# BLG 411E SOFTWARE ENGINEERING

## Week 1

## Introduction – Software Projects

Prof. Dr. Tolga OVATMAN        Assoc. Prof. Dr. Ayşe TOSUN
Istanbul Technical University
Faculty of Computer and Informatics
2024

# Course Introduction

1.1

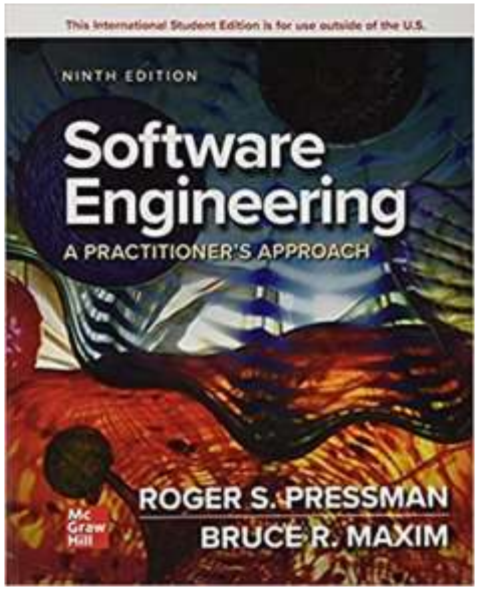
# Course - Introduction

- BLG411E – Software Engineering
  - Credits: 3-0-0
  - ECTS: 8
  - Course Web Site : [www.ninova.itu.edu.tr](www.ninova.itu.edu.tr)
    - Lecture notes, announcements, report templates and examples, tools, etc.

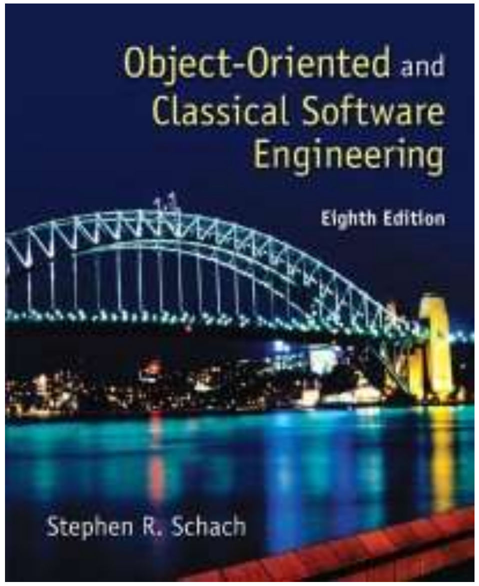
| 2024 – 2025 FALL Semester | | |
|---|---|---|
| **Instructor** | Prof. Dr. Tolga OVATMAN | Assoc. Prof. Dr. Ayşe TOSUN |
| **Lectures** | Tuesday, 13:30-16:30 | Tuesday, 13:30-16:30 |
| **Office Hours** | With appointment | With appointment |
| **Teaching Assistants** | | |

# Textbooks

Software Engineering:
A Practitioner's Approach
Roger S. Pressman,
Bruce R. Maxim
9th ed. McGraw-Hill, 2019


This International Student Edition is for use outside of the U.S.
NINTH EDITION
Software Engineering
A PRACTITIONER'S APPROACH
ROGER S. PRESSMAN
BRUCE R. MAXIM
Mc Graw Hill

Object-Oriented and Classical Software Engineering
Stephen R. Schach,
8th ed. McGraw-Hill, 2010



Object-Oriented Software Engineering: Practical Software Development Using UML and Java
T. Lethbridge, R. Laganiere,
2nd ed. McGraw-Hill, 2004

Object-Oriented Software Engineering
Practical Software Development Using UML and Java
2nd edition
Timothy C. Lethbridge
Robert Laganière
Mc Graw Hill

## Supplementary Material

| | |
|---|---|
| Other Books | • Software Engineering<br>*Ian Sommerville, 10th ed., Addison Wesley, 2017*<br>• Yazılım Mühendisliği<br>*Erhan Sarıdoğan, 1st ed., Papatya Yayıncılık, 2004* |
| Journals | • IEEE Transactions on Software Engineering<br>• IEEE Software<br>• ACM Transactions on Software Engineering and Methodology<br>*Please click here for a larger list of journals.* |
| Societies | • The IEEE Computer Society |
| Other Links | • The Software Engineering Institute |

# Outline

| Week | Date | Topic |
|---|---|---|
| 1 | 1.10.2024 | Introduction – Software Projects and Scope (Ch.1) |
| 2 | 8.10.2024 | Software Processes Models (Ch.2) + Agile |
| 3 | 15.10.2024 | Software Project Management (Ch. 9) |
| 4 | 22.10.2024 | Requirements Engineering (Ch.11) + DFD |
| **5** | **29.10.2024** | **HOLIDAY** |
| 6 | 5.11.2024 | How to start a project, configurations, test, documentation, CI |
| 7 | 12.11.2024 | Software Design (Ch. 14-17) |
| **8** | **19.11.2024** | **FALL BREAK** |
| 9 | 26.11.2024 | Software Design (Ch. 14-17) |
| 10 | 3.12.2024 | Software Architecture |
| 11 | 10.12.2024 | Software Architecture |
| 12 | 17.12.2024 | Testing I (Ch.14-15) |
| 13 | 24.12.2024 | Testing II (Ch.14-15) |
| **14** | **31.12.2024** | **HOLIDAY** |
| 15 | 7.01.2025 | Project Presentations (No Class) |

# Project

- You are free to decide on the project topic
  - Set your scope very carefully according to the designated deadlines!!!
- Minimum expected technologies are as follows:
  - Web application that has backend and frontend components, frontend may either be browser, mobile or both.
  - Service based design where APIs are offered by the backend components
  - Utilization of a structured or NoSQL database
  - Utilization of a cloud service to host the provided services (see https://github.com/cloudcommunity/Cloud-Free-Tier-Comparison)
  - Some form of authentication and authorization
  - Sustained usage of github technologies, workflows and github projects.
- Project is a TEAM work!!!
  - Project teams should be 6-8 persons
  - Each member should be designated with at least one technical and one management duty.
  - Establish a collaboration process, select your communcation tools and channels, set your communication expectations, run effective meetings (see https://cmu-313.github.io/assets/pdfs/06-teams-communication.pdf for more)

# Project Outline

| Week | Date | Recitation | Deliverables | Customer Meetings |
|---|---|---|---|---|
| 1 | 1.10.2024 | | | |
| 2 | 8.10.2024 | | Project Charter | |
| 3 | 15.10.2024 | Github Workflows | | Meeting 1 on Project Plan |
| 4 | 22.10.2024 | | Project Plan | |
| 5 | 29.10.2024 | | Peer Scoring 1 | Meeting 2 on Requirements |
| 6 | 5.11.2024 | Design Tools | Requirements Spec | |
| 7 | 12.11.2024 | | | |
| **8** | **19.11.2024** | -- | -- | -- |
| 9 | 26.11.2024 | | | Meeting 3 on Design |
| 10 | 3.12.2024 | Backend Tools | Design Spec | |
| 11 | 10.12.2024 | | Peer Scoring 2 | |
| 12 | 17.12.2024 | Testing Tools | | |
| 13 | 24.12.2024 | | | Meeting 4 on Tests |
| 14 | 31.12.2024 | | Test Spec | Meeting 5 Final Deliverables |
| 15 | 7.01.2025 | | Peer Scoring 3 | Final Demo Presentation |

# Rules

- You need to report all your meetings with the customers (TAs) and confirm the needs before moving forward. Remember that customers change their minds too often!
- During the project, everyone will evaluate his/her team members' performance and contribution to the project. This is going to be privately shared with the instructors only.
- Inform us (me or TAs) early if you have any difficulties during the project regarding the topic, technologies or with your group members!
- If a project member believes that some project members have not contributed the phase
  - Must submit a one-page summary for the phase, indicating how much he/she thinks each project member has contributed to the phase
- Plagiarism, cheating or copying other people's work/documents/source code will be dealt with extreme intolerance. Disciplinary actions will be applied immediately.

# Grading

| DELIVERABLE | GRADE | VF CRIT |
|---|---|---|
| Project Charter | - | YES |
| Project Plan | 7 | |
| Software Requirements Specification (SRS) | 10 | |
| Software Design | 10 | |
| Test Report | 8 | |
| Peer and TA Scoring | SCALE F. | |
| Use of Tools and Tooling | 5 | |
| Project Demo and Final Deliverables | 20 | YES |
| Final Exam | 40 | |

# Software History and Problems

1.2

# Nature of Software Systems

- Intangible
- Labor intensive
- Physically easy to change, but hard to modify - Complexity
- Mass production of duplicate items – Reuse
- Does not wear out, but its design deteriorates - Maintenance

# Software Engineering - History

- Engineering discipline since 1970-80s
- Hardware engineering → System engineering → Software engineering

P. Bourque et al. / The Journal of Systems and Software 62 (2002) 59–70

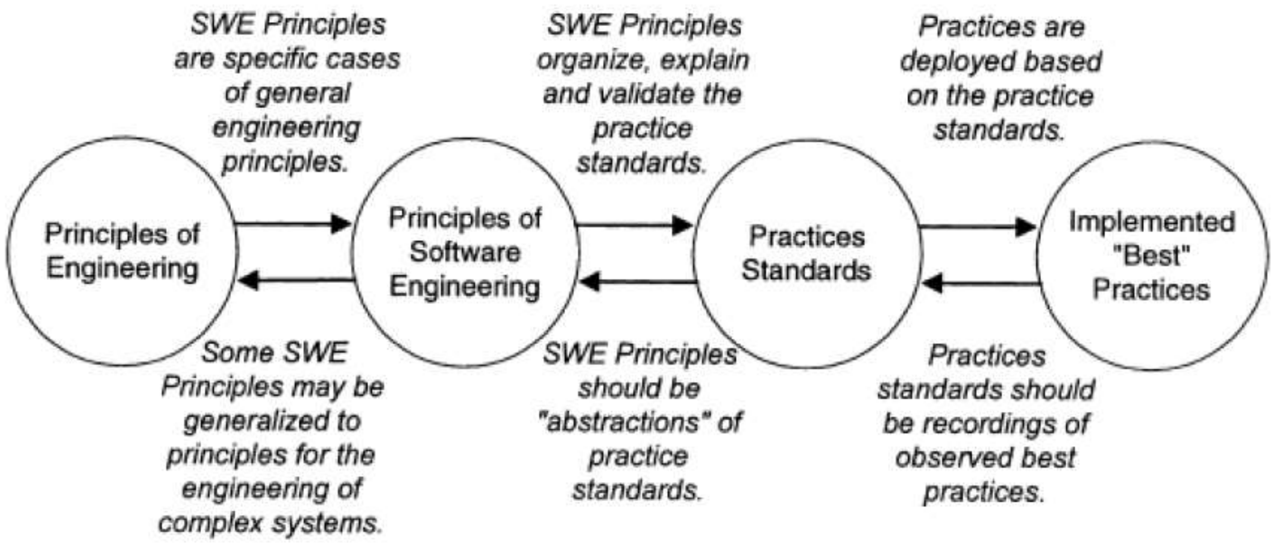
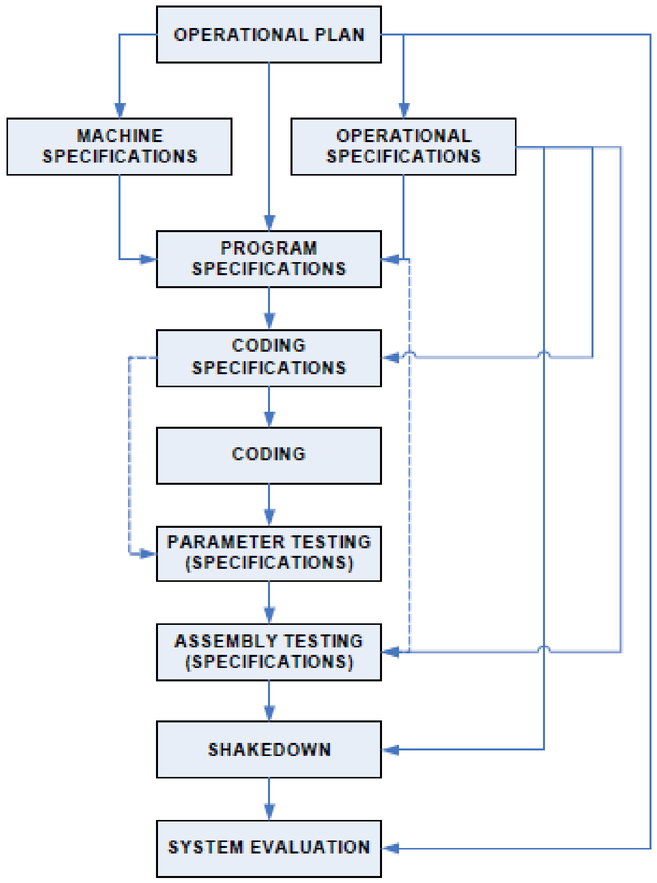

SWE Principles are specific cases of general engineering principles.
SWE Principles organize, explain and validate the practice standards.
Practices are deployed based on the practice standards.
Principles of Engineering
Principles of Software Engineering
Practices Standards
Implemented "Best" Practices
Some SWE Principles may be generalized to principles for the engineering of complex systems.
SWE Principles should be "abstractions" of practice standards.
Practices standards should be recordings of observed best practices.

Fig. 1. Relationship between principles and practice.

# 1950s

- *Software engineering is like Hardware engineering.*
- Sequential waterfall-type models have been in use for software development for a long time.
- If you put coding at the bottom, it becomes V-model.
- Newly established organizations:
  - Association for Computing Machinery
  - IEEE Computer Society



B. Boehm, A View of 20th and 21st Century Software Engineering, in Proc. ICSE 2005.
B. Boehm, Some Future Software Engineering Opportunities and Challenges, in Proc. of The Future of Software Engineering Symposium, 2010.

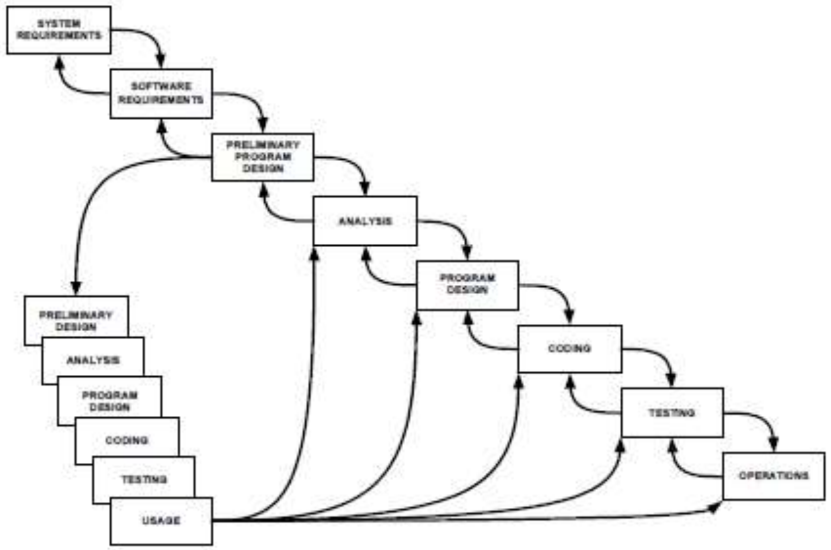
## 1960s

- *Crafting*
- Differences from hardware
  - Software was much easier to modify than hardware.
    - “code and fix” as opposed to “Critical Design Reviews”
  - Software reliability could only imperfectly be measured by hardware models.
  - It is invisible, weightless, but costs a lot.
    - Hard to tell whether it was on schedule, if more people should be added.
  - Many more states, modes, and paths to test, making its specifications much more difficult.
- Establishment of computer science and informatics departments of universities
- Software Engineering Conferences (NASA)

B. Boehm, A View of 20th and 21st Century Software Engineering, in Proc. ICSE 2005.
B. Boehm, Some Future Software Engineering Opportunities and Challenges, in Proc. of The Future of Software Engineering Symposium, 2010.

# 1970s

- *Formality and Waterfall Processes*
- Reaction to “code and fix” approach
- Royce’s waterfall model
  - Iterations
  - Prototyping activity
- Quantitative approaches
  - Coupling, cohesion, information hiding
  - Complexity metrics
  - Reliability estimation models
  - Software quality
  - Cost and effort estimation models
  - Software engineering laboratories (NASA)



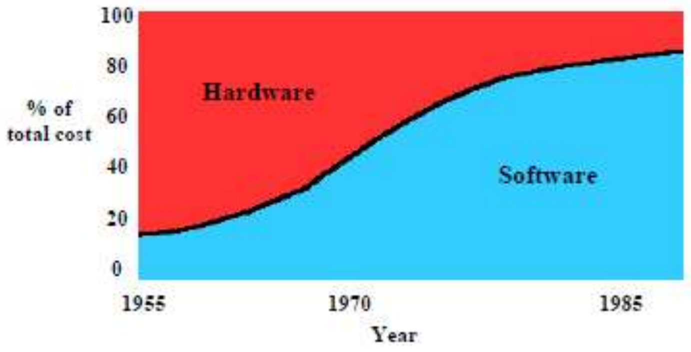
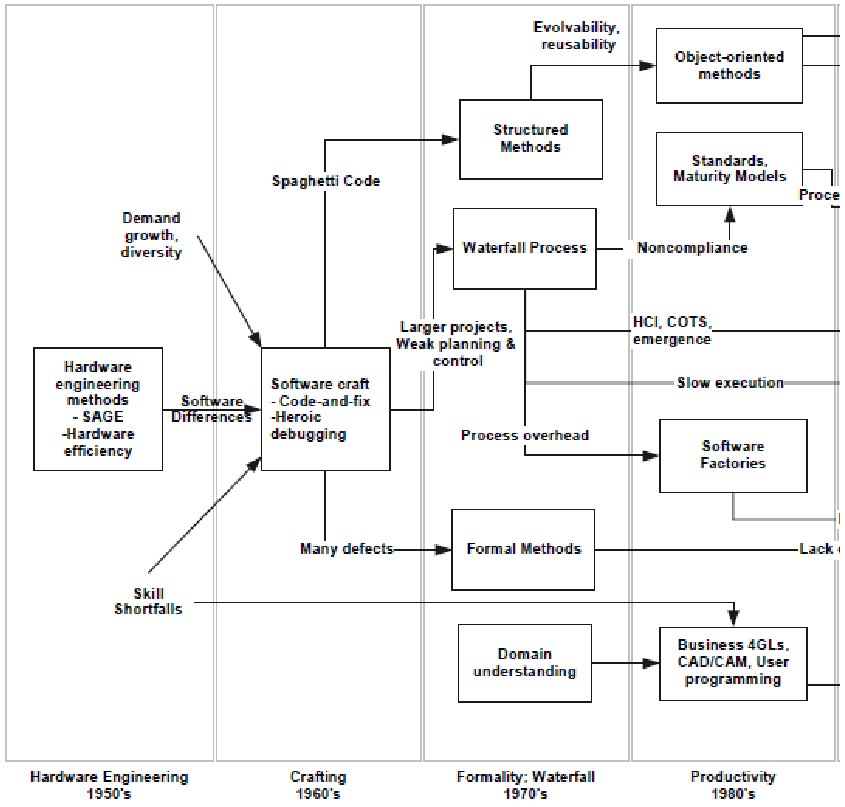


Figure 5. Large-Organization Hardware-Software Cost Trends (1973)

B. Boehm, A View of 20th and 21st Century Software Engineering, in Proc. ICSE 2005.
B. Boehm, Some Future Software Engineering Opportunities and Challenges, in Proc. of The Future of Software Engineering Symposium, 2010.

# 1980s

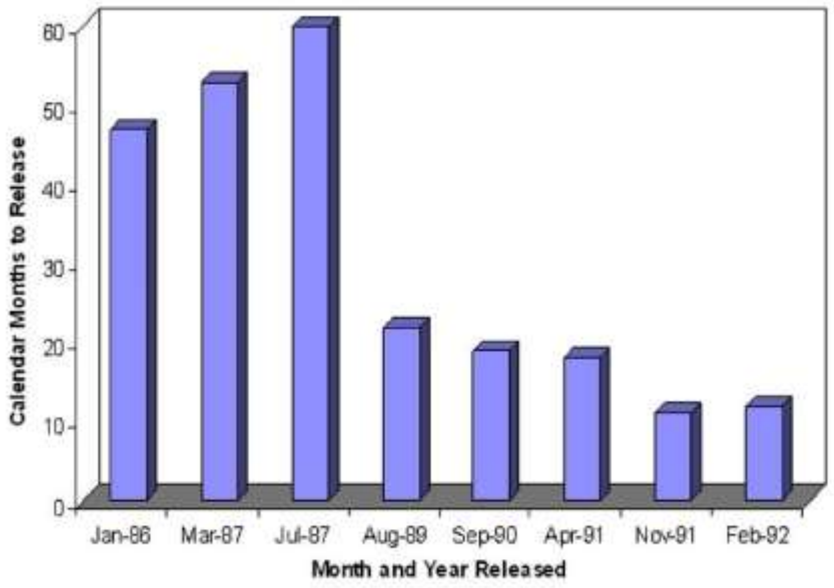
- *Productivity and Scalability*
- CMM forced by U.S. DoD
- Reinforcement of waterfall-model
- Software tools
- Software processes
- Software reuse
- C++ and Java
- No silver bullet



B. Boehm, A View of 20th and 21st Century Software Engineering, in Proc. ICSE 2005.
B. Boehm, Some Future Software Engineering Opportunities and Challenges, in Proc. of The Future of Software Engineering Symposium, 2010.

# 1990s

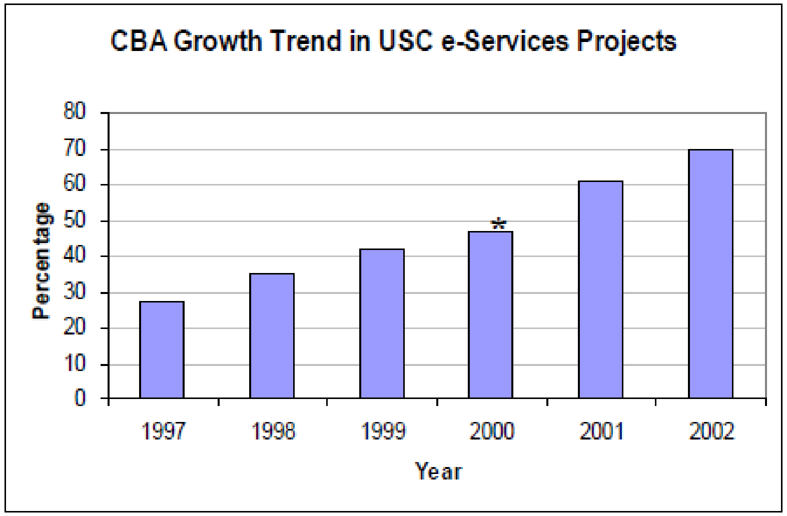
- *Concurrent vs. Sequential Processes*
- Emphasis on Time-to-Market
- Controlling Concurrency
- Open source development, General Public License, UML, Expansion of WWW
- Usability and HCI



B. Boehm, A View of 20th and 21st Century Software Engineering, in Proc. ICSE 2005.
B. Boehm, Some Future Software Engineering Opportunities and Challenges, in Proc. of The Future of Software Engineering Symposium, 2010.

# 2000s

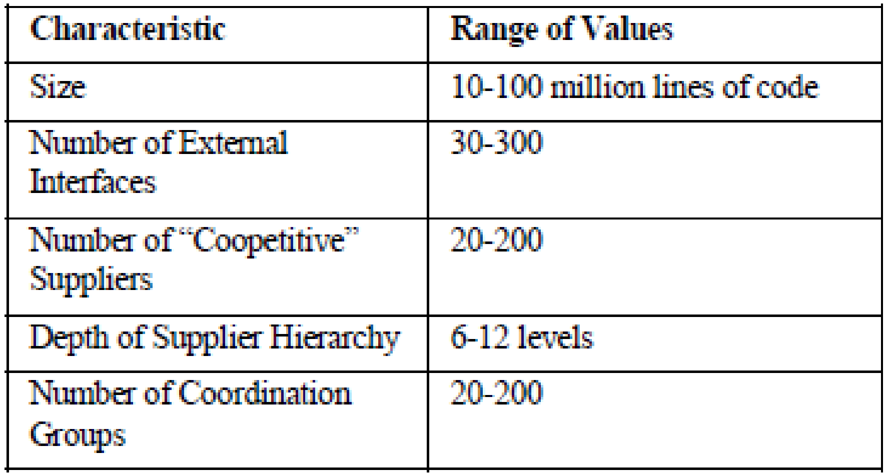
- *Agility and Value*
- Agile Methods
- Value-based Software Engineering
- Software Criticality and Dependability
- COTS, Open Source and Legacy Software
- Model-Driven Development
- Interacting Software and System Engineering



B. Boehm, A View of 20th and 21st Century Software Engineering, in Proc. ICSE 2005.
B. Boehm, Some Future Software Engineering Opportunities and Challenges, in Proc. of The Future of Software Engineering Symposium, 2010.

# 2010s

- *Globalization and Systems of Systems*
- Software-intensive SoS
- User Patterns and End Value Focus
- Rapid, Accelerating Change



| Characteristic | Range of Values |
|---|---|
| Size | 10-100 million lines of code |
| Number of External Interfaces | 30-300 |
| Number of "Coopetitive" Suppliers | 20-200 |
| Depth of Supplier Hierarchy | 6-12 levels |
| Number of Coordination Groups | 20-200 |

B. Boehm, A View of 20th and 21st Century Software Engineering, in Proc. ICSE 2005.
B. Boehm, Some Future Software Engineering Opportunities and Challenges, in Proc. of The Future of Software Engineering Symposium, 2010.

# Future Trends (2020-2050)

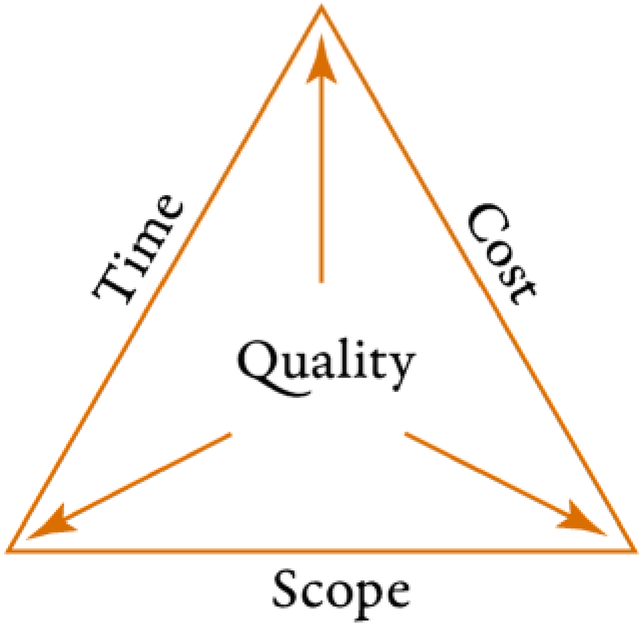
- Mega sensor-intensive smart systems
- Search and mining of ultra-large data aggregations
- Software implications of multicore chips
- Software as a service
- Social networking technologies
- Empirically evolved process technology: Languages, methods, metrics, models, tools
- Lean, value-based processes for balancing dependability and agility
- Autonomy and Bio-Computing
- Software implications of deep learning models
- Generative models for software development (e.g. CoPilot)

## What is Software Engineering?

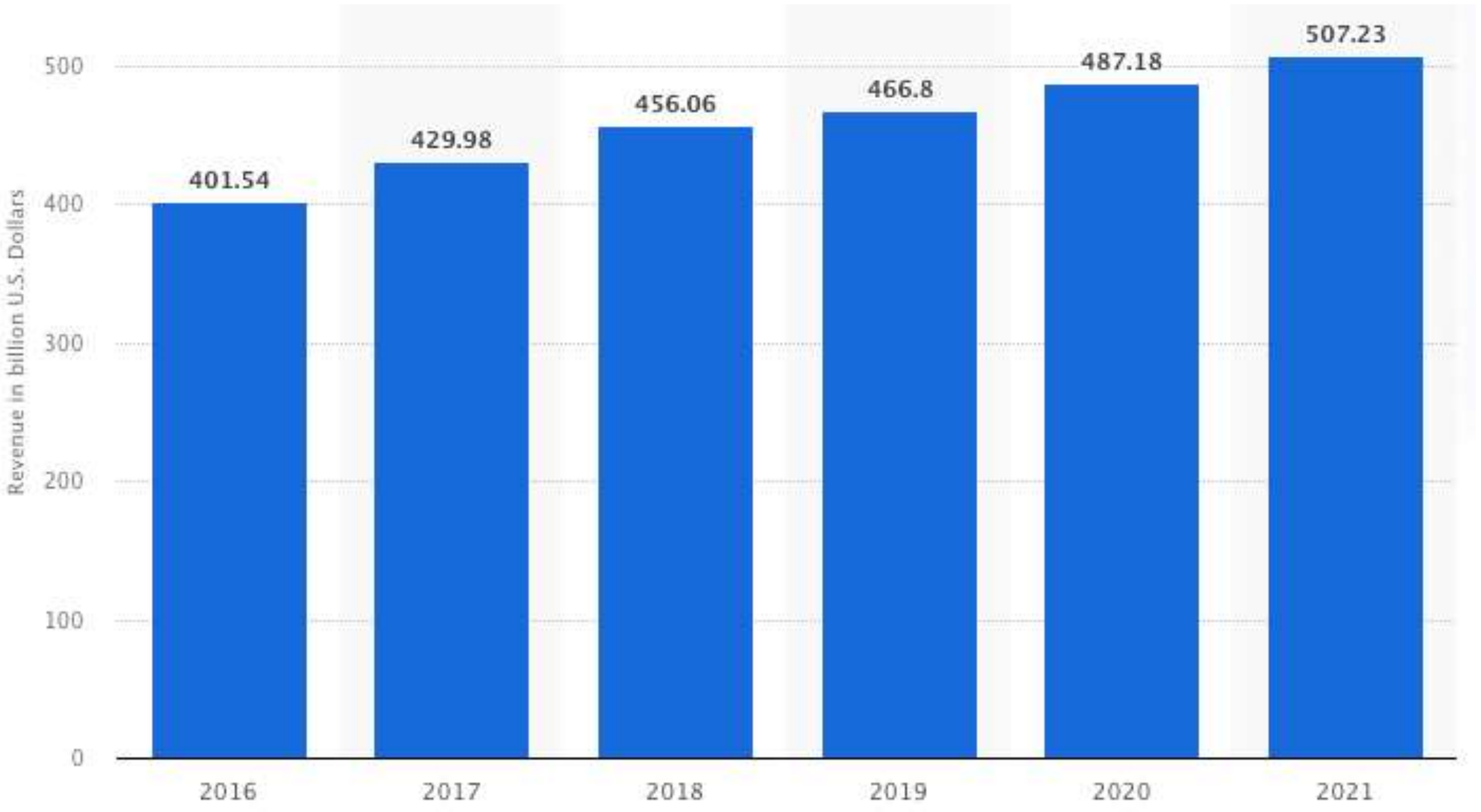
- Formal Definition
  - The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"
    [IEEE Standard, 610.12, 1990].

## What is Software Engineering?

- The study of systematic and effective processes and technologies for supporting software development and maintenance activities
  - Improve quality
  - Reduce costs
  - Deliver on-time



# The Software Market

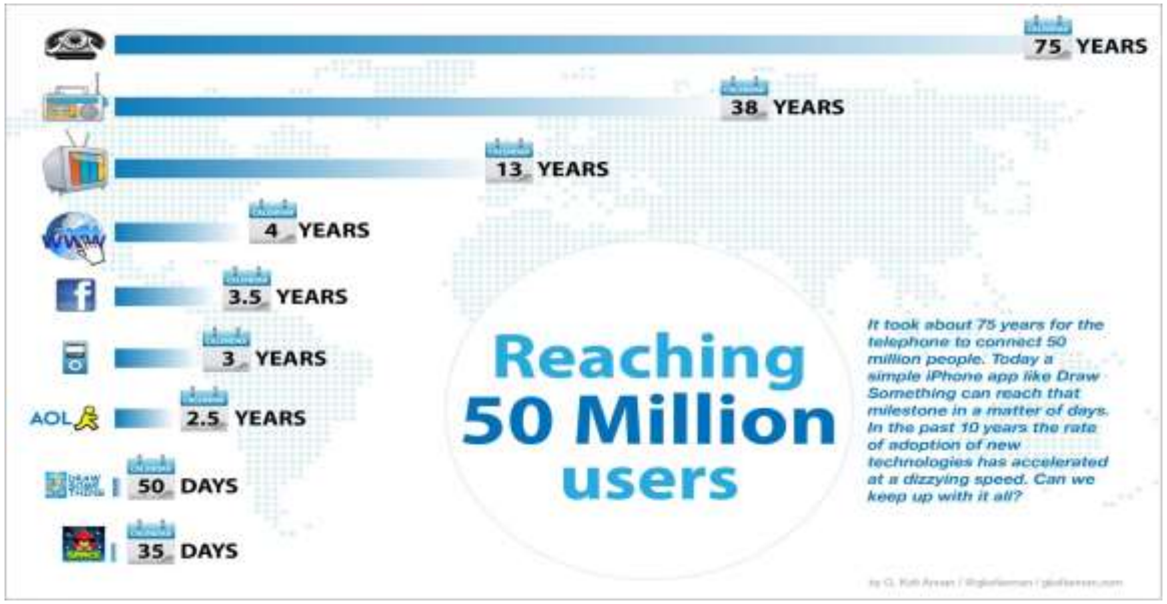


Software Market Revenue

https://www.statista.com/forecasts/963597/software-revenue-in-the-world

# What's the problem?

- Software cannot be built fast enough to keep up with technology
- Increasing need for high reliability software
- Software is difficult to maintain
- Difficult to estimate software costs and schedules
- Too many projects fail



## Software Disaster Examples - 1

### Therac-25 (1985)

- **Cost:** Three people dead, three people critically injured
- **Disaster:** Canada's Therac-25 radiation therapy machine malfunctioned and delivered lethal radiation doses to patients.
- **Cause:** Because of a subtle bug called a race condition, a technician could accidentally configure Therac-25 so the electron beam would fire in high-power mode without the proper patient shielding.

## Software Disaster Examples -2

### Patriot Missile (1991)

- **Cost:** 28 soldiers dead, 100 injured
- **Disaster:** During the first Gulf War, an American Patriot Missile system in Saudi Arabia failed to intercept an incoming Iraqi Scud missile. The missile destroyed an American Army barracks.
- **Cause:** A software rounding error incorrectly calculated the time, causing the Patriot system to ignore the incoming Scud missile.

# Software Disaster Examples - 3

## Ariane 5 Rocket (1996)

- **Cost:** $500 million
- **Disaster:** Ariane 5, Europe’s newest unmanned rocket, was intentionally destroyed seconds after launch on its first flight. Also destroyed was its cargo of four scientific satellites to study how the Earth’s magnetic field interacts with solar winds.
- **Cause:** Shutdown occurred when the guidance computer tried to convert the sideways rocket velocity from 64-bits to a 16-bit format. The number was too big, and an overflow error resulted. When the guidance system shut down, control passed to an identical redundant unit, which also failed because it was running the same algorithm.

## More recent failure examples

- London Stock Exchange's Transfer & Automated Registration of Uncertificated Stock (TAURUS) System
- CONFIRM travel reservation system
  to combine airline, car-rental & hotel information
- RACV Insurance's Workflow Management System (litigation)
- FoxMeyer's Delta III Information System
- British Sky Broadcasting's Call Center & Customer Relationship Management (CRM)(litigation)
- O2
- TSB Bank
- Security attacks: Spectre, WannaCry
- Cloudfare
- Nest

# Standish Chaos Reports

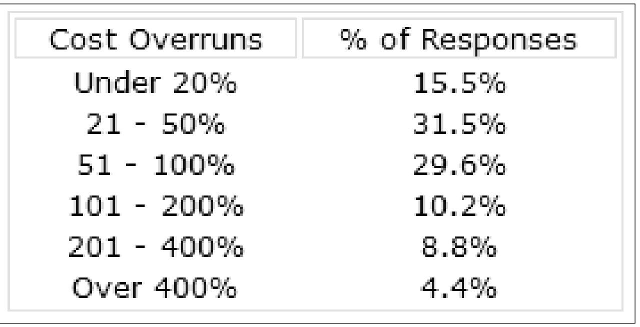
| Year | Successful (%) | Challenged (%) | Failed (%) |
|---|---|---|---|
| 1994 | 16 | 53 | 31 |
| 1996 | 27 | 33 | 40 |
| 1998 | 26 | 46 | 28 |
| 2000 | 28 | 49 | 23 |
| 2004 | 29 | 53 | 18 |
| 2006 | 35 | 46 | 19 |
| 2009 | 32 | 44 | 24 |
| 2011 | 29 | 49 | 22 |
| 2012 | 27 | 56 | 17 |
| 2013 | 31 | 50 | 19 |
| 2014 | 28 | 55 | 17 |
| 2015 | 29 | 52 | 19 |
| … | | | |
| 2020 | 31 | 50 | 19 |

https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

J. DeFranco and J. Voas, "Revisiting Software Metrology" in Computer, vol. 55, no. 06, pp. 12-14, 2022.

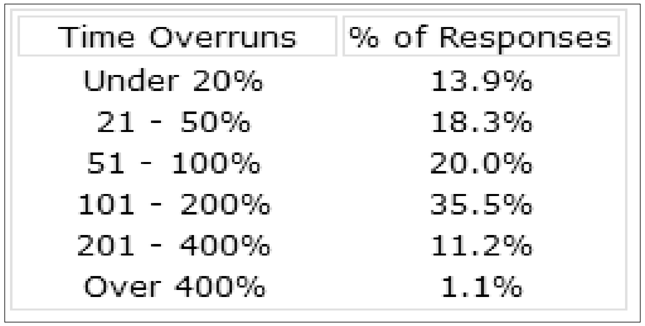
# Overrunning??

## Cost Overrun Data



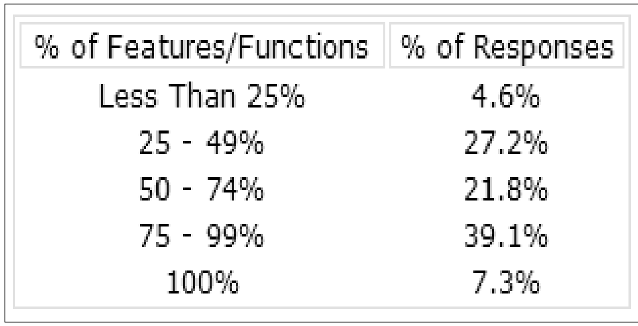
| Cost Overruns | % of Responses |
|---|---|
| Under 20% | 15.5% |
| 21 - 50% | 31.5% |
| 51 - 100% | 29.6% |
| 101 - 200% | 10.2% |
| 201 - 400% | 8.8% |
| Over 400% | 4.4% |

## Time Overrun Data



| Time Overruns | % of Responses |
|---|---|
| Under 20% | 13.9% |
| 21 - 50% | 18.3% |
| 51 - 100% | 20.0% |
| 101 - 200% | 35.5% |
| 201 - 400% | 11.2% |
| Over 400% | 1.1% |

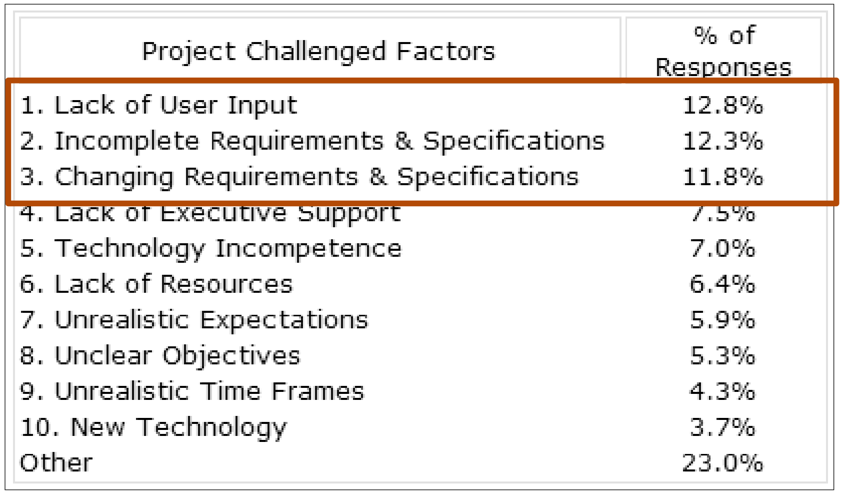
## # of Feature Dropped



| % of Features/Functions | % of Responses |
|---|---|
| Less Than 25% | 4.6% |
| 25 - 49% | 27.2% |
| 50 - 74% | 21.8% |
| 75 - 99% | 39.1% |
| 100% | 7.3% |

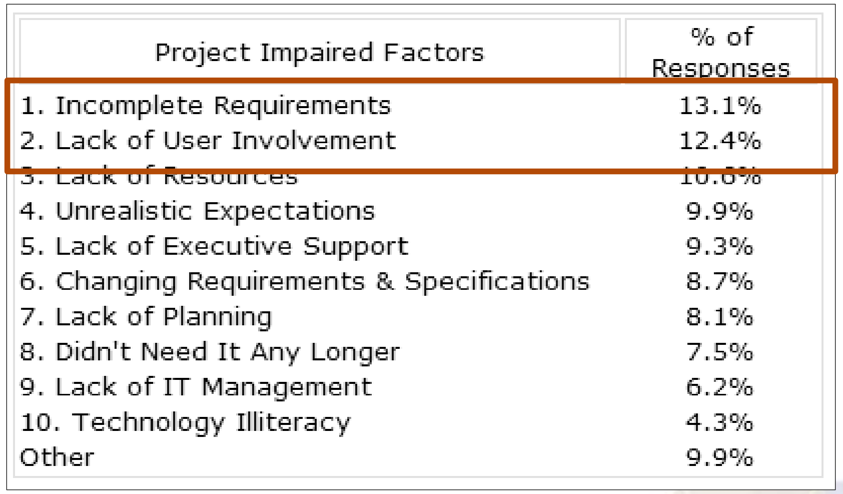
https://www.projectsmart.co.uk/white-papers/chaos-report.pdf

## Factors Making SD Difficult



| Project Challenged Factors | % of Responses |
|---|---|
| 1. Lack of User Input | 12.8% |
| 2. Incomplete Requirements & Specifications | 12.3% |
| 3. Changing Requirements & Specifications | 11.8% |
| 4. Lack of Executive Support | 7.5% |
| 5. Technology Incompetence | 7.0% |
| 6. Lack of Resources | 6.4% |
| 7. Unrealistic Expectations | 5.9% |
| 8. Unclear Objectives | 5.3% |
| 9. Unrealistic Time Frames | 4.3% |
| 10. New Technology | 3.7% |
| Other | 23.0% |

## Factors Making SD Fail



| Project Impaired Factors | % of Responses |
|---|---|
| 1. Incomplete Requirements | 13.1% |
| 2. Lack of User Involvement | 12.4% |
| 3. Lack of Resources | 10.6% |
| 4. Unrealistic Expectations | 9.9% |
| 5. Lack of Executive Support | 9.3% |
| 6. Changing Requirements & Specifications | 8.7% |
| 7. Lack of Planning | 8.1% |
| 8. Didn't Need It Any Longer | 7.5% |
| 9. Lack of IT Management | 6.2% |
| 10. Technology Illiteracy | 4.3% |
| Other | 9.9% |

**Source :** Standish Group

## Why is software development so difficult?

- Communication
  - Between customer and developer
  - Within development team
- Project characteristics
  - Advancing technology
  - Changing requirements
- Personnel characteristics
  - Personnel variability
  - High turnover
- Facilities and resources
- Management issues

# Reasons of software project failures

- Lack of high-level management support
- Lack of trust between customers and project team
- Too little involvement with the user community
- Poorly described requirements: Inadequate gathering, lack of user input, misunderstandings
- Estimation errors
- Poorly implemented development methodology
- Lack of risk management
- Planning, monitoring, control
- Staff turnover

F.P. Brooks Jr., The Mythical Man-Month, Essays on Software Engineering, Addison-Wesley, Reading, MA, 1975.
McCain, K. W., & Salvucci, L. J. (2006). How influential is Brooks' law? A longitudinal citation context analysis of Frederick Brooks' The Mythical Man-Month. *Journal of Information Science*, *32*(3), 277-295.
Verner, J. M., Overmyer, S. P., & McCain, K. W. (1999). In the 25years since The Mythical Man-Month what have we learned about project management?. *Information & Software Technology*, *41*(14), 1021-1026
Cerpa N., and Verner J., [2009], "Why did your project fail?" *Communications of the ACM*, December Vol 52. No 12, pp.130-134.

## Software Business

- Today's software development activities heavily rely on a project based approach.

- A project can be seen as a series of planned activities packed within a scope. The scope of a software project consists of
  - **Time:** How much time do we need to complete the project.
  - **Cost:** How much effort needed to complete the project on time.
  - **Quality:** What primary and secondary features and functionalities should be present regarding the time and budget.

- Software projects are generally carried out in phases (or stages) containing activities with the intent of better planning and management.

## Software Development Stages

- A very classical model called “waterfall” contains the following stages:
  1. Requirements Specification Phase
  2. Requirements Analysis phase
  3. Design phase
  4. Implementation phase
  5. Post-delivery maintenance
  6. Retirement

- Let’s work on a very basic example. Here are the questions you should ask for each stage of the software development for a mobile “alarm clock app”.

## Requirements Phase

- For the requirements phase, almost no technical detail should be considered in detail.
  - Explore the concept
  - Elicit the client’s requirements

- Software is treated as a black box, we enlist the features that we wish to see
  - Do we have snooze operation?
  - Should we be able to give alias to alarms?
  - Is there going to be a soft alarm?
  - Should we be able to save multiple alarms?
  - Do we support periodic alarms?
  - Should we be able to assign custom alarm sounds?
  - … and many more

# Analysis Phase

- In the analysis phase, primary requirements on the technical issues are analyzed in a broad perspective.
  - Analyze the client's requirements
  - Draw up the specification document
  - Draw up the software project management plan
  - "What the product is supposed to do"
- In this phase for the alarm clock app we ask questions like
  - What is the maximum snooze repetition, how much should we wait in between?
  - Should the user be able to edit snooze time?
  - How should we increase the sound in soft alarm, should we use a different melody?
  - How should we list multiple alarms?
  - Should we disable the periodic alarm in holidays? How should we get the holiday information?
  - ... and many more

# Design Phase

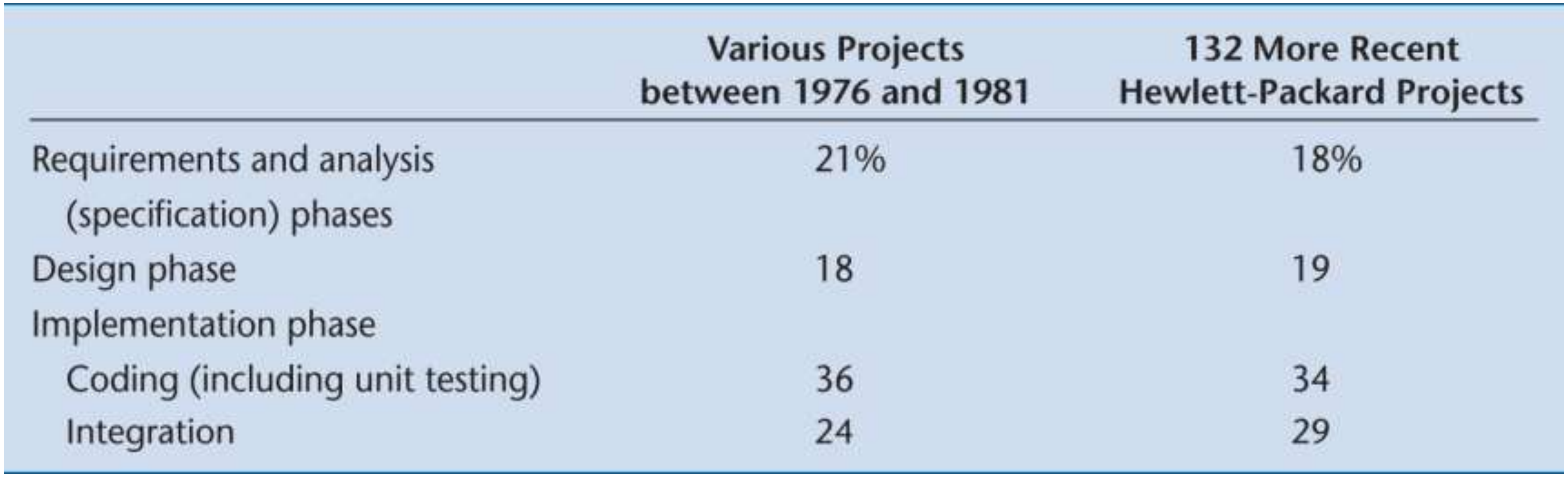
- In the design phase, most of the necessary decisions on the technical issues are made.
  - Architectural design, followed by
  - GUI design
  - Data and Functional design
- In this phase for the alarm clock app we discuss questions like
  - Where should we save the alarm parameters (local db, file, cloud)?
  - How should the alarm list look like?
  - How should the single alarm edit screen look like?
  - What kind of mechanism should we use to trigger alarm? Thread- daemon process?
  - Should we use a list or an array for the alarm list?
  - How should we cache the holiday dates?

## Rest of the Phases

- Implementation and Testing
  - Coding
  - Unit testing
  - Integration
  - Acceptance testing
- Post-delivery maintenance
  - Corrective maintenance
  - Perfective maintenance
  - Adaptive maintenance
- Retirement

# Cost of the Phases

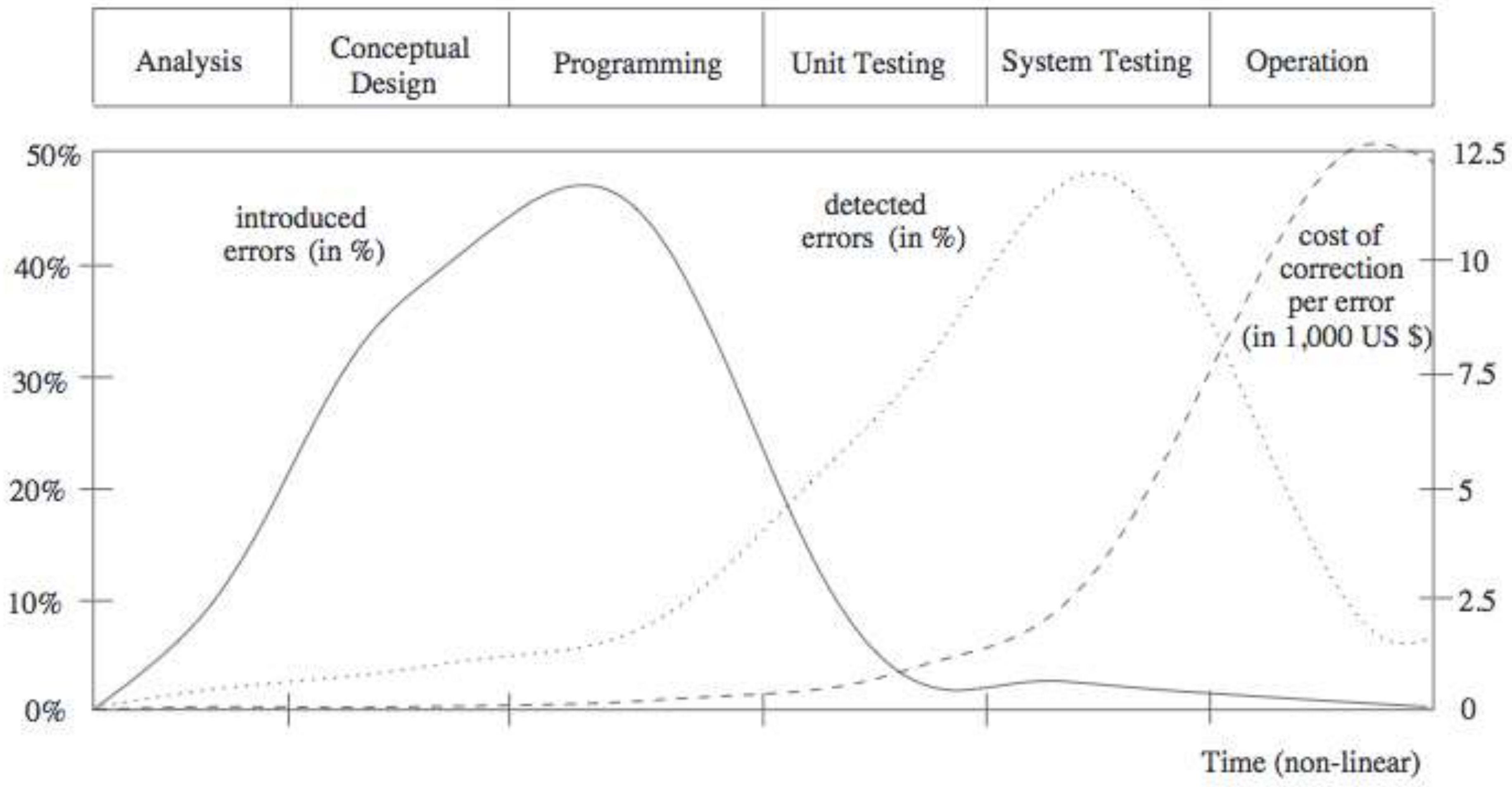
- Surprisingly, the costs of the classical phases have hardly changed



| | Various Projects between 1976 and 1981 | 132 More Recent Hewlett-Packard Projects |
|---|---|---|
| Requirements and analysis (specification) phases | 21% | 18% |
| Design phase | 18 | 19 |
| Implementation phase | | |
| Coding (including unit testing) | 36 | 34 |
| Integration | 24 | 29 |

# Cost of Faults

- The cost of detecting and correcting a fault at each phase



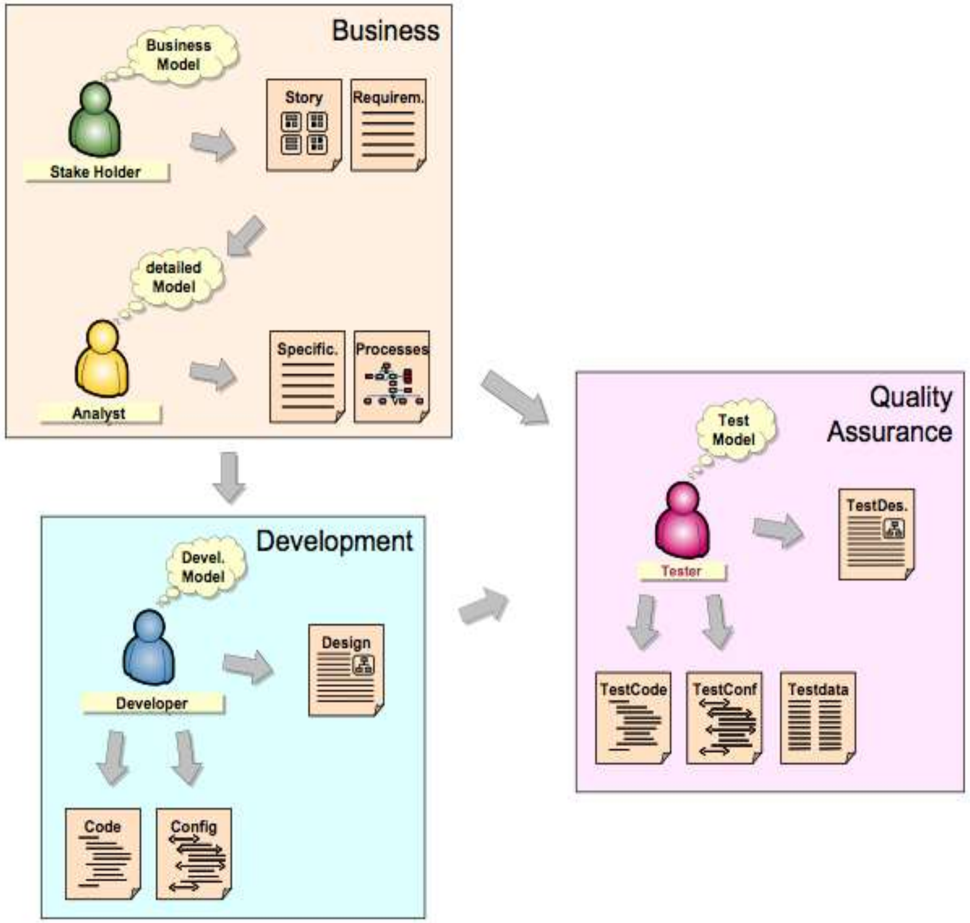
## Software Projects

- When carrying out a software project, social aspects are generally more important than technical aspects.

- **Stakeholder**: According to the Project Management Institute (PMI), "*the term project stakeholder refers to, 'an individual, group, or organization, who may affect, be affected by, or perceive itself to be affected by a decision, activity, or outcome of a project*"

# Software Project Stakeholders

- The most common software project stakeholders are listed below.
  - User
  - Customer
  - Project Manager
  - Team Leader
  - Analyst – Requirements Eng.
  - Configuration Manager
  - Designer – Architect
  - UI-Web designer
  - Programmer
  - Tester
  - QA people
  - IT people
  - DevOps – Maintenance People



## Communication Difficulties

How the customer explained it

# Project Scope

ꟻ 1.3 ꟻ

How to begin a project?

## Project Scope

- The very first thing that's done on a new project is the development of the project charter. That's the document that authorizes you to do your work.

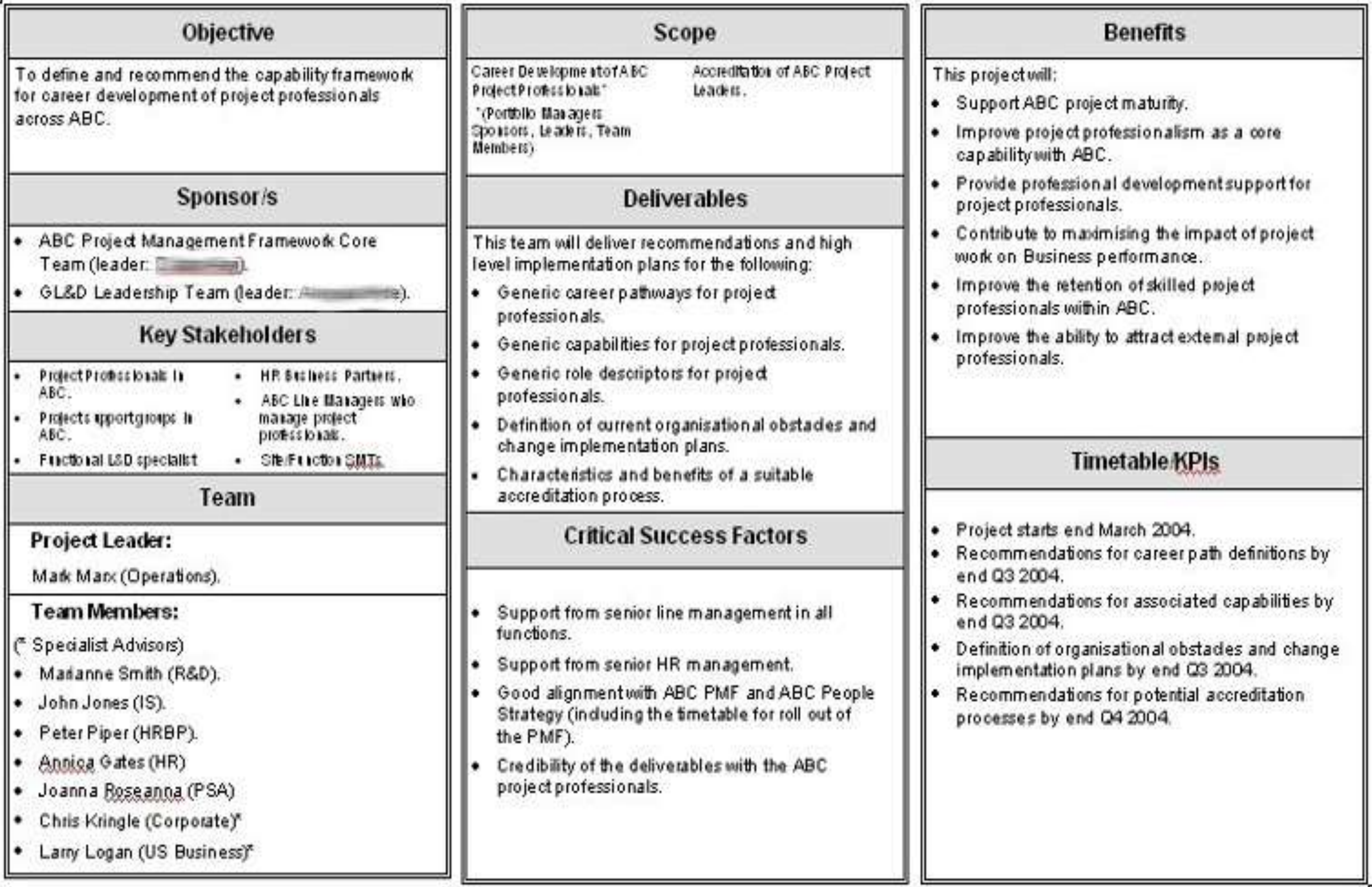
- Project Charter tells everyone in the company why the project is needed, and gives you the authority you need to make it happen.

- Then you identify stakeholders to figure out who is affected by the project and how to communicate with them

# Project Charter

- Even though this may change from case to case, a project charter (sometimes called as a “one-pager”) typically includes the following items in a single page:
  - Project Description
  - Project Objectives and Outcomes
  - Assigned Project Manager and Staff
  - Summary Milestone Schedule
  - Preliminary Cost Estimation
  - Preliminary Risks

# Project Charter



## Objective

To define and recommend the capability framework for career development of project professionals across ABC.

## Sponsor/s

- ABC Project Management Framework Core Team (leader: [redacted]).
- GL&D Leadership Team (leader: [redacted]).

## Key Stakeholders

- Project Professionals in ABC.
- Projects support groups in ABC.
- Functional L&D specialist
- HR Business Partners.
- ABC Line Managers who manage project professionals.
- Site/Function SMTs.

## Team

**Project Leader:**

Mark Marx (Operations).

**Team Members:**

(* Specialist Advisors)

- Marianne Smith (R&D).
- John Jones (IS).
- Peter Piper (HRBP).
- Annica Gates (HR)
- Joanna Roseanna (PSA)
- Chris Kringle (Corporate)*
- Larry Logan (US Business)*

## Scope

Career Development of ABC Project Professionals*

*(Portfolio Managers Sponsors, Leaders, Team Members)

Accreditation of ABC Project Leaders.

## Deliverables

This team will deliver recommendations and high level implementation plans for the following:

- Generic career pathways for project professionals.
- Generic capabilities for project professionals.
- Generic role descriptors for project professionals.
- Definition of current organisational obstacles and change implementation plans.
- Characteristics and benefits of a suitable accreditation process.

## Critical Success Factors

- Support from senior line management in all functions.
- Support from senior HR management.
- Good alignment with ABC PMF and ABC People Strategy (including the timetable for roll out of the PMF).
- Credibility of the deliverables with the ABC project professionals.

## Benefits

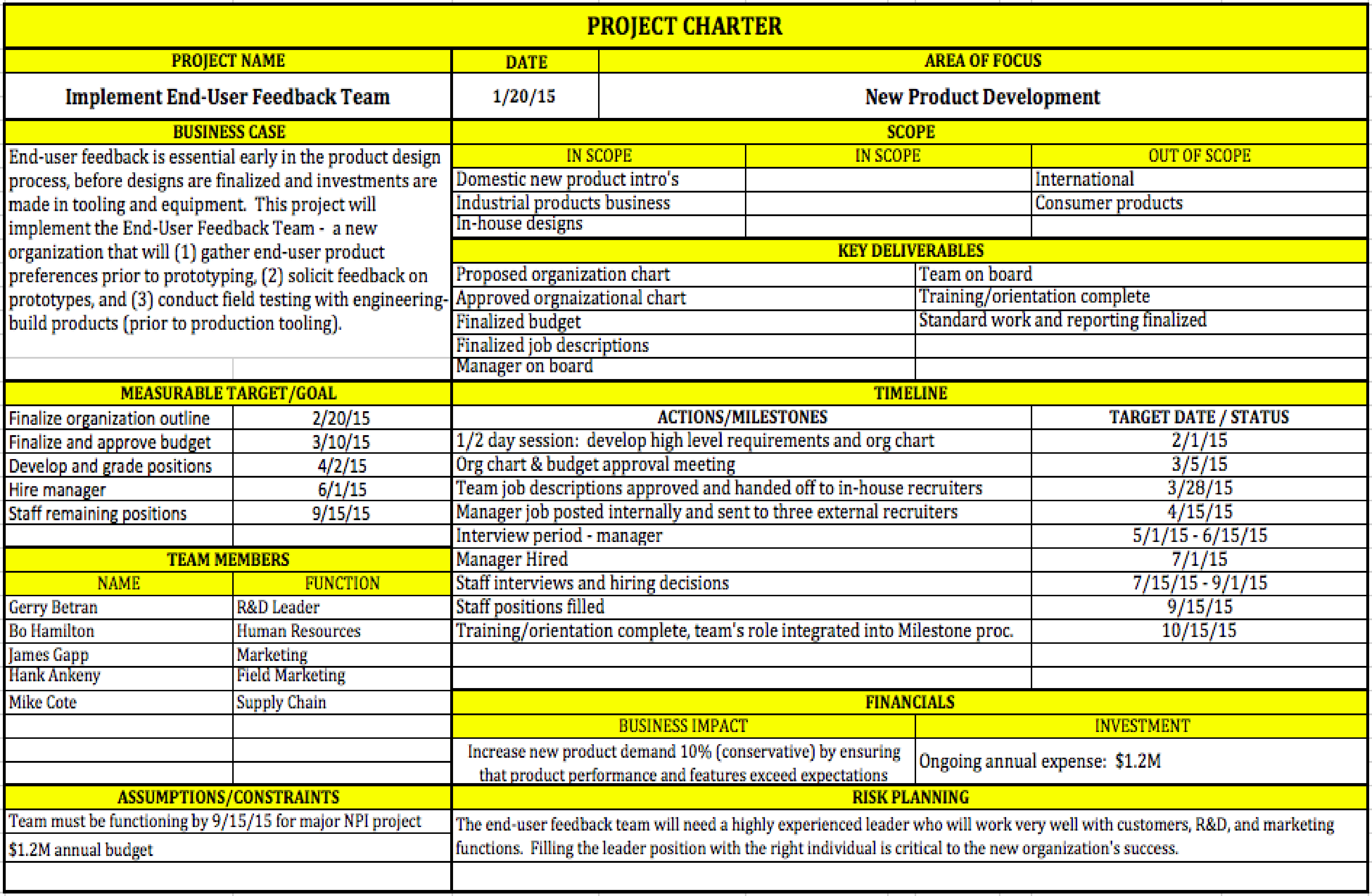
This project will:

- Support ABC project maturity.
- Improve project professionalism as a core capability with ABC.
- Provide professional development support for project professionals.
- Contribute to maximising the impact of project work on Business performance.
- Improve the retention of skilled project professionals within ABC.
- Improve the ability to attract external project professionals.

## Timetable/KPIs

- Project starts end March 2004.
- Recommendations for career path definitions by end Q3 2004.
- Recommendations for associated capabilities by end Q3 2004.
- Definition of organisational obstacles and change implementation plans by end Q3 2004.
- Recommendations for potential accreditation processes by end Q4 2004.

# Project Charter



## PROJECT CHARTER

| PROJECT NAME | DATE | AREA OF FOCUS |
|---|---|---|
| **Implement End-User Feedback Team** | 1/20/15 | **New Product Development** |

**BUSINESS CASE**

End-user feedback is essential early in the product design process, before designs are finalized and investments are made in tooling and equipment. This project will implement the End-User Feedback Team - a new organization that will (1) gather end-user product preferences prior to prototyping, (2) solicit feedback on prototypes, and (3) conduct field testing with engineering-build products (prior to production tooling).

**SCOPE**

| IN SCOPE | IN SCOPE | OUT OF SCOPE |
|---|---|---|
| Domestic new product intro's | | International |
| Industrial products business | | Consumer products |
| In-house designs | | |

**KEY DELIVERABLES**

| | |
|---|---|
| Proposed organization chart | Team on board |
| Approved orgnaizational chart | Training/orientation complete |
| Finalized budget | Standard work and reporting finalized |
| Finalized job descriptions | |
| Manager on board | |

**MEASURABLE TARGET/GOAL**

| | |
|---|---|
| Finalize organization outline | 2/20/15 |
| Finalize and approve budget | 3/10/15 |
| Develop and grade positions | 4/2/15 |
| Hire manager | 6/1/15 |
| Staff remaining positions | 9/15/15 |

**TIMELINE**

| ACTIONS/MILESTONES | TARGET DATE / STATUS |
|---|---|
| 1/2 day session: develop high level requirements and org chart | 2/1/15 |
| Org chart & budget approval meeting | 3/5/15 |
| Team job descriptions approved and handed off to in-house recruiters | 3/28/15 |
| Manager job posted internally and sent to three external recruiters | 4/15/15 |
| Interview period - manager | 5/1/15 - 6/15/15 |
| Manager Hired | 7/1/15 |
| Staff interviews and hiring decisions | 7/15/15 - 9/1/15 |
| Staff positions filled | 9/15/15 |
| Training/orientation complete, team's role integrated into Milestone proc. | 10/15/15 |

**TEAM MEMBERS**

| NAME | FUNCTION |
|---|---|
| Gerry Betran | R&D Leader |
| Bo Hamilton | Human Resources |
| James Gapp | Marketing |
| Hank Ankeny | Field Marketing |
| Mike Cote | Supply Chain |

**FINANCIALS**

| BUSINESS IMPACT | INVESTMENT |
|---|---|
| Increase new product demand 10% (conservative) by ensuring that product performance and features exceed expectations | Ongoing annual expense: $1.2M |

**ASSUMPTIONS/CONSTRAINTS**

Team must be functioning by 9/15/15 for major NPI project

$1.2M annual budget

**RISK PLANNING**

The end-user feedback team will need a highly experienced leader who will work very well with customers, R&D, and marketing functions. Filling the leader position with the right individual is critical to the new organization's success.

# Project Scope

- Once you have a good idea of what needs to be done, you need to track your scope as the project work is happening. Determining the project scope is setting goals for the project team and keep everybody on track.
  - Product scope means the features and functions of the product or service that you and your team are building.
  - Project scope is all of the work that needs to be done to make the product.
  - Scope creep means uncontrolled changes that cause the team to do extra work.

# Project Scope

- The five Scope Management processes that can be used in scope management are
  - Collecting the requirements to form a requirements document
  - Defining the Scope to form a Project Scope document
  - Creating a work breakdown structure
  - Consider change requests to modify project scope
  - Verify the scope iteratively by accepted deliverables

# Project Scope Document

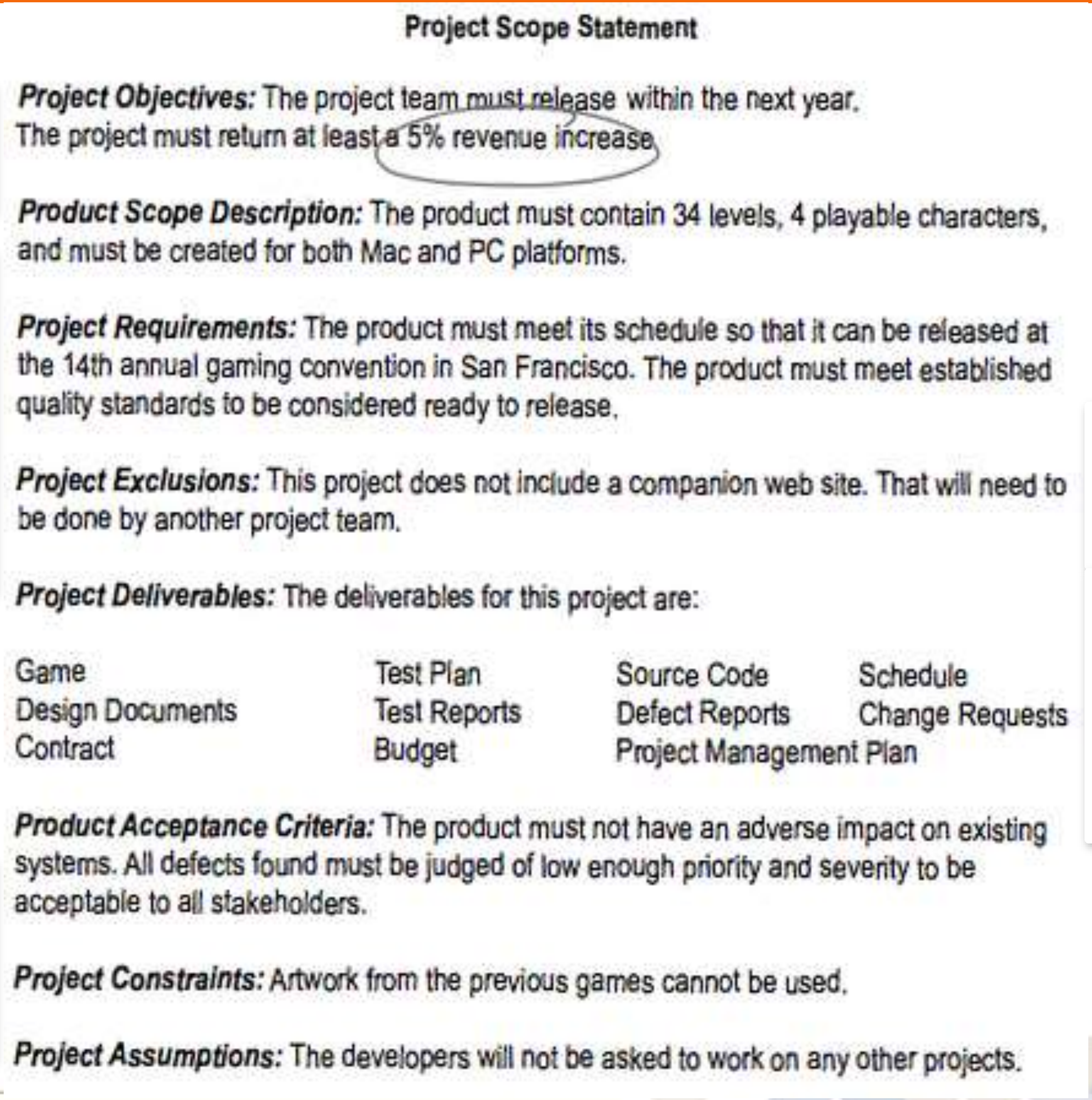
- The Project Scope document is created by considering the following documents:
  - Project Charter
  - Requirements Document
  - Organizational Templates/Forms
- When creating the project scope statement, you can perform the following actions
  - Stakeholder meetings → Output: Quantifiable goals
  - Product analysis
  - Alternatives Identification
  - Expert Judgement

# Project Scope Document

All the project objectives need to be measureable

Even though you probably can't fit ALL of the requirements here, there should be enough detail to let you keep on planning and refer back to it late

Constraints are known limitations. Assumptions are things you think are true.



**Project Scope Statement**

***Project Objectives:*** The project team must release within the next year. The project must return at least a 5% revenue increase.

***Product Scope Description:*** The product must contain 34 levels, 4 playable characters, and must be created for both Mac and PC platforms.

***Project Requirements:*** The product must meet its schedule so that it can be released at the 14th annual gaming convention in San Francisco. The product must meet established quality standards to be considered ready to release.

***Project Exclusions:*** This project does not include a companion web site. That will need to be done by another project team.

***Project Deliverables:*** The deliverables for this project are:

| | | | |
|---|---|---|---|
| Game | Test Plan | Source Code | Schedule |
| Design Documents | Test Reports | Defect Reports | Change Requests |
| Contract | Budget | Project Management Plan | |

***Product Acceptance Criteria:*** The product must not have an adverse impact on existing systems. All defects found must be judged of low enough priority and severity to be acceptable to all stakeholders.

***Project Constraints:*** Artwork from the previous games cannot be used.

***Project Assumptions:*** The developers will not be asked to work on any other projects.

This means looking for all the work the project DOESN'T include.

The deliverables listed here are EVERYTHING the project creates, including project management stuff.

## Wrap-up

- Building good quality software requires the coordination of various planning, engineering and management activities

- While beginning a software project, it is a common procedure to use project charters (or project offer or one-pager).

- Analyzing and keeping up with scope is also very important which is carried out during the whole project.