Software Engineering
G22.2440-001

Session 12 - Main Theme
Risk Management in Software Engineering Projects

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

• Review
• Architecture-Driven Iterative Process
• Project Planning and Estimation
• Cooperative Roles of Software Engineering and Project Management
• Summary
Summary of Previous Session

- Review of SDLC
- Traditional Life Cycle Models
- Alternative Techniques
- Architectural Principles
- Use Case Driven Development
- Extreme Programming
- Agile Software Development
- Roles and Types of Standards
  - ISO 12207: Life Cycle Standard
  - IEEE Standards for Software Engineering Processes and Specifications
- Summary
  - Course Assignments
  - Course Project (Project #1 extended)
  - Readings

Software Architecture

- A software architecture is a description of the subsystems and components of a software system and the relationships between them
- Subsystems and components are typically specified in different views to show the relevant functional and non-functional properties of a software system
- The software system is an artifact.
  - It is the result of the software design activity
Component

• A component is an encapsulated part of a software system
  • A component has an interface
• Components serve as the building blocks for the structure of a system
• At a programming-language level, components may be represented as modules, classes, objects or as a set of related functions

Subsystems

• A subsystem is a set of collaborating components performing a given task
  • A subsystem is considered a separate entity within a software architecture
• It performs its designated task by interacting with other subsystems and components…
Part I

Architecture-Driven Iterative Process

Section Summary

• Define Iterative development
• Define the Benefits of an iterative approach
  – Risk Mitigation
  – Accommodation of Change
  – Higher Quality
  – Discovery, Learning and Improvement
  – Opportunities for Reuse
• Introduce an example industry standard process: RUP™
• Discuss the role and contents of an iteration plan
What is Iterative Development?

- Iterative development is a software development lifecycle consisting of a number of iterations or cycles.
- Each iteration is composed of a loose set of some or all of the following activities:
  - Business Modeling
  - Requirements Management
  - Analysis and Design
  - Implementation
  - Test
  - Deployment
- While they look like a waterfall lifecycle they may be concurrent or simply omitted!

Why Develop Iteratively?

- Risks are mitigated earlier, and the results are incrementally evaluated.
- Change is recognized as unavoidable, short, manageable iterations provide scope for strategic and tactical adjustments.
- The product is incrementally developed with many checkpoints for identifying quality improvements.
- Processes can be incrementally validated and improved.
- Iteration supports incremental opportunities for reusing existing artifacts.
Risk Mitigation

• Processes, technologies, architectures, individuals are all exercised in early iterations
• This potentially identifies many of the risks “up-front”
• Perceived risks may be harmless, new risks may introduce themselves

The Change Monster

• Change is unavoidable
  – Scope creep
  – New Discoveries
  – Stakeholders changing their minds
• Iteration facilitates directional change, tactical adjustment to meet new conditions
• “Two steps forward, one step back” is still moving in a positive direction!
Reaching Higher Quality

• Iteration facilitates robustness
  – Errors are potentially discovered earlier
  – Product incrementally matures
  – Non-functional requirements e.g. performance, are addressed early

• Testing
  – iterations include continuous testing and integration

Process Improvement

• A given iteration enables us to reason about two things:
  – The state of the product
  – The state of the process

• We can evaluate how the process is performing on an iterative basis and make procedural / organizational changes as required.
Greater Opportunities for Reuse

- Incremental development opens up opportunities for identifying and applying reusable artifacts

- Early iterations help us manage the *build versus buy versus reuse* dilemma

Case Study: Rational Unified Process

- Also known as RUP™
- An emerging industry standard process model for iterative development
- RUP is a product that can (and must!) be tailored to meet the specific requirements of an organization and business domain
Overview of the Process

Disciplines
- Business Modeling
- Requirements
- Analysis and Design
- Implementation
- Test
- Deployment
- Configuration Management
- Project Management
- Environment

Overview

- One pass through the 4 phases of the lifecycle delivers a “release” of a product
- Number of iterations with each phase varies
  - Size, complexity, ....
- Each discipline occurs within each iteration
  - More or less emphasis depending on Phase
Achieve These “Best Practices”

- Iterative development
- Requirements management
- Develop using component-based architecture
- Visual modeling
- Pro-active quality assurance
- Effective change control and configuration management

4 Phases

- The Life cycle contains 4 *phases*
- One execution of a lifecycle delivers a *release*
- The end of each phase is a *milestone*
Inception Phase Milestone

- Objectives are defined for this life cycle
  - Stakeholders agree
    - scope, cost and schedule
  - Requirements identified and prioritized
  - Project plan in place
    - Schedule, scope, budget, risk, quality, procurement, staffing, communications, configuration

Elaboration Phase Milestone

- Baseline architecture is defined
  - Vision statement and requirements are stable
  - Architecture is stable
  - Iteration plans are ready
    - Upcoming iteration plan is well-defined
    - Cost estimates are realistic
  - Stakeholders agree on scope, cost and schedule
Construction Phase Milestone

- Initial operational capability is established
  - Product is stable and ready to be released
  - Stakeholders are ready to implement it
  - Planned expenditure for deployment is acceptable

Transition Phase Milestone

- System is delivered and is in production
  - User is satisfied
  - Resource estimates for ongoing operation are acceptable
Iterations Within Phases

• Each phase contains one or more iterations
  – Depending upon project size, complexity, ….

![Diagram of Iterations](image)

Iteration Focus

• In RUP the emphasis of an iteration changes by phase:
  – Inception and Elaboration
    • Management
    • Requirements
    • Analysis and Design Activity
  – Construction Phase
    • Design Activity
    • Implementation
    • Test
  – Transition Phase
    • Test
    • Deployment
An Architectural Emphasis

• While use-cases drive the overall development initiative, key design activities are based around the concept of architecture
• These are undertaken in all RUP phases

Architecture and Inception

• Compare and contrast competing architectural strategies as the requirements emerge
• Potentially develop “proof of concept” models and even executable prototype architectures to validate or invalidate an approach
Architecture and Elaboration

- “Baseline” a stable architecture
  - includes the process of developing, validating and refining a viable software architecture
- Develop an executable architectural prototype
  - used to identify and mitigate risks associated with functional and non-functional requirements (e.g. Scalability, Performance etc)

Architecture and Construction

- Provides a basis for implementation
  - The architecture represents and enforces the “architectural intent” of the system under implementation
- Incrementally refined as new issues are raised as part of implementation
  - New risks
  - Additional Technical Discoveries
  - Scope Creep
- Used to allocate developer resources
Architecture and Transition

• Resolution of critical defects against the architecture
  – What compromises or changes will need to be made to the existing architecture?
  – What are the implications of change at the system / subsystem level?

Inception  Elaboration  Construction  Transition

Project Planning

• Iteration Plans
  – Standard set of disciplines, workflows, tasks
“Coarse-Grain Planning”

- Plan for the phase (Inception, Elaboration…)
- This is done as part of Inception
- Estimate the accomplishment date for major milestones
  - What is each phase lifecycle objective?
  - What are the goals for the architecture?
  - When do we reach initial operating capability?
  - When do we release the product?
- What resources will be assigned to the project?
- What and when will we do specific iterations?
- Typically incorporated into an overall project plan

Iteration Issues

- How many iterations do we need?
- How long should each iteration be?
- What are the objectives of each iteration?
- How do I track and measure iteration progress?
How Many Iterations do we Need?

• This really depends on the size and scope of the project!
• This can change drastically if:
  – The project is large and complex
  – The domain is unknown
  – The organization is not used to iterative development processes

A Small RUP Project

• Inception – (1 iteration) development of a simple “proof of concept”… or nothing if an evolutionary project!
• Elaboration – (1 iteration) the production of a baseline executable architecture
• Construction – (1 iteration) build a beta release
• Transition – (1 iteration) build the final product
A Medium-Size RUP Project

- Inception – (1 iteration) the production of a proof of concept
- Elaboration (2 iterations) – the production of a prototype, the production of a baseline executable architecture
- Construction (2 iterations) – to expose a partial system, to build a beta release
- Transition – (1 iteration) beta to final release

A Large RUP Project

- Inception – (2 iterations) the production of a proof of concept and extensive prototyping
- Elaboration (3 iterations) – the exploration of project technologies, the production of a prototype, the production of a baseline executable architecture
- Construction (3 iterations) – to expose N partial systems, to build a beta release
- Transition – (2 iterations) extensive beta-testing and feedback iterations
How Long is an Iteration?

- This is typically shaped by estimation and experience
  - Have we done this before?
  - Results of COCOMO II / Wideband Delphi etc…
  - Clearly depends on goals and activities
- > 1 month …
  - Need to be clearly and carefully scoped
  - Best suited for construction where most risks / requirements are known
  - Not suitable for iterations with “discovery” goals
- > 3 months
  - Need well established goals to keep the iteration focused
- > 12 months
  - Additional risks as financial years may be spanned
  - Is this really iterative?

Iteration Planning

- Typically two iteration plans are in play at any given time:
  - The current iteration plan – used to track and judge progress with the current iteration
  - The next iteration plan – started in the later parts of the current iteration
- Typically built using traditional project management tools
  - Pert Charts
  - Gantt Charts
Building an Iteration Plan

- Define realistic criteria to evaluate the success of an iteration
- Identify empirically measurable goals for the development of concrete artifacts, these by implication will be associated with specific activities
- Assign resource and schedule estimates

“Time boxing”

- Typical iteration processes emphasize that meeting the iteration end date is the key concern
- Instead of “pushing out” the delivery date, the Project Manager is encouraged to manage iteration scope
- Reduction of scope is preferred to the extension of timelines
- It makes the results of an iteration more visible, allowing expectations and future iteration plans to be more “realistically” adjusted
Iteration Planning and Elaboration

- Risk Management
  - One of the principal objects
  - Risk Identification, Mitigation and Retirement
- Criticality
  - Risk aside, what needs to be built and in what order?
  - Fundamental functionality built first!
- Coverage
  - What areas will require development?
  - Deal with the risky bits as a priority!

Iteration Planning and Construction

- Focuses on the completion of specific use cases
- Planning undertaken in terms of use-case driven features
- Goals will be to complete functionality on a use case basis, to facilitate the development of (use case driven) test plans and suites
Iteration Plan for Construction

Iteration Planning and Transition

- Plans are made with respect to bug-fixes or late feature requests

- Typically focused on releases of the “finished” product with emphasis on Beta or Candidate releases
Iteration Overlap

- A degree of overlap is useful as it can be useful to keep all roles engaged!
- Remember: part of the advantage is learning from the previous iteration, overlapping inhibits this!

Iteration Parallelism

- Temptation to try and do N iterations at once
- This isn't ideal if there are co-dependencies between iterations
- One iteration may be held up by deliverables from the other!
Part II

Project Planning and Estimation

(Adapted from Chapter 4 of Ian Sommerville 2000 Software Engineering, 6th edition)

Project Management

- Project management scope:
  - organization, planning and scheduling of software projects
Section Objectives

- To introduce software project management and to describe its distinctive characteristics
- To discuss project planning and the planning process
- To show how graphical schedule representations are used by project management
- To discuss the notion of risks and the risk management process

Topics Covered

- Management activities
- Project planning
- Project scheduling
- Risk management
Software Project Management

• Concerned with activities involved in ensuring that software is delivered on time and on schedule and in accordance with the requirements of the organizations developing and procuring the software

• Project management is needed because software development is always subject to budget and schedule constraints that are set by the organization developing the software

Software Management Distinctions

• The product is intangible
• The product is uniquely flexible
• Software engineering is not recognized as an engineering discipline with the same status as mechanical, electrical engineering, etc.
• The software development process is not standardized
• Many software projects are 'one-off' projects
Management Activities

- Proposal writing
- Project planning and scheduling
- Project costing
- Project monitoring and reviews
- Personnel selection and evaluation
- Report writing and presentations

Management Commonalities

- These activities are not peculiar to software management
- Many techniques of engineering project management are equally applicable to software project management
- Technically complex engineering systems tend to suffer from the same problems as software systems
**Project Staffing**

- May not be possible to appoint the ideal people to work on a project
  - Project budget may not allow for the use of highly-paid staff
  - Staff with the appropriate experience may not be available
  - An organization may wish to develop employee skills on a software project
- Managers have to work within these constraints especially when (as is currently the case) there is an international shortage of skilled IT staff

**Project Planning**

- Probably the most time-consuming project management activity
- Continuous activity from initial concept through to system delivery. Plans must be regularly revised as new information becomes available
- Various different types of plan may be developed to support the main software project plan that is concerned with schedule and budget
Types of Project Plan

<table>
<thead>
<tr>
<th>Plan</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality plan</td>
<td>Describes the quality procedures and standards that will be used in a project.</td>
</tr>
<tr>
<td>Validation plan</td>
<td>Describes the approach, resources and schedule used for system validation.</td>
</tr>
<tr>
<td>Configuration management plan</td>
<td>Describes the configuration management procedures and structures to be used.</td>
</tr>
<tr>
<td>Maintenance plan</td>
<td>Predicts the maintenance requirements of the system, maintenance costs and effort required.</td>
</tr>
<tr>
<td>Staff development plan</td>
<td>Describes how the skills and experience of the project team members will be developed.</td>
</tr>
</tbody>
</table>

Project Planning Process

Establish the project constraints
Make initial assessments of the project parameters
Define project milestones and deliverables
while project has not been completed or cancelled loop
  Draw up project schedule
  Initiate activities according to schedule
  Wait ( for a while )
  Review project progress
  Revise estimates of project parameters
  Update the project schedule
  Re-negotiate project constraints and deliverables
if ( problems arise )then
  Initiate technical review and possible revision
end if
end loop
### Project Plan Structure

- Introduction
- Project organization
- Risk analysis
- Hardware and software resource requirements
- Work breakdown
- Project schedule
- Monitoring and reporting mechanisms

### Activity Organization

- Activities in a project should be organized to produce tangible outputs for management to judge progress
- *Milestones* are the end-point of a process activity
- *Deliverables* are project results delivered to customers
- The waterfall process allows for the straightforward definition of progress milestones
Milestones in the RE process

Project Scheduling

- Split project into tasks and estimate time and resources required to complete each task
- Organize tasks concurrently to make optimal use of workforce
- Minimize task dependencies to avoid delays caused by one task waiting for another to complete
- Dependent on project managers intuition and experience
The Project Scheduling Process

Scheduling Problems

- Estimating the difficulty of problems and hence the cost of developing a solution is hard
- Productivity is not proportional to the number of people working on a task
- Adding people to a late project makes it later because of communication overheads
- The unexpected always happens. Always allow contingency in planning
Bar Charts and Activity Networks

- Graphical notations used to illustrate the project schedule
- Show project breakdown into tasks. Tasks should not be too small. They should take about a week or two
- Activity charts show task dependencies and the critical path
- Bar charts show schedule against calendar time

Task Durations and Dependencies

<table>
<thead>
<tr>
<th>Task</th>
<th>Duration (days)</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>15</td>
<td>T1 (M1)</td>
</tr>
<tr>
<td>T4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>10</td>
<td>T2, T4 (M2)</td>
</tr>
<tr>
<td>T6</td>
<td>5</td>
<td>T1, T2 (M3)</td>
</tr>
<tr>
<td>T7</td>
<td>20</td>
<td>T1 (M1)</td>
</tr>
<tr>
<td>T8</td>
<td>25</td>
<td>T4 (M5)</td>
</tr>
<tr>
<td>T9</td>
<td>15</td>
<td>T3, T6 (M4)</td>
</tr>
<tr>
<td>T10</td>
<td>15</td>
<td>T5, T7 (M7)</td>
</tr>
<tr>
<td>T11</td>
<td>7</td>
<td>T9 (M6)</td>
</tr>
<tr>
<td>T12</td>
<td>10</td>
<td>T11 (M8)</td>
</tr>
</tbody>
</table>
Activity network

Activity timeline
Staff allocation

<table>
<thead>
<tr>
<th>Date</th>
<th>Fred</th>
<th>Jane</th>
<th>Anne</th>
<th>Jim</th>
<th>Mary</th>
</tr>
</thead>
<tbody>
<tr>
<td>4/7</td>
<td>T4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11/7</td>
<td></td>
<td>T8</td>
<td></td>
<td></td>
<td>T11</td>
</tr>
<tr>
<td>18/7</td>
<td>T1</td>
<td></td>
<td>T3</td>
<td>T6</td>
<td></td>
</tr>
<tr>
<td>25/8</td>
<td>T9</td>
<td>T11</td>
<td>T9</td>
<td>T10</td>
<td></td>
</tr>
<tr>
<td>1/8</td>
<td></td>
<td></td>
<td></td>
<td>T7</td>
<td></td>
</tr>
<tr>
<td>8/8</td>
<td>T11</td>
<td></td>
<td></td>
<td>T5</td>
<td></td>
</tr>
<tr>
<td>15/8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22/8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29/8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5/9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12/9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T12</td>
</tr>
<tr>
<td>19/9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Risk management

- Risk management is concerned with identifying risks and drawing up plans to minimize their effect on a project.
- A risk is a probability that some adverse circumstance will occur.
  - Project risks affect schedule or resources
  - Product risks affect the quality or performance of the software being developed
  - Business risks affect the organization developing or procuring the software
Software risks

<table>
<thead>
<tr>
<th>Risk</th>
<th>Risk type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Staff turnover</td>
<td>Project</td>
<td>Experienced staff will leave the project before it is finished.</td>
</tr>
<tr>
<td>Management change</td>
<td>Project</td>
<td>There will be a change of organizational management with different priorities.</td>
</tr>
<tr>
<td>Hardware unavailability</td>
<td>Project</td>
<td>Hardware which is essential for the project will not be delivered on schedule.</td>
</tr>
<tr>
<td>Requirements change</td>
<td>Project and product</td>
<td>There will be a larger number of changes to the requirements than anticipated.</td>
</tr>
<tr>
<td>Specification delays</td>
<td>Project and product</td>
<td>Specifications of essential interfaces are not available on schedule.</td>
</tr>
<tr>
<td>Size underestimate</td>
<td>Project and product</td>
<td>The size of the system has been underestimated.</td>
</tr>
<tr>
<td>CASE tool under-performance</td>
<td>Product</td>
<td>CASE tools which support the project do not perform as anticipated.</td>
</tr>
<tr>
<td>Technology change</td>
<td>Business</td>
<td>The underlying technology on which the system is built is superseded by new technology.</td>
</tr>
<tr>
<td>Product competition</td>
<td>Business</td>
<td>A competitive product is marketed before the system is completed.</td>
</tr>
</tbody>
</table>

The Risk Management Process

- Risk identification
  - Identify project, product and business risks
- Risk analysis
  - Assess the likelihood and consequences of these risks
- Risk planning
  - Draw up plans to avoid or minimize the effects of the risk
- Risk monitoring
  - Monitor the risks throughout the project
The Risk Management Process

Risk Identification

- Technology risks
- People risks
- Organizational risks
- Requirements risks
- Estimation risks
## Risks and Risk Types

<table>
<thead>
<tr>
<th>Risk type</th>
<th>Possible risks</th>
</tr>
</thead>
</table>
| Technology   | The database used in the system cannot process as many transactions per second as expected.  
               | Software components which should be reused contain defects which limit their functionality. |
| People       | It is impossible to recruit staff with the skills required.  
               | Key staff are ill and unavailable at critical times.  
               | Required training for staff is not available. |
| Organizational | The organization is restructured so that different management are responsible for the project.  
               | Organizational financial problems force reductions in the project budget. |
| Tools        | The code generated by CASE tools is inefficient.  
               | CASE tools cannot be integrated. |
| Requirements | Changes to requirements which require major design rework are proposed.  
               | Customers fail to understand the impact of requirements changes. |
| Estimation   | The time required to develop the software is underestimated.  
               | The rate of defect repair is underestimated.  
               | The size of the software is underestimated. |

## Risk Analysis

- Assess probability and seriousness of each risk
- Probability may be very low, low, moderate, high or very high
- Risk effects might be catastrophic, serious, tolerable or insignificant
Risk Analysis

<table>
<thead>
<tr>
<th>Risk</th>
<th>Probability</th>
<th>Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organizational financial problems force reductions in the project</td>
<td>Low</td>
<td>Catastrophic</td>
</tr>
<tr>
<td>budget</td>
<td></td>
<td></td>
</tr>
<tr>
<td>It is impossible to recruit staff with the skills required for the</td>
<td>High</td>
<td>Catastrophic</td>
</tr>
<tr>
<td>project</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key staff are ill at critical times in the project</td>
<td>Moderate</td>
<td>Serious</td>
</tr>
<tr>
<td>Software components which should be reused contain defects which</td>
<td>Moderate</td>
<td>Serious</td>
</tr>
<tr>
<td>limit their functionality</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changes to requirements which require major design rework are</td>
<td>Moderate</td>
<td>Serious</td>
</tr>
<tr>
<td>proposed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The organization is restructured so that different management</td>
<td>High</td>
<td>Serious</td>
</tr>
<tr>
<td>are responsible for the project</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The database used in the system cannot process as many transactions</td>
<td>Moderate</td>
<td>Serious</td>
</tr>
<tr>
<td>per second as expected</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The time required to develop the software is underestimated.</td>
<td>High</td>
<td>Serious</td>
</tr>
<tr>
<td>CASE tools cannot be integrated.</td>
<td>High</td>
<td>Tolerable</td>
</tr>
<tr>
<td>Customers fail to understand the impact of requirements changes.</td>
<td>Moderate</td>
<td>Tolerable</td>
</tr>
<tr>
<td>Required training for staff is not available.</td>
<td>Moderate</td>
<td>Tolerable</td>
</tr>
<tr>
<td>The rate of defect repair is underestimated.</td>
<td>Moderate</td>
<td>Tolerable</td>
</tr>
<tr>
<td>The size of the software is underestimated.</td>
<td>High</td>
<td>Tolerable</td>
</tr>
<tr>
<td>The code generated by CASE tools is inefficient</td>
<td>Moderate</td>
<td>Insignificant</td>
</tr>
</tbody>
</table>

Risk Planning

- Consider each risk and develop a strategy to manage that risk
- Avoidance strategies
  - The probability that the risk will arise is reduced
- Minimization strategies
  - The impact of the risk on the project or product will be reduced
- Contingency plans
  - If the risk arises, contingency plans are plans to deal with that risk
Risk Management Strategies

<table>
<thead>
<tr>
<th>Risk</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organizational financial problems</td>
<td>Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.</td>
</tr>
<tr>
<td>Recruitment problems</td>
<td>Alert customer of potential difficulties and the possibility of delays, investigate buying-in components.</td>
</tr>
<tr>
<td>Staff illness</td>
<td>Reorganize team so that there is more overlap of work and people therefore understand each other’s jobs.</td>
</tr>
<tr>
<td>Defective components</td>
<td>Replace potentially defective components with bought-in components of known reliability.</td>
</tr>
<tr>
<td>Requirements changes</td>
<td>Derive traceability information to assess requirements change impact, maximize information hiding in the design.</td>
</tr>
<tr>
<td>Organizational restructuring</td>
<td>Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.</td>
</tr>
<tr>
<td>Database performance</td>
<td>Investigate the possibility of buying a higher-performance database.</td>
</tr>
<tr>
<td>Underestimated development time</td>
<td>Investigate buying in components, investigate use of a program generator.</td>
</tr>
</tbody>
</table>

Risk Monitoring

- Assess each identified risks regularly to decide whether or not it is becoming less or more probable
- Also assess whether the effects of the risk have changed
- Each key risk should be discussed at management progress meetings
Risk Factors

<table>
<thead>
<tr>
<th>Risk type</th>
<th>Potential indicators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>Late delivery of hardware or support software, many reported technology problems</td>
</tr>
<tr>
<td>People</td>
<td>Poor staff morale, poor relationships amongst team member, job availability</td>
</tr>
<tr>
<td>Organizational</td>
<td>organisational gossip, lack of action by senior management</td>
</tr>
<tr>
<td>Tools</td>
<td>reluctance by team members to use tools, complaints about CASE tools, demands for higher-powered workstations</td>
</tr>
<tr>
<td>Requirements</td>
<td>many requirements change requests, customer complaints</td>
</tr>
<tr>
<td>Estimation</td>
<td>failure to meet agreed schedule, failure to clear reported defects</td>
</tr>
</tbody>
</table>

Key Points

- Good project management is essential for project success
- The intangible nature of software causes problems for management
- Managers have diverse roles but their most significant activities are planning, estimating and scheduling
- Planning and estimating are iterative processes which continue throughout the course of a project
Key Points
(continued)

• A project milestone is a predictable state where some formal report of progress is presented to management.
• Risks may be project risks, product risks or business risks
• Risk management is concerned with identifying risks which may affect the project and planning to ensure that these risks do not develop into major threats

Part III

Cooperative Roles of Software Engineering and Project Management
See: http://www.usenix.org/events/usenix-win2000/invitedtalks/34
(Windows A Software Engineering Odyssey)
Projects Launching Guidelines

Project Launch Workshop

- produce project plan
- define baseline control mechanisms
- launch in compliance with policies

Project Support & Mentoring

- periodic status checks, usually informal
- coaching and mentoring managers

Project Post-mortem Analysis

- lessons learned
- improvement needs
- asset and data capture

Project Launching Guidelines
(continued)

- Project Plan
  - Contents:
    - Project Organization (life cycle model, team model, roles,...)
    - Managerial Process (assumptions, dependencies, constraints, risk approach, reporting & reviews, staffing approach)
    - Technical Process (methods, tools, techniques, work product being built, reviews of products, and record collection)
    - Work Items, Schedule, & Budget (Work Breakdown Structure (WBS), resource requirements, budget, schedule)
  - Keep plan “alive”
    - The project plan is NOT “shelfware”, revisit and update it at regular intervals during project
Project Management

• Goals
  – Software delivered within budget
  – Software delivered within schedule
  – Software is built according to requirements

• Why?
  – Well-managed projects sometimes fail
  – Badly managed projects inevitably fail
  – Software development process is not standardized

Project Manager Responsibilities

• Proposal Writing
• Project Costing
• Project Planning & Scheduling
• Project Monitoring & Reviews
• Personnel Selection & Evaluation
• Report Writing & Presentations
Project Planning Process

Establish the project constraints
Make initial assessments of the project parameters
Define project milestones and deliverables

while project has not been completed or cancelled loop
  Draw up project schedule
  Initiate activities according to schedule
  Wait (for a while)
  Review project progress
  Revise estimates of project parameters
  Update the project schedule
  Re-negotiate project constraints and deliverables
  if (problems arise) then
    Initiate technical review and possible revision
  end if
end loop

So How Do We Do This?

• Spend time understanding the problem
• Estimate amount of effort required
  – Number of major functions
  – Difficulty of each function
• Develop schedule with built in safety nets
  – Increase estimates by some factor
  – Have a backup plan for worst case
  – Make sure schedule is realistic
• Revise schedule as project understanding increases
Estimation Overview

• Difficult & error prone
• Gradual refinement
  – At beginning of project, have a “fuzzy” idea of problem, therefore estimate of time and effort will be “fuzzy” too
  – Only as the project develops and the problem and solution become clearer, will the estimates increase in accuracy

• Estimation Process
  – Estimate the size of the product
    • Lines of code (LOC)
    • Function Points
    • Number of functions
  – Estimate the effort
    • Person-months
  – Estimate the schedule
    • Calendar time

From Estimation to Scheduling

• Refinement
  – Initial problem statement
  – Requirements Specification
  – High Level Design
  – Detailed Design Specification
  – Implementation

• Cases
  – Best Case
  – Most Likely Case
  – Current Case
  – Worst Case
Scheduling

- **Activities**
  - Split project into tasks
    - Estimate time & resources required
  - Organize tasks concurrently to make optimal use of workforce
  - Minimize task dependencies to avoid delays

- **Problems**
  - Estimating is difficult
  - Productivity is not proportional to the number of people
  - Adding people to a late project makes it later
  - The unexpected always happens - allow contingency

Scheduling

- Derived from estimated level of effort required
- Build in mid-project checkpoints
- Don’t forget testing & integration take time too
- Be realistic
  - Other classes
  - Outside work/activities
  - Eat & sleep
- Build in safety nets & backup plans
Project Plan

- Introduction
- Project organization
- Risk analysis
- Hardware and software resource requirements
- Work breakdown
  - Milestones - end of process activity
  - Deliverables - project results delivered to customer
- Project schedule
- Monitoring and reporting mechanisms

In Summary...

- Good project management is essential for project success
- Managers have diverse roles, but focus on
  - Planning
  - Estimating
  - Scheduling
- Planning and estimating are iterative processes
Part IV

Other Approaches to Planning

Gantt Chart

- List tasks
- Graphically represent dependencies among tasks
- Show duration and time period of each task
- Heavily dependent on prediction regarding:
  - Activities involved
  - Effort and time required
Gantt chart example

- Programmer working on a small software project

<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Requirement gathering</td>
<td>12/5/2002</td>
<td>12/6/2002</td>
<td>3d</td>
</tr>
<tr>
<td>2</td>
<td>Analysis</td>
<td>12/6/2002</td>
<td>12/9/2002</td>
<td>1d</td>
</tr>
<tr>
<td>3</td>
<td>Design</td>
<td>12/10/2002</td>
<td>12/11/2002</td>
<td>2d</td>
</tr>
<tr>
<td>4</td>
<td>Coding</td>
<td>12/12/2002</td>
<td>12/17/2002</td>
<td>4d</td>
</tr>
<tr>
<td>5</td>
<td>Testing</td>
<td>12/18/2002</td>
<td>12/31/2002</td>
<td>16d</td>
</tr>
</tbody>
</table>

Explicit start time, end time, and duration (in days)

Explicit calendar bar

Pert chart

- Alternative to Gantt chart
- Different perspective
  - Focuses on dependencies more than calendar time
- No fixed format

Start time

Duration

End time
Function Points

• FP is a unit for estimating time and effort
  – A.J. Albrecht of IBM, c. 1979

• Identify set of application activities (building blocks) and sum the weights assigned to each

• Building blocks identification and weight assignment depend critically on:
  – A world-wide database of FP practices
  – History of the firm
  – Experience of the FP estimators (certification)
Function Points

- Number of basic FP building blocks determined from application, not implementation:
  - Input files
  - Output files
  - Inquiries (snapshot request, no state change)
  - Internal files (transformations)
  - External interfaces (to other systems)

- Score for each block based on complexity: low, medium, high

- Unadjusted FP (UFP) is sum of the scores

### UFP Scores

<table>
<thead>
<tr>
<th></th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input files</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Output files</td>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>Inquiries</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Internal files</td>
<td>7</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>External interfaces</td>
<td>5</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>
**Function Points**

- 14 “technical factors” related to complexity
  - Grouped under 3 classes of complexity: system, I/O, application
  - Each factor ranked from 0 to 5
- Technical complexity factor (TCF)
  \[
  TCF = \left(\sum_{i=1}^{14} TCF_i\right) \times 0.01
  \]
- Adjusted function points (AFP or FP)
  \[
  FP = UFP \times (0.65 + TCF)
  \]

**Using FPs to Estimate Time/Effort**

- Previous measurements of FP per staff month or FP per calendar month
- Analogous to XP “project velocity”
- Applies to maintenance as well as development (considering “enhancement function points”)
- Tables available for lines of code per FP in various programming languages
Technical Factors

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Data communication</td>
<td>2.1 Reliable and transaction-oriented data management</td>
<td>3.1 Algorithms and processing ability</td>
</tr>
<tr>
<td>1.2 Distributed data processing</td>
<td>2.2 Online data management</td>
<td>3.2 Need to reuse the code later</td>
</tr>
<tr>
<td>1.3 Relevance of performance</td>
<td>2.3 Usability and efficiency of end user</td>
<td>3.3 Installation easiness</td>
</tr>
<tr>
<td>1.4 Configuration of hardware and software</td>
<td>2.4 Online update of the data</td>
<td>3.4 Startup, shutdown, and operation easiness</td>
</tr>
</tbody>
</table>

Partial (1) Partial (2)

Total

Partial (3)

UFP for Making Cappuccino

<table>
<thead>
<tr>
<th>Name</th>
<th>Type (building block)</th>
<th>Complexity</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milk</td>
<td>Input File</td>
<td>Medium</td>
<td>4</td>
</tr>
<tr>
<td>Coffee</td>
<td>Input File</td>
<td>Medium</td>
<td>4</td>
</tr>
<tr>
<td>Water</td>
<td>Input File</td>
<td>Low</td>
<td>3</td>
</tr>
<tr>
<td>Cappuccino</td>
<td>Output File</td>
<td>High</td>
<td>7</td>
</tr>
<tr>
<td>Water Temperature</td>
<td>Inquiry</td>
<td>Low</td>
<td>3</td>
</tr>
<tr>
<td>External Temperature</td>
<td>External Interface</td>
<td>Medium</td>
<td>7</td>
</tr>
</tbody>
</table>

Total Unadjusted Function Points 28
FP for Making Cappuccino

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Data communication</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>1.2 Distributed data processing</td>
<td>3</td>
<td>2.1 Reliable and transaction-oriented data management</td>
</tr>
<tr>
<td>1.3 Relevance of performances</td>
<td>4</td>
<td>2.2 Online data management</td>
</tr>
<tr>
<td>1.4 Configuration of the hardware and the software</td>
<td>4</td>
<td>2.3 Usability and efficiency of the end user</td>
</tr>
</tbody>
</table>

Partial (1) 16  Partial (2) 10  Partial (3) 13

Total = 39

FP for Making Cappuccino

• $FP = UFP \times (0.65 + TCF) = 28 \times (0.65 + (39 \times 0.01)) = 29.12$

• So what was the time/effort required last time your firm implement 29 FPs?
Constructive Cost Model

- COCOMO estimates best/likely/worst case range for cost, effort and schedule required to develop software
- Barry Boehm of TRW (now USC), c. 1981, updated c. 1995 (COCOMO II, original renamed COCOMO 81)
- Also based on empirical data from numerous projects, divided according to, e.g., 3 development modes: organic, semidetached, embedded [COCOMO 81]
- Assumes separate guestimate of lines of code (or “backfired” from function points), then considers additional factors

Development Modes

- **Organic**: relatively small software teams develop software in a highly familiar, in-house environment
- **Semidetached**: intermediate stage between the organic and embedded modes
- **Embedded**: Product must operate within (is embedded in) a strongly coupled complex of hardware, software, regulations, and operational procedures
Additional Factors

<table>
<thead>
<tr>
<th>Platform</th>
<th>Personnel</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Time Constraints</td>
<td>Analyst Capability</td>
<td>Use of Modern Programming Practices</td>
</tr>
<tr>
<td>Main Storage Constraints</td>
<td>Programmer Capability</td>
<td>Use of Software Tools</td>
</tr>
<tr>
<td>Platform Volatility</td>
<td>Applications Experience</td>
<td>Multi-site Development</td>
</tr>
<tr>
<td>Computer Turnaround Time</td>
<td>Platform Experience</td>
<td>Required Development Schedule</td>
</tr>
<tr>
<td></td>
<td>Language and Tool Experience</td>
<td>Classified Security Application</td>
</tr>
<tr>
<td></td>
<td>Personnel Continuity</td>
<td></td>
</tr>
</tbody>
</table>

COCOMO

- Polynomial model
  \[ \text{Effort}(\text{Size}) = A \times \text{Size}^B \]
  where \( A, B > 0 \)

- \( A \) and \( B \) are computed based on the development mode (or scale factors in COCOMO II) and additional factors
- Handbooks available as a guide for the calculations of \( A \) and \( B \)
- Continued data collection to improve prediction accuracy (questionnaires, software, NDAs)
Summary

• FP and COCOMO macro-estimation based partially on large historical database of contributing software projects from multiple domains and partially on in-house experience
• Use case and user story micro-estimation entirely in-house, usually based on same or similar projects by same or related project team

Part IV

Conclusion
Course Assignments

• Individual Assignments
  • Problems and reports based on case studies or exercises

• Project-Related Assignments
  • All assignments (other than the individual assessments) will correspond to milestones in the team project.
  • As the course progresses, students will be applying various methodologies to a project of their choice. The project and related software system should relate to a real-world scenario chosen by each team. The project will consist of inter-related deliverables which are due on a (bi-) weekly basis.
  • There will be only one submission per team per deliverable and all teams must demonstrate their projects to the course instructor.
  • A sample project description and additional details will be available under handouts on the course Web site.

Course Project

• Project Logistics
  • Teams will pick their own projects, within certain constraints: for instance, all projects should involve multiple distributed subsystems (e.g., web-based electronic services projects including client, application server, and database tiers). Students will need to come up to speed on whatever programming languages and/or software technologies they choose for their projects - which will not necessarily be covered in class.
  • Students will be required to form themselves into "pairs" of exactly two (2) members each; if there is an odd number of students in the class, then one (1) team of three (3) members will be permitted. There may not be any "pairs" of only one member! The instructor and TA(s) will then assist the pairs in forming "teams", ideally each consisting of two (2) "pairs", possibly three (3) pairs if necessary due to enrollment, but students are encouraged to form their own 2-pair teams in advance. If some students drop the course, any remaining pair or team members may be arbitrarily reassigned to other pairs/teams at the discretion of the instructor (but are strongly encouraged to reform pairs/teams on their own). Students will develop and test their project code together with the other member of their programming pair.
Sample Project Methodology
Very eXtreme Programming (VXP)

- After teams formed, 1/2 week to Project Concept
- 1/2 week to Revised Project Concept
- 2 to 3 iterations
- For each iteration:
  - 1/2 week to plan
  - 1 week to iteration report and demo

Sample Project Methodology
Very eXtreme Programming (VXP)
(continued)

- Requirements: Your project focuses on two application services
- Planning: User stories and work breakdown
- Doing: Pair programming, write test cases before coding, automate testing
- Demoing: 5 minute presentation plus 15 minute demo
- Reporting: What got done, what didn’t, what tests show
- 1st iteration: Any
- 2nd iteration: Use some component model framework
- 3rd iteration: Refactoring, do it right this time
Revised Project Concept (Tips)

1. Cover page (max 1 page)
2. Basic concept (max 3 pages): Briefly describe the system your team proposes to build. Write this description in the form of either user stories or use cases (your choice). Illustrations do not count towards page limits.
3. Controversies (max 1 page)

First Iteration Plan (Tips)

- Requirements (max 2 pages):
- Select user stories or use cases to implement in your first iteration, to produce a demo by the last week of class
- Assign priorities and points to each unit - A point should correspond to the amount of work you expect one pair to be able to accomplish within one week
- You may optionally include additional medium priority points to do “if you have time”
- It is acceptable to include fewer, more or different use cases or user stories than actually appeared in your Revised Project Concept
**First Iteration Plan (Tips)**

- Work Breakdown (max 3 pages):
- Refine as *engineering tasks* and assign to pairs
- Describe specifically what will need to be coded in order to complete each task
- Also describe what unit and integration tests will be implemented and performed
- You may need additional engineering tasks that do not match one-to-one with your user stories/use cases
- Map out a *schedule* for the next weeks
- Be realistic – demo has to been shown before the end of the semester

**2nd Iteration Plan (Tips): Requirements**

- Max 3 pages
- Redesign/reengineer your system to use a component framework (e.g., COM+, EJB, CCM, .NET or Web Services)
- Select the user stories to include in the new system
  - Could be identical to those completed for your 1st Iteration
  - Could be brand new (but explain how they fit)
- Aim to maintain project velocity from 1st iteration
- Consider what will require new coding vs. major rework vs. minor rework vs. can be reused “as is”
**2nd Iteration Plan (Tips): Breakdown**

- Max 4 pages
- Define engineering tasks, again try to maintain project velocity
- Describe new unit and integration testing
- Describe regression testing
  - Can you reuse tests from 1st iteration?
  - If not, how will you know you didn’t break something that previously worked?
- 2nd iteration report and demo to be presented before the end of the semester

**2nd Iteration Report (Tips): Requirements**

- Max 2 pages
- For each engineering task from your 2nd Iteration Plan, indicate whether it succeeded, partially succeeded (and to what extent), failed (and how so?), or was not attempted
- Estimate how many user story points were actually completed (these might be fractional)
- Discuss specifically your success, or lack thereof, in porting to or reengineering for your chosen component model framework(s)
2nd Iteration Report (Tips): Testing

• Max 3 pages
• Describe the general strategy you followed for unit testing, integration testing and regression testing
• Were you able to reuse unit and/or integration tests, with little or no change, from your 1st Iteration as regression tests?
• What was most difficult to test?
• Did using a component model framework help or hinder your testing?

Project Presentation and Demo

• All Iterations Due
• Presentation slides (optional)
Readings

• Readings
  • Slides and Handouts posted on the course web site
  • Documentation provided with business and application modeling tools
    (e.g., Popkin Software Architect)
  • SE Textbook: Chapters 21-25 (Part 4)

• Project Frameworks Setup (ongoing)
  • As per references provided on the course Web site

• Individual Assignment
  • See Session 4 Handout: “Assignment #3”

• Team Assignment
  • See Session 2 Handout: “Team Project Specification” (Part 1)

Next Session:
Business Model Engineering
Software Analysis and Design

• Traditional Data and Process Modeling Approaches
• From Requirements Analysis to Business and Application Models
• Business Model Capture Tools
• Process Modeling
• Capturing the Organization and Location Aspects
• Developing a Process Model
• Roles of Software Analysis and Design
• Object-Oriented Analysis and Design with UML
• Selecting and Combining Approaches
• Creating a Data Model
• Homework #3
• Project #1 (ongoing)
• Summary