STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

SSW-555: Agile Methods for Software Development

Software Project Challenges

Prof. Jim Rowland
Software Engineering
School of Systems and Enterprises

Software development > coding

Although programming (coding) of software is an important part of any software effort, it is usually less than one quarter (1/4) of the total effort. On large projects it is less than one sixth (1/6).

Most software project failures are the result of:

- poor planning
- inadequate understanding of the requirements
- inadequate attention to quality
- failure to respond to problems until too late

Software project challenges

Feasibility and profitability – Should the project be done?

Requirements – What needs to be done?

Planning and controlling – Who does what and when

Implementation – Creating the software

Delivery and maintenance – Delivery to users, responding to needed changes

Support – Environment and tools needed for project

Teamwork – Making sure everyone is on the same page

Two examples to consider

Medical practice information system

- cloud-based
- highly reliable and secure
- opportunities for additional products and services
- Estimated size: 1 MLOC
- Estimated effort:
  700 staff-months by 40 staff over 21 months

Running App for Apple Watch

- speed to market is imperative
- reliability needs are low
- cost must be low
- Estimated size: 10 KLOC
- Estimated effort:
  8 staff-months by 2 staff over 4 months

Feasibility and profitability

What is the market?

- Who will pay for and use the system?
- How much will they pay?
- How will the market change?

How expensive will the project be?

- Effort
- Calendar time
- Available resources
- Purchased or leased resources

What are the risks?

Requirements

Which features should be produced?

What are the non-functional requirements?

- reliability
- security
- maintainability
- efficiency: run-time performance and space
- usability
- price
- time to market

Who knows what is needed?

Who decides what to implement?

Medical system
Non-functional Requirements

Reliability: must be available 24/7

Security: patient-sensitive information must be protected

Maintainability: expect to perform several updates in a controlled manner over the next few years

Efficiency: some run-time requirements

Usability: need to work with clients and users

Price: fairly expensive

Time to market: not too much urgency

Running App
Non-functional Requirements

Reliability: not an issue

Security: not an issue

Maintainability: not an issue

Efficiency: need to meet space limitations

Usability: expect to change it later

Price: very low

Time to market: critical

Planning and controlling

- Long-term planning
  - release schedule
  - lifetime of product or service
- Project planning
  - who does what
  - relationship and communication with stakeholders
  - scheduling of tasks
- Project management
  - reviewing and tracking progress
  - deciding when something is "done"
  - taking corrective action
- Project communication
  - what information is shared across project
  - how information is shared across project
  - how decisions are made and communicated

Implementation

High-level design and architecture

- creation
- communication to project staff
- maintenance

Low-level design

- (similar to high-level design)

Programming

- creation methods (e.g., pairs, coding standards)
- review
- maintenance

Verification and Validation

- types of reviews and tests
- who does what

Medical System Implementation

- High-level design and architecture
  - Carefully developed and maintained
- Low-level design
  - Use best practices (e.g., design patterns)
- Programming
  - Need good standards
- Verification and Validation
  - Extensive review and testing
  - Need separate team for testing

Running app implementation

High-level design and architecture

- Throw-away sketch

Low-level design

- Some comments in the code

Programming

- As fast as possible

Verification and Validation

- Let the users find the bugs

Delivery and maintenance

How often releases are delivered and installed?

Who does what?

How issues are detected, recorded, reported and tracked?

How change requests are managed?

Support

- Software development methods
  - training
  - mentors and guides
- Tools
  - version control and configuration management
  - editors and programming environments
  - compilers and code generators
  - static analysis tools
  - testing tools
  - bug tracking tools
- Physical space
  - project workspace
  - individual workspace

Teamwork

Software development is not a solitary activity

Communication between team members is essential

- some knowledge needs to be shared immediately
- no one knows everything

Collaboration amongst team members is essential

- group activities
- dependencies between activities and components

Project success depends on the success of the entire team

Medical system teamwork

Software development

- Teams of developers working on different sub-systems

Communication

- Regular meetings
- Published documents for designs, plans, user support

Collaboration activities

- Code reviews
- Protocols for shared sub-system modifications

Entire team

- Operations and support teams are critical

Running app teamwork

Software development

- Two developers talk to one another

Communication

- Two developers talk to one another

Collaboration activities

- Two developers talk to one another

Entire team

- Two developers talk to one another

Summary: Software Project Challenges

Feasibility and profitability – Should the project be done?

Requirements – What needs to be done?

Planning and controlling – Who does what and when

Implementation – Creating the software

Delivery and Maintenance – Delivery to users, responding to needed changes

Support – Environment and tools needed for project

Teamwork – Making sure everyone is on the same page