The Process

Dr. Paul West

Department of Computer Science
College of Charleston

April 10, 2014

Learning Objectives

- Understand the purposes of planning and monitoring
- Distinguish strategies from plans, and understand their relation
- Understand the role of risks in planning
- Understand the potential role of tools in monitoring a quality process
- Understand team organization as an integral part of planning

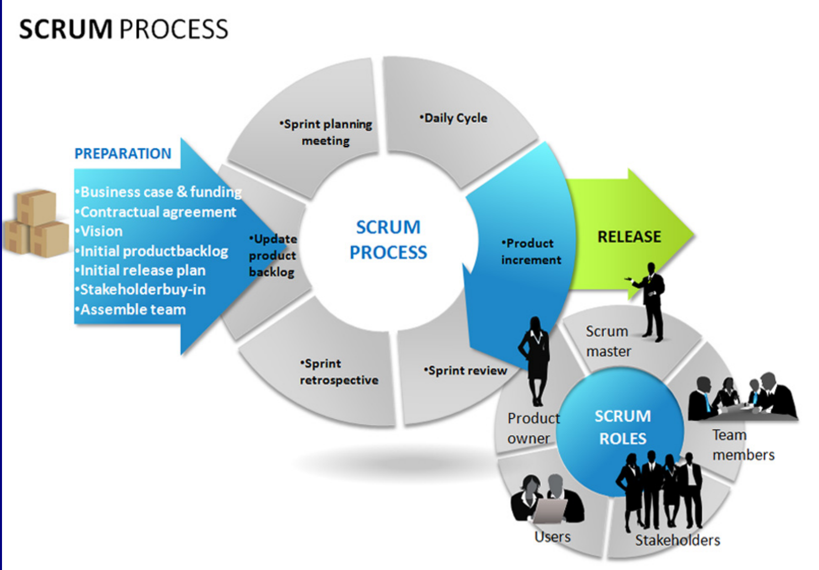
Planning and Monitoring

- Planning:
  - Scheduling activities (what steps? in what order?)
  - Allocating resources (who will do it?)
  - Devising unambiguous milestones for monitoring
- Monitoring: Judging progress against the plan
  - How are we doing?
- A good plan must have visibility :
  - Ability to monitor each step, and to make objective judgments of progress
  - Counter wishful thinking and denial

Quality and Process

- Quality process: Set of activities and responsibilities
  - focused primarily on ensuring adequate dependability
  - concerned with project schedule or with product usability
- A framework for
  - selecting and arranging activities
  - considering interactions and trade-offs
- Follows the overall software process in which it is embedded
  - Example: waterfall software process –> “V model”: unit testing starts with implementation and finishes before integration
  - Example: XP and agile methods –> emphasis on unit testing and rapid iteration for acceptance testing by customers

Scrum


SCRUM PROCESS
PREPARATION
•Business case & funding
•Contractual agreement
•Vision
•Initial productbacklog
•Initial release plan
•Stakeholderbuy-in
•Assemble team
•Sprint planning meeting
•Daily Cycle
•Update product backlog
SCRUM PROCESS
•Product increment
RELEASE
•Sprint retrospective
•Sprint review
Scrum master
Product owner
SCRUM ROLES
Team members
Users
Stakeholders

Overall Organization of a Quality Process

- Key principle of quality planning
  - the cost of detecting and repairing a fault increases as a function of time between committing an error and detecting the resultant faults
- therefore ...
  - an efficient quality plan includes matched sets of intermediate validation and verification activities that detect most faults within a short time of their introduction
- and ...
  - V&V steps depend on the intermediate work products and on their anticipated defects

Verification Steps for Intermediate Artifacts

- Internal consistency checks
  - compliance with structuring rules that define “well-formed” artifacts of that type
  - a point of leverage: define syntactic and semantic rules thoroughly and precisely enough that many common errors result in detectable violations
- External consistency checks
  - consistency with related artifacts
  - Often: conformance to a “prior” or “higher-level” specification
- Generation of correctness conjectures
  - Correctness conjectures: lay the groundwork for external consistency checks of other work products
  - Often: motivate refinement of the current product

Test and Analysis Strategy

- Lessons of past experience
  - an organizational asset built and refined over time
- Body of explicit knowledge
  - more valuable than islands of individual competence
  - amenable to improvement
  - reduces vulnerability to organizational change (e.g., loss of key individuals)
- Essential for
  - avoiding recurring errors
  - maintaining consistency of the process
  - increasing development efficiency

Considerations in Fitting a Strategy to an Organization

- Structure and size
  - example
    - Distinct quality groups in large organizations, overlapping of roles in smaller organizations
    - greater reliance on documents in large than small organizations
- Overall process
  - example
    - XP prescribes “test first” and pair programming fit with fluid specifications and rapid evolution
- Application domain
  - example
    - Safety critical domains may impose particular quality objectives and require documentation for certification (e.g,RTCA/DO-178B standard requires MC/DC coverage)

Elements of a Strategy

- Common quality requirements that apply to all or most products
  - unambiguous definition and measures
- Set of documents normally produced during the quality process
  - contents and relationships
- Activities prescribed by the overall process
  - standard tools and practices
- Guidelines for project staffing and assignment of roles and responsibilities

Test and Analysis Plan

- What quality activities will be carried out?
- What are the dependencies among the quality activities and between quality and other development activities?
- What resources are needed and how will they be allocated?
- How will both the process and the product be monitored?

Main Elements of a Plan

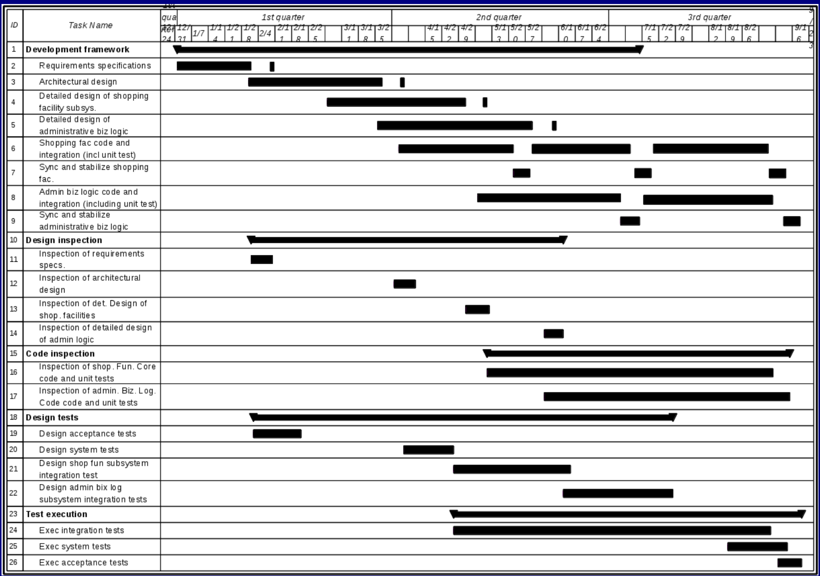
- Items and features to be verified
  - Scope and target of the plan
- Activities and resources
  - Constraints imposed by resources on activities
- Approaches to be followed
  - Methods and tools
- Criteria for evaluating results

Quality Goals

- Expressed as properties satisfied by the product
  - must include metrics to be monitored during the project
  - example: before entering acceptance testing, the product must pass comprehensive system testing with no critical or severe failures
  - not all details are available in the early stages of development
- Initial plan
  - based on incomplete information
  - incrementally refined

Task Schedule

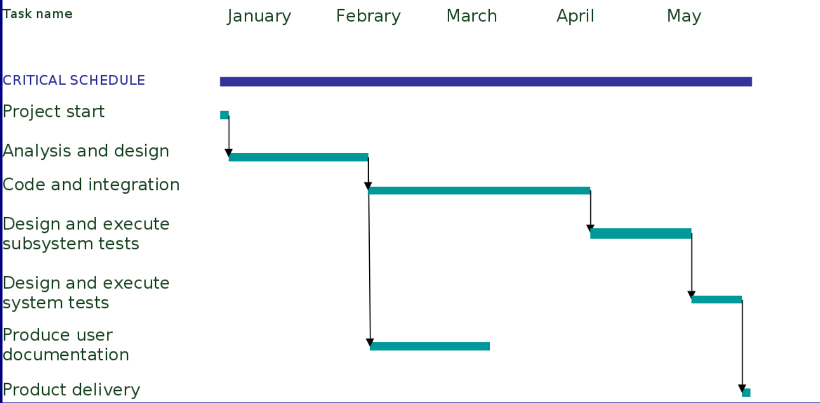
- Initially based on
  - quality strategy
  - past experience
- Breaks large tasks into subtasks
  - refine as process advances
- Includes dependencies
  - among quality activities
  - between quality and development activities
- Guidelines and objectives:
  - schedule activities for steady effort and continuous progress and evaluation without delaying development activities
  - schedule activities as early as possible
  - increase process visibility (how do we know we're on track?)


ID
Task Name
1st quarter
2nd quarter
3rd quarter
1 Development framework
2 Requirements specifications
3 Architectural design
4 Detailed design of shopping facility subsys.
5 Detailed design of administrative biz logic
6 Shopping fac code and integration (incl unit test)
7 Sync and stabilize shopping fac.
8 Admin biz logic code and integration (including unit test)
9 Sync and stabilize administrative biz logic
10 Design inspection
11 Inspection of requirements specs.
12 Inspection of architectural design
13 Inspection of det. Design of shop. facilities
14 Inspection of detailed design of admin logic
15 Code inspection
16 Inspection of shop. Fun. Core code and unit tests
17 Inspection of admin. Biz. Log. Code code and unit tests
18 Design tests
19 Design acceptance tests
20 Design system tests
21 Design shop fun subsystem integration test
22 Design admin bix log subsystem integration tests
23 Test execution
24 Exec integration tests
25 Exec system tests
26 Exec acceptance tests

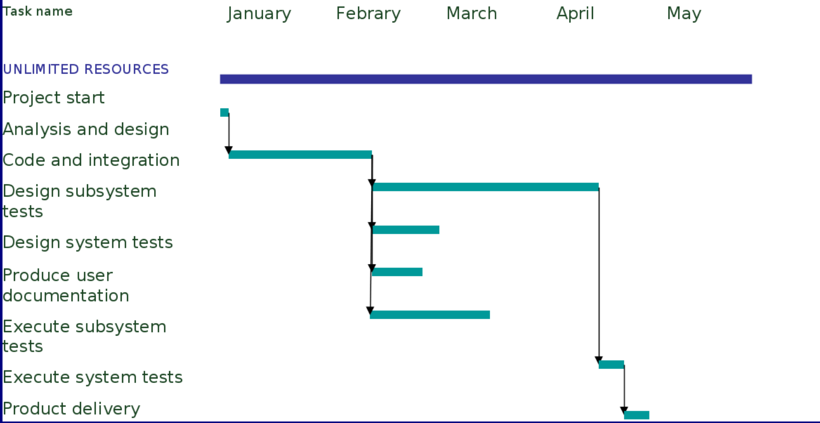
Schedule Risk

- critical path = chain of activities that must be completed in sequence and that have maximum overall duration
  - Schedule critical tasks and tasks that depend on critical tasks as early as possible to
    - provide schedule slack
    - prevent delay in starting critical tasks
- critical dependence = task on a critical path scheduled immediately after some other task on the critical path
  - May occur with tasks outside the quality plan (part of the project plan)
  - Reduce critical dependences by decomposing tasks on critical path, factoring out subtasks that can be performed earlier

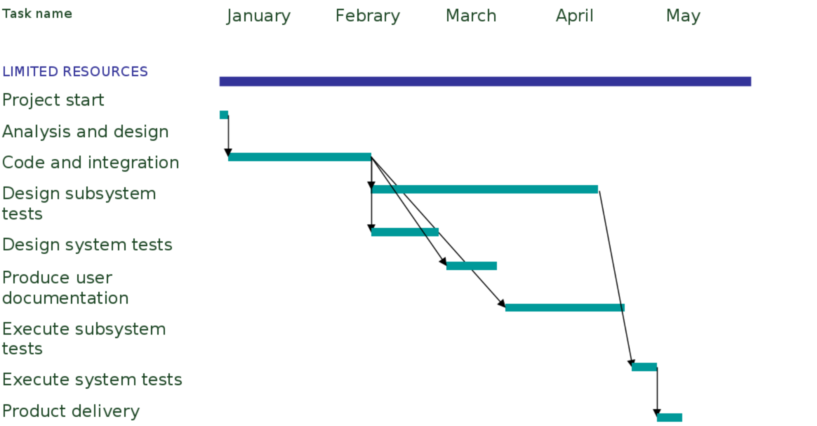
Reducing the Impact of Critical Paths



Reducing the Impact of Critical Paths



Reducing the Impact of Critical Paths



Risk Planning

- Risks cannot be eliminated, but they can be assessed, controlled, and monitored
- Generic management risk
  - personnel
  - technology
  - schedule
- Quality risk
  - development
  - execution
  - requirements

Personnel

Example Risks

- Loss of a staff member
- Staff member under-qualified for task

Control Strategies

- cross training to avoid over-dependence on individuals
- continuous education
- identification of skills gaps early in project
- competitive compensation and promotion policies and rewarding work
- including training time in project schedule

Technology

Example Risks

- High fault rate due to unfamiliar COTS component interface
- Test and analysis automation tools do not meet expectations

Control Strategies

- Anticipate and schedule extra time for testing unfamiliar interfaces.
- Invest training time for COTS components and for training with new tools
- Monitor, document, and publicize common errors and correct idioms.
- Introduce new tools in lower-risk pilot projects or prototyping exercises

Schedule

Example Risks

- Inadequate unit testing leads to unanticipated expense and delays in integration testing
- Difficulty of scheduling meetings makes inspection a bottleneck in development

Control Strategies

- Track and reward quality unit testing as evidenced by low fault densities in integration
- Set aside times in a weekly schedule in which inspections take precedence over other meetings and work
- Try distributed and asynchronous inspection techniques, with a lower frequency of face-to-face

Development

Example Risks

- Poor quality software delivered to testing group
- Inadequate unit test and analysis before committing to the code base

Control Strategies

- Provide early warning and feedback
- Schedule inspection of design, code and test suites
- Connect development and inspection to the reward system
- Increase training through inspection
- Require coverage or other criteria at unit test level

Test Execution

Example Risks

- Execution costs higher than planned
- Scarce resources available for testing

Control Strategies

- Minimize parts that require full system to be executed
- Inspect architecture to assess and improve testability
- Increase intermediate feedback
- Invest in scaffolding

Requirements

Example Risks

- High assurance critical requirements increase expense and uncertainty

Control Strategies

- Compare planned testing effort with former projects with similar criticality level to avoid underestimating testing effort
- Balance test and analysis
- Isolate critical parts, concerns and properties

Contingency Plan

- Part of the initial plan
  - What could go wrong? How will we know, and how will we recover?
- Evolves with the plan
- Derives from risk analysis
  - Essential to consider risks explicitly and in detail
- Defines actions in response to bad news
  - Plan B at the ready (the sooner, the better)

Process Monitoring

- Identify deviations from the quality plan as early as possible and take corrective action
- Depends on a plan that is
    - realistic
    - well organized
    - sufficiently detailed with clear, unambiguous milestones and criteria
- A process is visible to the extent that it can be effectively monitored

Orthogonal Defect Classification (ODC)

- Accurate classification schema
  - for very large projects
  - to distill an unmanageable amount of detailed information
- Two main steps
  - Fault classification
    - when faults are detected
    - when faults are fixed
  - Fault analysis

ODC Fault Classification

- When faults are detected
  - activity executed when the fault is revealed
  - trigger that exposed the fault
  - impact of the fault on the customer
- When faults are fixed
  - Target: entity fixed to remove the fault
  - Type: type of the fault
  - Source: origin of the faulty modules (in-house, library, imported, outsourced)
  - Age of the faulty element (new, old, rewritten, re-fixed code)

ODC Fault Analysis Example 1

- Distribution of fault types versus activities
  - Different quality activities target different classes of faults EX:
  - algorithmic faults are targeted primarily by unit testing.
    - a high proportion of faults detected by unit testing should belong to this class
  - proportion of algorithmic faults found during unit testing
    - unusually small
    - larger than normal
    - unit tests may not have been well designed
  - proportion of algorithmic faults found during unit testing unusually large
  - integration testing may not focused strongly enough on interface faults

ODC Fault Analysis Example 2

- Distribution of triggers over time during field test
  - Faults corresponding to simple usage should arise early during field test, while faults corresponding to complex usage should arise late.
  - The rate of disclosure of new faults should asymptotically decrease
  - Unexpected distributions of triggers over time may indicate poor system or acceptance test
    - Triggers that correspond to simple usage reveal many faults late in acceptance testing
    - The sample may not be representative of the user population
    - Continuously growing faults during acceptance test
    - System testing may have failed

ODC Fault Analysis Example 3

- Age distribution over target code
  - Most faults should be located in new and rewritten code
  - The proportion of faults in new and rewritten code with respect to base and re-fixed code should gradually increase
  - Different patterns
  - may indicate holes in the fault tracking and removal process
  - may indicate inadequate test and analysis that failed in revealing faults early
  - Example
    - increase of faults located in base code after porting
    - may indicate inadequate tests for portability

ODC Fault Analysis Example 4

- Distribution of fault classes over time
  - The proportion of missing code faults should gradually decrease
  - The percentage of extraneous faults may slowly increase, because missing functionality should be revealed with use
    - increasing number of missing faults
    - may be a symptom of instability of the product
    - sudden sharp increase in extraneous faults
    - may indicate maintenance problems

Improving the Process

- Many classes of faults that occur frequently are rooted in process and development flaws
  - examples
    - Shallow architectural design that does not take into account resource allocation can lead to resource allocation faults
    - Lack of experience with the development environment, which leads to misunderstandings between analysts and programmers on rare and exceptional cases, can result in faults in exception handling.
- The occurrence of many such faults can be reduced by modifying the process and environment
  - examples
    - Resource allocation faults resulting from shallow architectural design can be reduced by introducing specific inspection tasks
    - Faults attributable to inexperience with the development environment can be reduced with focused training

Improving Current and Next Processes

- Identifying weak aspects of a process can be difficult
- Analysis of the fault history can help software engineers build a feedback mechanism to track relevant faults to their root causes
  - Sometimes information can be fed back directly into the current product development
  - More often it helps software engineers improve the development of future products

Root cause analysis (RCA)

- Technique for identifying and eliminating process faults
  - First developed in the nuclear power industry; used in many fields.
- Four main steps
  - What are the faults?
  - When did faults occur? When, and when were they found?
  - Why did faults occur?
  - How could faults be prevented?

What are the faults?

- Identify a class of important faults
- Faults are categorized by
  - severity = impact of the fault on the product
  - Kind
    - No fixed set of categories; Categories evolve and adapt
    - Goal: Identify the few most important classes of faults and remove their causes
    - Goal: Differs from ODC: Not trying to compare trends for different classes of faults, but rather focusing on a few important classes

Pareto Distribution (80/20)

- Pareto rule (80/20)
  - in many populations, a few (20%) are vital and many (80%) are trivial
- Fault analysis
  - 20% of the code is responsible for 80% of the faults
  - Faults tend to accumulate in a few modules: identifying potentially faulty modules can improve the cost effectiveness of fault detection
  - Some classes of faults predominate: removing the causes of a predominant class of faults can have a major impact on the quality of the process and of the resulting product

Why did faults occur?

- Core RCA step
  - trace representative faults back to causes
  - objective of identifying a “root” cause
- Iterative analysis
  - explain the error that led to the fault
  - explain the cause of that error
  - explain the cause of that cause
  - ...
- Rule of thumb
  - “ask why six times”

Example of fault tracing

- Tracing the causes of faults requires experience, judgment, and knowledge of the development process
- examples
  - most significant class of faults = memory leaks
  - cause = forgetting to release memory in exception handlers
  - cause = lack of information: "Programmers can't easily determine what needs to be cleaned up in exception handlers"
  - cause = design error: "The resource management scheme assumes normal flow of control"
  - root problem = early design problem: "Exceptional conditions were an afterthought dealt with late in design"

How could faults be prevented?

- Many approaches depending on fault and process:
- From lightweight process changes
  - Ex:adding consideration of exceptional conditions to a design inspection checklist
- To heavyweight changes:
  - Ex: making explicit consideration of exceptional conditions a part of all requirements analysis and design steps
- **Goal is not perfection, but cost-effective improvement**

The Quality Team

- The quality plan must assign roles and responsibilities to people
- assignment of responsibility occurs at
  - strategic level
    - test and analysis strategy
    - structure of the organization
    - external requirements (e.g., certification agency)
  - tactical level
    - test and analysis plan

Roles and Responsibilities at Tactical Level

- balance level of effort across time
- manage personal interactions
- ensure sufficient accountability that quality tasks are not easily overlooked
- encourage objective judgment of quality
- prevent it from being subverted by schedule pressure
- foster shared commitment to quality among all team members
- develop and communicate shared knowledge and values regarding quality

Alternatives in Team Structure

- Conflicting pressures on choice of structure
  - Ex:autonomy to ensure objective assessment
  - Ex:cooperation to meet overall project objectives
- Different structures of roles and responsibilities
  - same individuals play roles of developer and tester
  - most testing responsibility assigned to a distinct group
  - some responsibility assigned to a distinct organization
- Distinguish
  - oversight and accountability for approving a task
  - responsibility for actually performing a task

Roles and responsibilities pros and cons

- Same individuals play roles of developer and tester
  - potential conflict between roles
  - Ex:a developer responsible for delivering a unit on schedule
  - Ex:responsible for integration testing that could reveal faults that delay delivery
  - requires countermeasures to control risks from conflict
- Roles assigned to different individuals
  - Potential conflict between individuals
  - Ex:developer and a tester who do not share motivation to deliver a quality product on schedule
  - requires countermeasures to control risks from conflict

Independent Testing Team

- Minimize risks of conflict between roles played by the same individual
  - EX: project manager with schedule pressures cannot
    - bypass quality activities or standards
    - reallocate people from testing to development
    - postpone quality activities until too late in the project
- Increases risk of conflict between goals of the independent quality team and the developers
- Plan
  - should include checks to ensure completion of quality activities
    - Ex: developers perform module testing
    - Ex: independent quality team performs integration and system testing
    - Ex: quality team should check completeness of module tests

Managing Communication

- Testing and development teams must share the goal of shipping a high-quality product on schedule
  - testing team
    - must not be perceived as relieving developers from responsibility for quality
    - should not be completely oblivious to schedule pressure
- Independent quality teams require a mature development process
  - Test designers must
    - work on sufficiently precise specifications
    - execute tests in a controllable test environment
- Versions and configurations must be well defined
- Failures and faults must be suitably tracked and monitored across versions

Testing within XP

- Full integration of quality activities with development
  - Minimize communication and coordination overhead
  - Developers take full responsibility for the quality of their work
  - Technology and application expertise for quality tasks match expertise available for development tasks
- Plan
  - check that quality activities and objective assessment are not easily tossed aside as deadlines loom
  - Ex: XP “test first” together with pair programming guard against some of the inherent risks of mixing roles

Outsourcing Test and Analysis

- (Wrong) motivation
  - testing is less technically demanding than development and can be carried out by lower-paid and lower-skilled individuals
- Why wrong
  - confuses test execution (straightforward) with analysis and test design (as demanding as design and programming)
- A better motivation
  - to maximize independence
    - and possibly reduce cost as (only) a secondary effect
- The plan must define
  - milestones and delivery for outsourced activities
  - checks on the quality of delivery in both directions

Summary

- Planning is necessary to
  - order, provision, and coordinate quality activities
    - coordinate quality process with overall development
    - includes allocation of roles and responsibilities
  - provide unambiguous milestones for judging progress
- Process visibility is key
  - ability to monitor quality and schedule at each step
    - intermediate verification steps: because cost grows with time between error and repair
  - monitor risks explicitly, with contingency plan ready
- Monitoring feeds process improvement
  - of a single project, and across projects

Choose 2 exercises from the end of Chapter 20.
Due April 23, 2014 2359 in the dropbox

Reading

Chapter 21 & 22.