- **Interoperability** is ability of the software system to exchange information with other external systems.

Desirable software abilities – developer-centered

- **Maintainability** is the measure of how easily the software system can be modified to fix errors, to improve its functions or to adapt it to different environments. Maintainability is positively affected by adaptability, simplicity, modularity and traceability, and other quality factors.

Desirable software abilities – developer-centered

- **Portability** is the ease with which the software system can be moved to a different software or hardware environment or platform. Portability is positively affected by generality and modularity, and other quality factors.

Desirable software abilities – developer-centered

- **Reusability** is the degree of ease with which a software component can be reused in the same software system or used to build new systems. Reusability is positively affected by generality and modularity, and other quality factors.

Desirable software abilities – developer-centered

- **Testability** is the measure of how easily test cases can be selected and executed to test the software. Testability is positively affected by modularity, simplicity and traceability, and other quality factors.

Pioneers in software and software engineering

- **Alan Kay** is known for his pioneering work on window based graphical user interfaces. He is also known for coining the term object-oriented programming in 1966.

Pioneers in software and software engineering

- **Ali Mili** contributed to the formalization of software fault tolerance, now a major concern for developing secure software systems.

Pioneers in software and software engineering

- **Barry Boehm** contributed to the area of software engineering economics and software metrics. He has also introduced the spiral model for software development.

Pioneers in software and software engineering

- **Bill Joy** contributed to the development of the Unix operating system and the Java programming language.

Pioneers in software and software engineering

- **Brian Kernighan and Dennis Ritchie** were instrumental in developing the Unix operating system and the C programming language.

Pioneers in software and software engineering

- **C.A.R. Hoare** introduced the concepts of assertions and program proof of correctness, designed and analyzed well-known algorithms, and developed communicating sequential processes, a formal language for the specification of concurrent processes.

Pioneers in software and software engineering

- **David Parnas** introduced the concepts of information hiding, software interfaces and software modularity. He has also contributed to software engineering education and the ethical responsibilities of a software engineer.

Pioneers in software and software engineering

- **Donald Knuth** is known for the design of many well-known computer algorithms and the use of rigorous mathematical techniques for the formal analysis of the complexity of algorithms.

Pioneers in software and software engineering

- **Edsger Dijkstra** introduced the concept of structured programming and carried out in-depth studies of the problems of concurrency and synchronization needed in complex distributed systems.

Pioneers in software and software engineering

- **Erich Gamma** with **Richard Helm, Ralph Johnson** and **John Vlissides** introduced the concept of object oriented design patterns to facilitate software design reuse.

Pioneers in software and software engineering

- **Fred Brooks** contributed to the development of the OS/360 operating system software. He is also known for introducing the mythical man-month in software projectmanagement.

Pioneers in software and software engineering

- **Friedrich Bauer** introduced the use of stacks for the evaluation of expressions in programming language compilers. He also contributed to the development of the ALGOL programming language. Moreover, he coined the term software engineering in the NATO conference held in Germany in 1968.

Pioneers in software and software engineering

- **Grace Hopper** contributed extensively to the first compiler and the COBOL programming language in the 1959's.

Pioneers in software and software engineering

- **Grady Booch** is known for his method for object oriented analysis and design and his codevelopment of the unified modeling language (UML).

Pioneers in software and software engineering

- **John Backus** is well known for the invention of FORTRAN, compiler optimization, and the Backus-Naur Form for the formal description of programming language syntax.

Pioneers in software and software engineering

- **Michael Fagan** contributed extensively to the area of software inspection.

Pioneers in software and software engineering

- **Niklaus Wirth** introduced the Pascal programming language and other languages, and contributed to the idea of decomposition and stepwise refinement.

Pioneers in software and software engineering

- **Ole-Johan Dahl and Kristen Nygaard** introduced the Simula language which was the first object-oriented programming language.

Pioneers in software and software engineering

- **Peter Naur** is known for his contributions to the creation of the ALGOL programming language and the introduction of formal syntax description language.

Pioneers in software and software engineering

- **Tom DeMarco and Edward Yourdon** introduced the structured analysis and design approach for specifying and designing software systems.

Pioneers in software and software engineering

- **Watts Humphrey** is known for his work on the capability maturity model developed by the software engineering institute and the personal software process.

Pioneers in software and software engineering

- **Winston Royce** contributed the well-known model for the management of large software systems which was later named the Waterfall model.

Software Building and Development

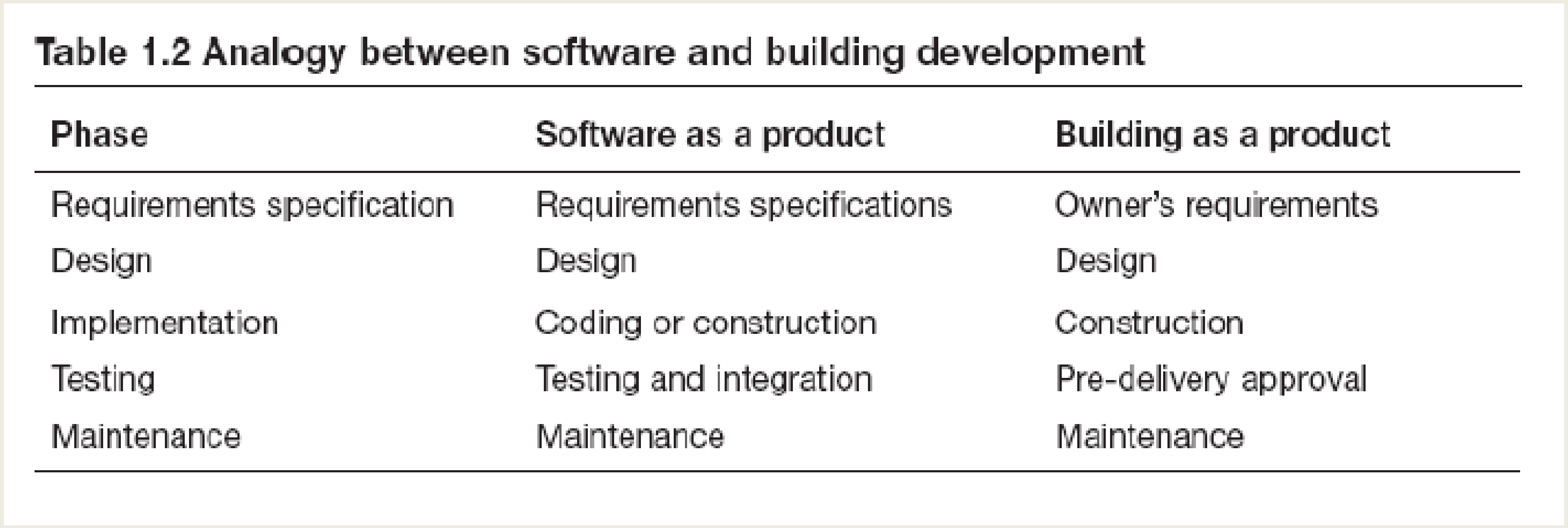


Table 1.2 Analogy between software and building development

Phase	Software as a product	Building as a product
Requirements specification	Requirements specifications	Owner's requirements
Design	Design	Design
Implementation	Coding or construction	Construction
Testing	Testing and integration	Pre-delivery approval
Maintenance	Maintenance	Maintenance

References

- **Pressman, Roger S**. Software Engineering: A Practioner'sApproach (7th ed).
- **Saleh, Kassem A**. Software Engineering. J Ross Publishing. ISBN-13: 978-981-3419-27-0. 2009 Philippine reprint 2010, Cengage Learning Asia Pte Ltd
- **Sommerville, Ian**. Software Engineering (9th ed).