# Software process models


## Motivation

- We have analyzed activities
  - Requirement, design, ..
- And the techniques that can be used to perform them
  - UML modeling, prog languages, WB BB testing, ..
- But how to organize all?

SOftEng
http://softeng.polito.it

## Outline

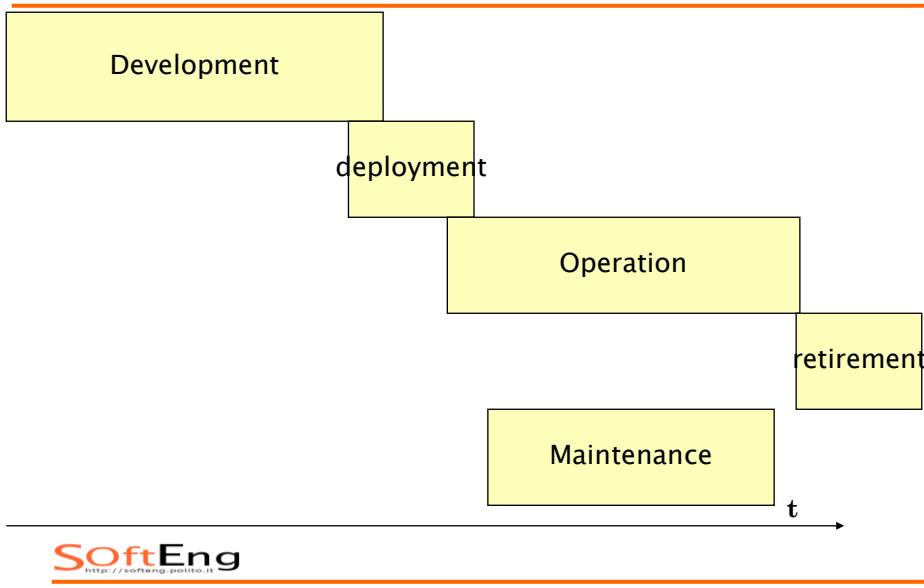
- Phases and Activities
- Processes, process models
- Projects
- Selection of process for project

SOftEng
http://softeng.polito.it

## Phases and activities

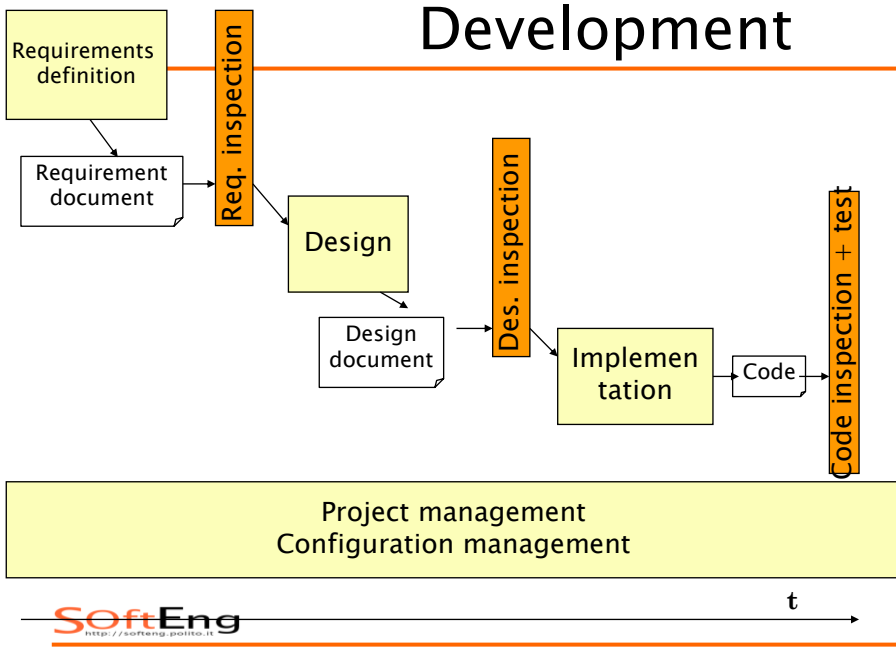
SOftEng
http://softeng.polito.it

# Main Phases




SOftEng
http://softeng.polito.it

# Development



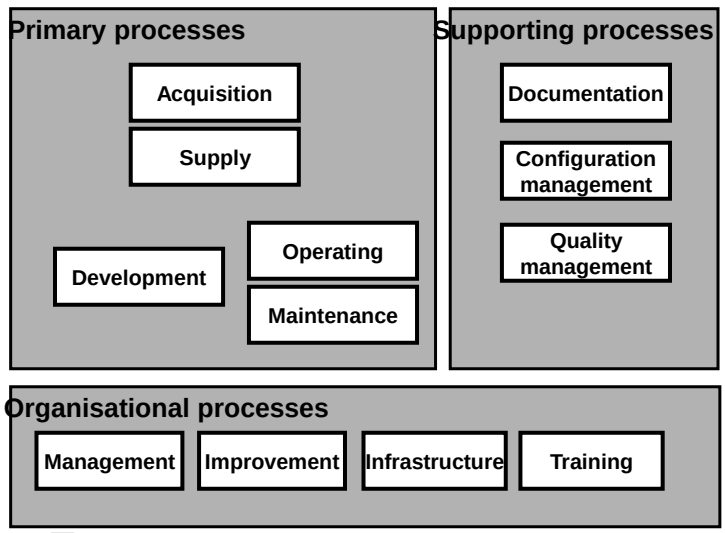

SOftEng
http://softeng.polito.it

# ISO/IEC 12207

- ISO == Int Standard Organization
  - Identifies processes
  - Identifies entities responsibles of processes
  - Identifies products of processes

SOftEng
http://softeng.polito.it

# ISO/IEC 12207




SOftEng
http://softeng.polito.it

## Primary processes

- Acquisition (manage suppliers)
- Supply (interaction with customer)
- Development (develop sw)
- Operation (deploy, operate service)
- Maintenance

SOftEng
http://softeng.polito.it

## Supporting

- Documentation of product
- Configuration management
- Quality assurance
  - Verification and Validation
  - Reviews with customer
  - Internal audits
  - Problem analysis and resolution

SOftEng
http://softeng.polito.it

# Organizational

- Project management
- Infrastructure management
  - Facilities, networks, tools
- Process monitoring and improvement
- Training

SOftEng
http://softeng.polito.it

# Ex. Software development

- Activity 5.3 Software development is decomposed in tasks
  - 5.3.1 Process Instantiation
  - 5.3.2 System requirements analysis
  - 5.3.3 System architecture definition
  - 5.3.4 Software requirements analysis
  - 5.3.5 Software architecture definition
  - 5.3.6 Software detailed design
  - 5.3.7 Coding and unit testing
  - 5.3.8 Integration of software units
  - 5.3.9 Software validation
  - 5.3.10 System integration
  - 5.3.11 System validation

SOftEng
http://softeng.polito.it

# V&V Tasks

- Coding and verification of components (5.3.7.)
- Integration of components (5.3.8.)
- Validation of software (5.3.9.)
- System Integration (5.3.10.)
- System validation (5.3.11.)

SOftEng http://softeng.polito.it

# Subtasks

- Coding and verification of components (5.3.7.)
  - Definition of test data and test procedures (5.3.7.1.)
  - Execute and document tests (5.3.7.2.)
  - Update documents, plan integration tests (5.3.7.4.)
  - Evaluate tests (5.3.7.5.)
- Integration of components (5.3.8.)
  - Definition of integration test plan (5.3.8.1.)
  - Execute and document tests (5.3.8.2.)
  - Update documents, plan validation tests(5.3.8.4.)
  - Evaluate tests (5.3.8.5.)

SOftEng http://softeng.polito.it

## ISO 12207

- Only list of activities
- Independent of process model
  - Waterfall, iterative, ..
- Independent of technology
- Independent of application domain
- Independent of documentation

SOftEng
http://softeng.polito.it

## Process models

SOftEng
http://softeng.polito.it

## How to organize activities?

- What?
  - Activities, documents
- Who does what?
  - Roles and responsibilities
- When?
  - Temporal constraints between activities
- Under constraints
  - Laws, standards

SOftEng http://softeng.polito.it

## Software process

- Activities
  - Requirement, design, ..
- Products
  - Requirement document, design document ..
- Roles
  - Project manager, analyst, tester, ..
- Process model
  - Temporal constraints on how activities should be executed, responsibilities

SOftEng http://softeng.polito.it

## Process models

- From standards / documents
  - ISO 15288 (system engineering activities)
  - ISO 12207 (sw engineering activities)
  - ISO 9001, ISO 9000-3
  - CMM-I
- From literature
  - Waterfall, RUP, Agile,

SOftEng http://softeng.polito.it

## Mature company approach

- Company process model
  - Suggested / required activities, documents, milestones
- Project process model
  - (see 5.3.1 Process Instantiation in 12207)
  - Project defines specific model to follow, in function of criticality, cost, size, technology and application domain
  - Quality team reviews the decision

SOftEng http://softeng.polito.it

## Process conformance

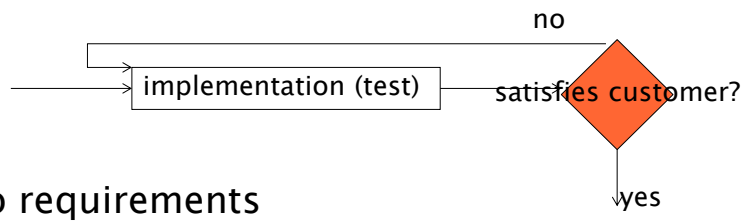
- Consistency between
  - Actual process followed in a project
  - Process model defined in documents

SOftEng
http://softeng.polito.it

## Process Models

SOftEng
http://softeng.polito.it

# Build and fix




- No requirements
- No design
- No validation of requirements/design

- May be ok for solo programming
- Does not scale up for larger projects

SOftEng http://softeng.polito.it

# Models

- Main constraints in defining a process model
  - New development vs maintenance
  - Compliance to standards, laws
    - Correlates with criticality: safety critical, mission critical, no critical
  - Size of end product
    - Correlates with effort, calendar duration, size of staff involved, number and geographical distribution of teams

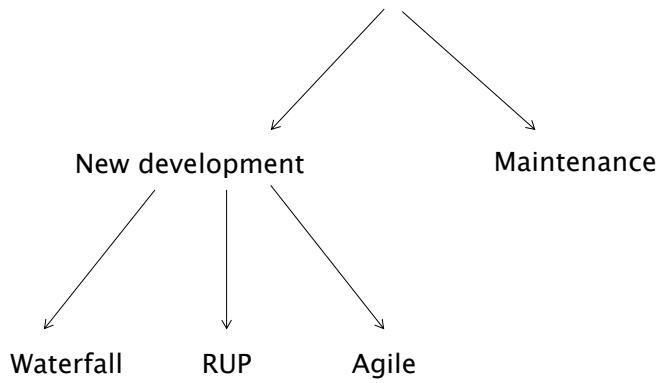
SOftEng http://softeng.polito.it

# Models

- Main factors in choosing a process model
  - New development vs maintenance
  - Sequential vs parallel activities
  - Iterations (a long one vs many short ones)
  - Emphasis on documents (yes vs no)

SOftEng
http://softeng.polito.it

# Models




SOftEng
http://softeng.polito.it

# Models, new development

- Waterfall and variants
  - Waterfall, waterfall + prototype
  - Incremental
- RUP
- Agile

- Reuse
  - as modifier in all models

SOftEng http://softeng.polito.it

# Waterfall

| | Waterfall |
|---|---|
| Sequential parallel activities | Sequential |
| Iterations | One, long |
| Emphasis on documents | yes |

SOftEng http://softeng.polito.it

# Waterfall

- [Royce1970]
- Sequential activities
  - Activity produces document/deliverable
  - Next activity starts when previous is over and freezes the deliverable
- Document driven

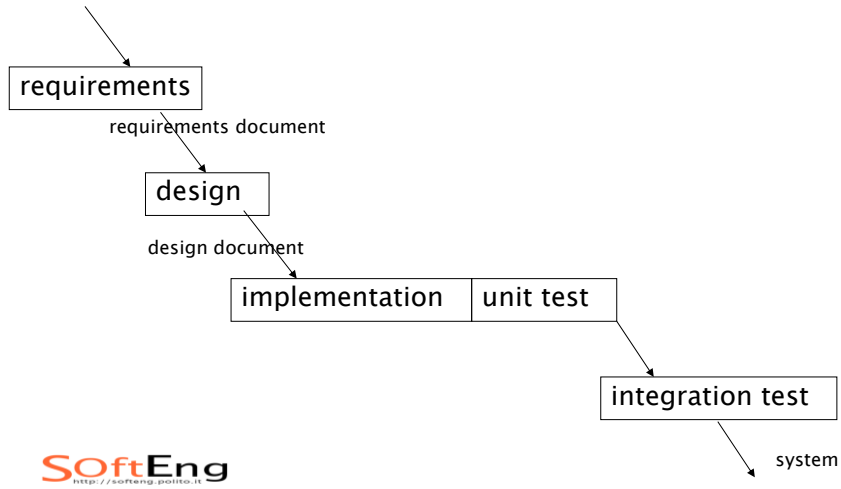
SOftEng http://softeng.polito.it

# Waterfall

- (One) long iteration
  - Ideal, never feasible in practice
  - Change to a document implies re-doing all activities depending on it
    - Change to requirement → redo all activities
    - Change to design → redo all except requirements
    - Etc
  - Changes are long and expensive

SOftEng http://softeng.polito.it

# Waterfall




SOftEng
http://softeng.polito.it

# Problems

- Lack of flexibility
  - Rigid sequentiality
  - Change to a document has heavy impact
    - tension to avoid changes
    - worst impact changes are to requirements and design
      - that normally are the most faulty
      - that should change to follow changes in context/customer needs
- Burocratic

SOftEng
http://softeng.polito.it

# Suitability

| | Waterfall |
|---|---|
| New development / maintenance | Mostly new developments |
| Compliance | Yes, documents |
| Size | Large projects, distributed teams, distributed contractors |

SOftEng
http://softeng.polito.it

# Variants of waterfall

- V Model
  - ISO 26262, IEC 61508
- Waterfall + prototype
- Incremental

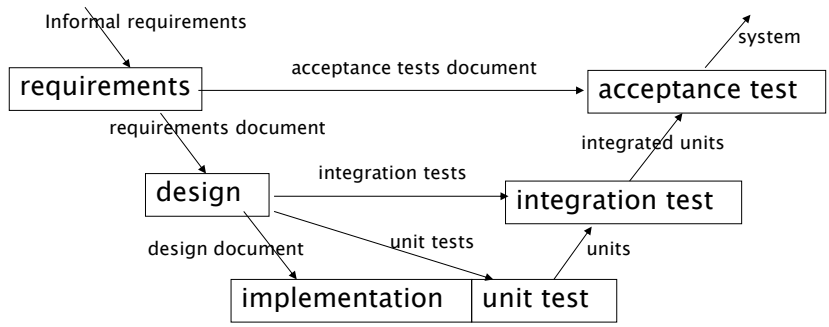
SOftEng
http://softeng.polito.it

# V Model

- Similar to waterfall
- Emphasis on VV activities
- Acceptance tests written after/with requirements
- Unit/integration tests written after/during design

SOftEng
http://softeng.polito.it

# V Model




SOftEng
http://softeng.polito.it

## ISO 26262, IEC 61508

- As examples of system process model, waterfall / V model

SOftEng http://softeng.polito.it

## ISO 26262

- Road Vehicles, functional safety
- Adaptation of IEC 61508
  - Functional safety for systems with EEPs (computers)
  - SIL – Safety Integrity Level
  - ASIL – automotive SIL, A (light red) B C D (red)

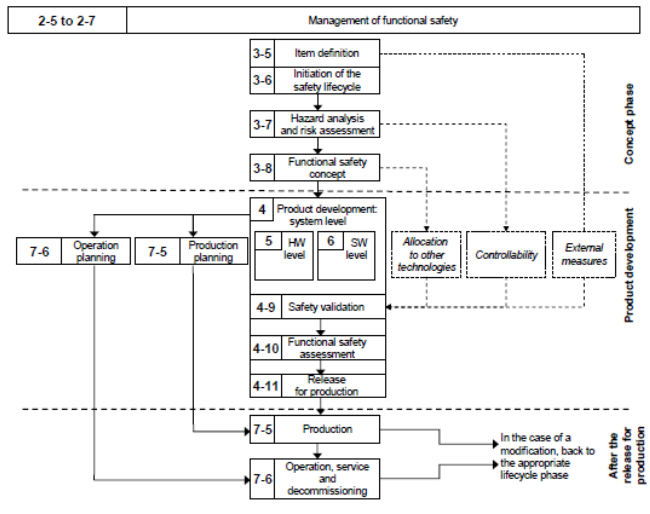
SOftEng http://softeng.polito.it

# Structure

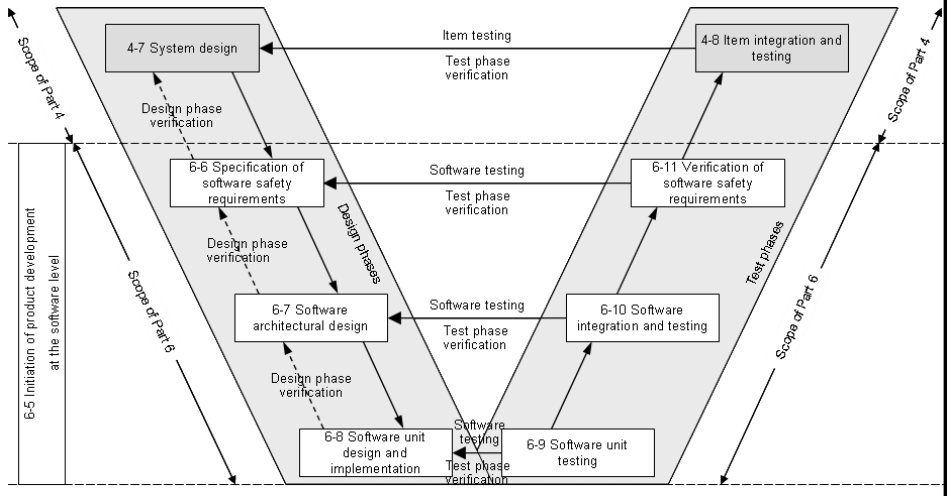
- Vocabulary
- Management of functional safety
- Concept phase
- Product development at the system level
- Product development at the hardware level
- Product development at the software level
- Production and operation
- Supporting processes
- Automotive safety integrity levels

SOftEng http://softeng.polito.it

# Safety Lifecycle



# Software lifecycle



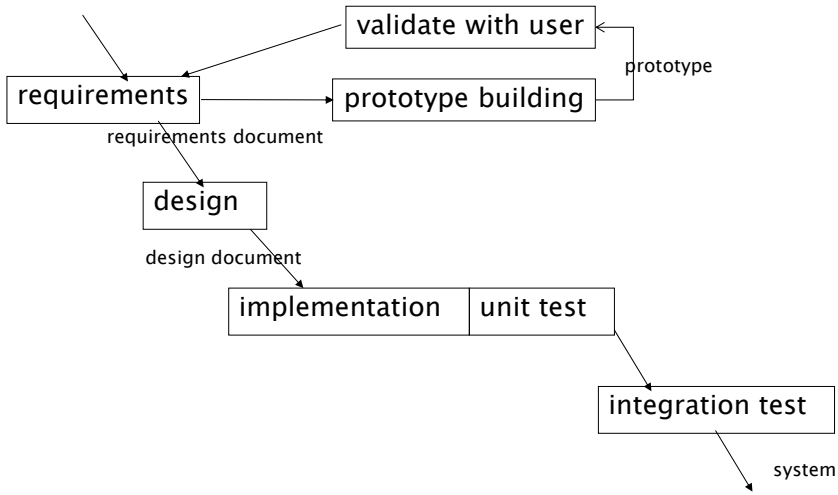

SOftEng
http://softeng.polito.it

# Prototyping + waterfall

- Quick and dirty prototype to validate/analyze requirements
- Then same as waterfall

SOftEng
http://softeng.polito.it

# Prototyping + waterfall




SOftEng
http://softeng.polito.it

# Prototyping

- Advantages
  - Clarify requirements
- Problems
  - Requires specific skills to build prototype (prototyping language)
  - Business pressures to keep prototype (when successful) as final deliverable

SOftEng
http://softeng.polito.it

## Prototype in sw

- Less functions
- Other platform
  - Final product in C, embedded
  - Prototype in Java, PC
  - Or
  - Prototype in Matlab

SOftEng http://softeng.polito.it

## Prototyping

- The idea can be applied to other parts of a project
  - GUI prototyping (see requirements chapter)
  - Design prototyping
  - Performance

SOftEng http://softeng.polito.it

## Incremental

- In Waterfall integration activity produces the complete system, that is delivered to customer in one shot
- Incremental is same as waterfall, but integration is split, every loop in integration produces a part of the system (the system is delivered in several increments)
  - One increment == 'build'

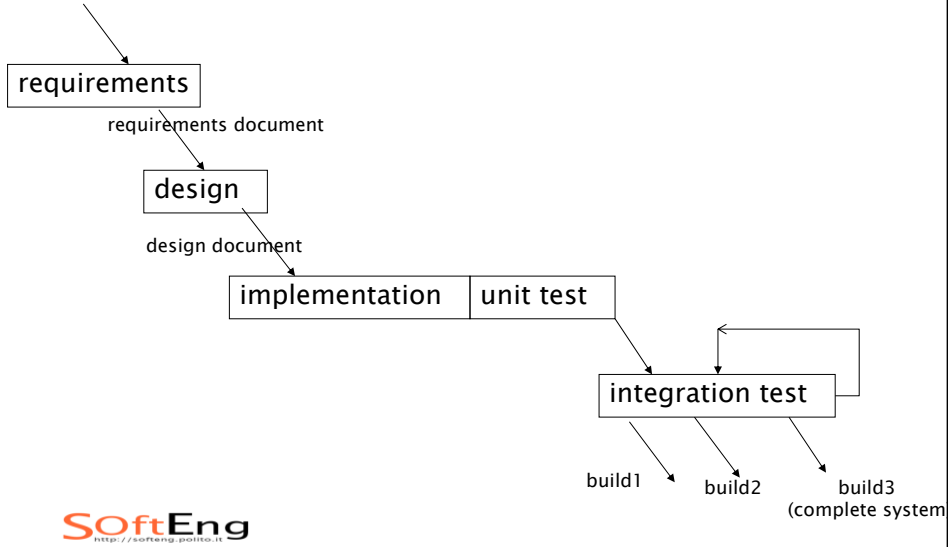
SOftEng http://softeng.polito.it

## Incremental

- Pros
  - Earlier feedback from user / customer
  - Increments that depend on external components can be delayed

SOftEng http://softeng.polito.it

# Incremental

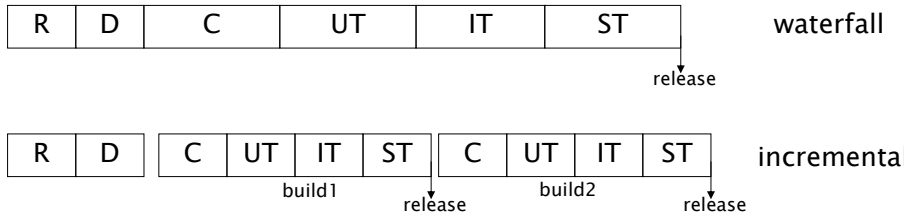



SOftEng
http://softeng.polito.it

- Requirements: R1, R2, R3
- Architecture: C1, C2, C3, C4
- Planning: 3 iterations
- Iteration1
  - R1, requires C1, C2
  - Develop and integrate C1, C2,
  - Deliver R1
- Iteration2
  - R2, requires C1, C3
  - Develop C3, integrate C1, C2, C3
  - Deliver R1 + R2
- Iteration3
  - R3, requires C3, C4
  - Develop C4, integrate C1, C2, C3, C4
  - Deliver R1 + R2 + R3

SOftEng
http://softeng.polito.it

# Comparison



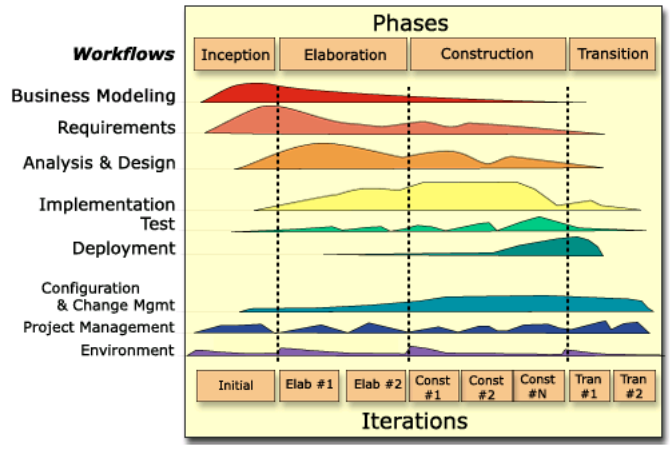

Legenda:R= Requirements, D= Design, C=Coding, UT=Unit Test,
IT= Integration Test, ST = System Test

# RUP

| | RUP |
|---|---|
| Sequential /parallel activities | parallel |
| Iterations | One, long with sub iterations |
| Emphasis on documents | partial |

SOftEng
http://softeng.polito.it

# (R)UP




SOftEng http://softeng.polito.it

# (Rational) Unified Process

- Proposed in 1999 by
  - Grady Booch
  - Ivar Jacobson
  - James Rumbaugh
- Characteristics
  - Based on architecture
  - Iterative incremental

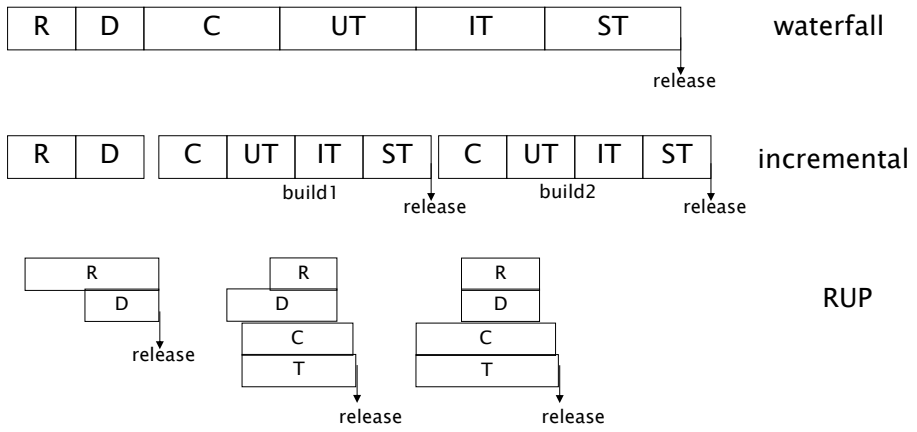
SOftEng http://softeng.polito.it

# UP Phases

- Inception
  - Feasibility study; risk analysis; essential requirements; prototyping (not mandatory)
- Elaboration
  - Requirement analysis; risk analysis; architecture definition; project plan
- Construction
  - analysis, design, implementation, testing
- Transition
  - Beta testing, performance tuning; documentation; training, user manuals; packaging for shipment

SOftEng

# Comparison




SOftEng

Legenda:R= Requirements, D= Design, C=Coding, UT=Unit Test, IT= Integration Test, ST = System Test

# Suitability

| | Waterfall |
|---|---|
| New development / maintenance | Mostly new developments |
| Compliance | partial |
| Size | Also large projects, distributed teams |

SOftEng
http://softeng.polito.it

# Agile

- Scrum
- XP
  - See other chapter

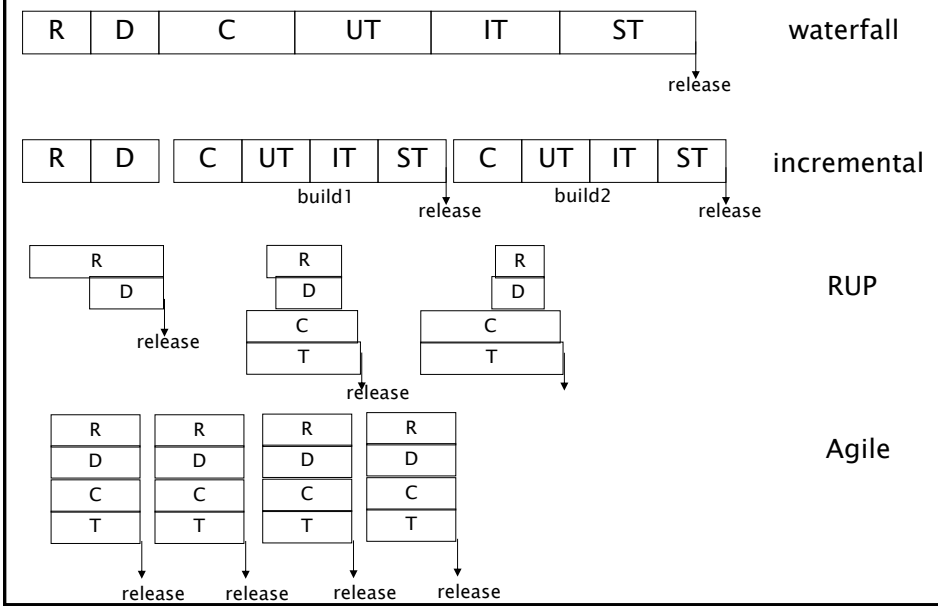
SOftEng
http://softeng.polito.it

# Agile

- Very skinny requirements and design
  - Requirements can change at every iteration
- Code and automated test cases
- Continuous integration
  - Every day, from day 0
- Iterations: 4 weeks

SOftEng
http://softeng.polito.it

# Comparison



# Agile

| | Agile |
|---|---|
| Sequential parallel activities | parallel |
| Iterations | Many, short |
| Emphasis on documents | no |

SOftEng

# Suitability

| | Agile |
|---|---|
| New development / maintenance | both |
| Compliance | no |
| Size | Small projects, co located teams |

SOftEng

# Options

| | Waterfall | RUP, S&S | Agile |
|---|---|---|---|
| Sequential parallel activities | sequential | parallel | parallel |
| Iterations | One, long | One, long with sub iterations | Many, short |
| Emphasis on documents | heavy | mild | no |

SOftEng http://softeng.polito.it

# Suitability

| | Waterfall | RUP | Agile |
|---|---|---|---|
| New development / maintenance | Mostly new developments | Mostly new developments | both |
| Compliance | yes | partial | no |
| Size | Large projects, distributed teams and contractors | Also large projects, distributed teams | Small projects, colocated teams |

SOftEng http://softeng.polito.it

## Reuse

- Most projects reuse components
  - open/closed source
  - free/not free
- Pros
  - Immediate availability
  - Often higher quality and lower cost
- Cons
  - Ownership of source
  - Trade offs needed
  - Loss of control on evolution

SOftEng http://softeng.polito.it

## Reuse – process

- Requirements
  - Must consider what is (un)available from components, and change requirements accordingly
- Design
  - Must evaluate and select components
  - Must consider constraints / issues from components and adapt design

SOftEng http://softeng.polito.it

# Reuse – ex tradeoff in req

- R1, in an accounting package invoices should be produced in pdf, png, jpg

| | No reuse | Component 1 | Component 2 |
|---|---|---|---|
| R1 | Pdf, png, jpg | Yes pdf, jpg<br>No png | Yes pdf, png<br>No jpg |
| Cost | 50 | 10 | 12 |
| Time | 3 months | 1 month | 1 month |

SOftEng

# Reuse – Activities

- Search and analysis of components
- Adaptation of requirements
  - Trade off between requirements and available components
- Design
- Implementation
  - 'Glue' software to integrate components

SOftEng

## Maintenance process

SOftEng
http://softeng.polito.it

## Change

- The key element in a maintenance process
- Can be
  - A defect to be fixed (corrective maintenance)
  - A modification to an existing function / characteristic (perfective maintenance)
  - A new function / characteristic (evolutive maintenance)

SOftEng
http://softeng.polito.it

## Change

- Can be originated by end users or developers
- Are received and processed by a group of maintainers

SOftEng http://softeng.polito.it

## Product evolution

- The product evolves by applying a flow of changes
  - Tipically with regular releases
    - Major: every few months
    - Minor/critical: when needed
- Stable baseline
  - Released regularly
- Working baseline
  - Where the maintainers work

SOftEng http://softeng.polito.it

## Process

- Receive change request (CR)
- Filter
  - merge same/similar CR
  - discard unfeasible, incorrect CR
- Assess
  - Evaluate impact of CR (effort/cost, feasibility, impact on architecture and/or functionality)
  - Rank CR (using severity – for corrective CR, and importance – for evolutive CR)

SOftEng http://softeng.polito.it

## Process (2)

- Assign CR to a (team of) maintainer, picking from ranked CRs
- Implement CR
  - Design, code, unit test, integration test
    - Maintainer
  - System test
    - Quality group
  - Insert in next release

SOftEng http://softeng.polito.it

## Issues

- The flow of changes can radically change a product over time (years)
  - Need to control / mantain architecture (architecture erosion)
  - Need to control suitability for market /users
- Only a part of CRs (even corrective) are implemented

SOftEng http://softeng.polito.it

## Variants

- A project manager assigns tasks (CRs) to mantainers – vs – maintainer picks CR from list
- A board (including product architect, market analyst, quality responsible) ranks CRs – vs – project manager does

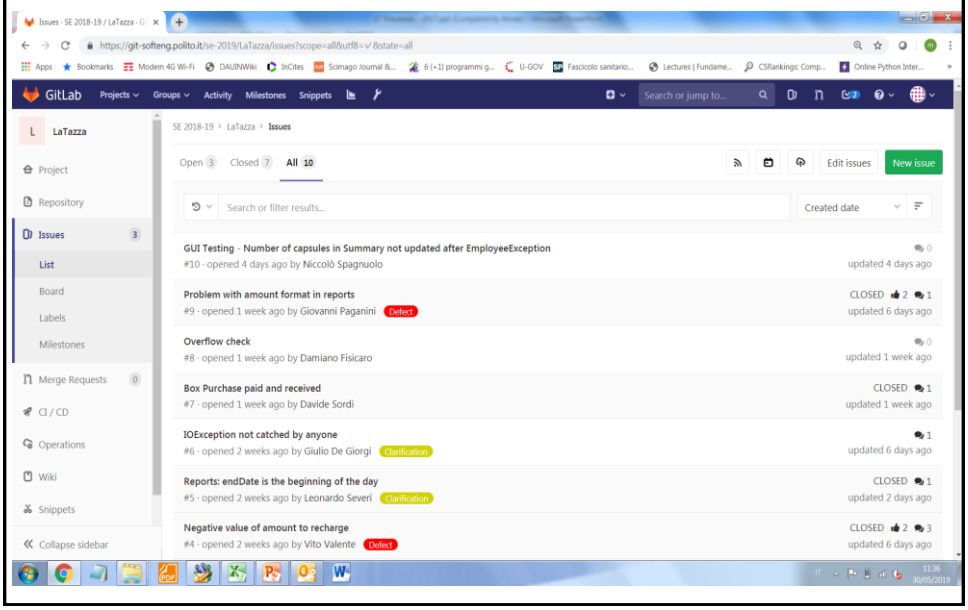
SOftEng http://softeng.polito.it

# Tools to support change process

- Jira
- Redmine
- Pivotal tracker
- Zenhub
- Github – issues
- Bugzilla

SOftEng
http://softeng.polito.it

# Git – Issues



# Ex Trac

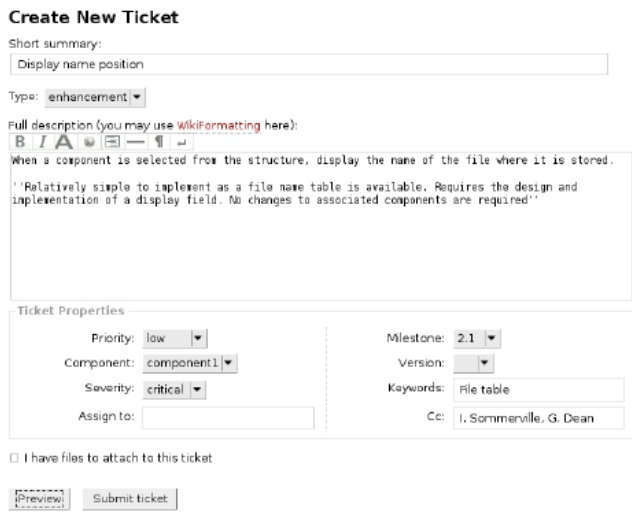
See Trac demo

- For trac: change → ticket
- Usr demo pwd demo

- http://www.hosted-projects.com/trac/TracDemo/Demo

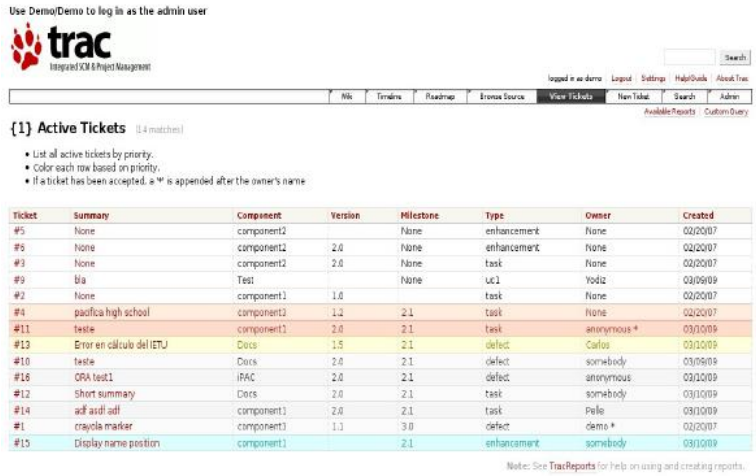
SOftEng

# Trac – create ticket



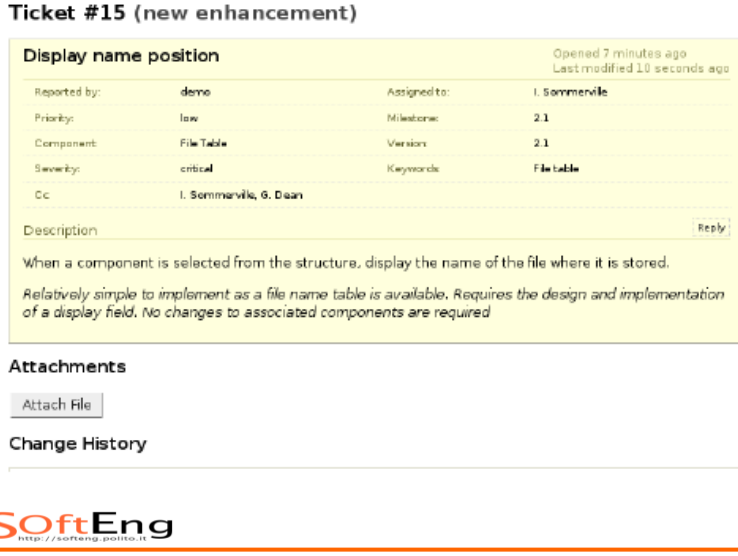

SOftEng

# Trac – see all (active) tickets



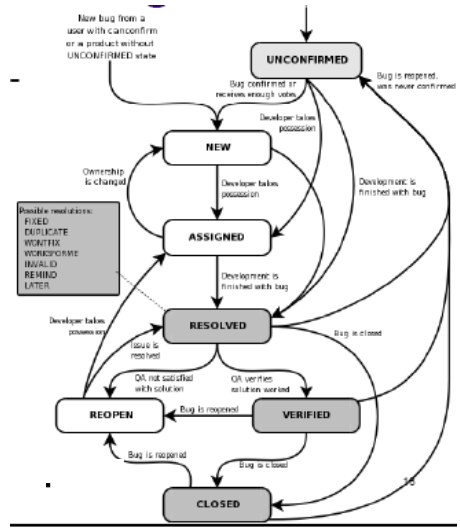

SOftEng
http://softeng.polito.it

# Trac – open ticket




SOftEng
http://softeng.polito.it

# Lifecycle for change (bug)




SOftEng
http://softeng.polito.it

# Real cases

SOftEng
http://softeng.polito.it

# Ferrari

SOftEng
http://softeng.polito.it

# Products

- Electronic Control Unit ECU (embedded sw)
  - power unit: engine, energy recovery (KERS)
  - gearbox
  - brakes
- Support tools
  - simulation (of car)
  - telemetry
  - configuration (30K parameters)
  - pit stop control

SOftEng
http://softeng.polito.it

## Roles

- Mechanical engineer
- Software engineer
  - Application (control theory)
  - Embedded
  - PC

SOftEng http://softeng.polito.it

## Process and toolchain

- Simulink/stateflow model
- Translation to C
- Compilation to assembly (freescale)
- Configuration and Tuning
- Upload to firmware

SOftEng http://softeng.polito.it

## Process and Timing

«The race starts at 2pm»

2 weeks between races

Requirements: 2 days

Coding : 2 days

Test : 4 days

FIA freezing: 3 days

Race: 3 days

SOftEng http://softeng.polito.it

## In one season (mar – nov)

- 300 versions embedded code
- 100 versions tools
- 1000 change requests
  - Few trivial bugs
  - Tens of conceptual defects (requirement misunderstanding)

SOftEng http://softeng.polito.it

## Key issues

- Tight schedule
- Interaction mechanical engineers- sw engineers
  - Understanding mechanical / control issues
  - Communicating

SOftEng http://softeng.polito.it

## Apache

(and Linux, Mozilla, ..)

SOftEng http://softeng.polito.it

## Process

- Organized as an evolution/maintenance process
- Time span: many years (10+)
- very successful (at least in successful projects)

SOftEng http://softeng.polito.it

## Tools and products

- Tools
  - CVS (config management system)
  - Mailing list
  - Bugzilla (Bug tracking)
- Products
  - Source code
  - Test cases
  - (mails, bugs, comments)

SOftEng http://softeng.polito.it

## Roles and activities

- [Mockus 2000]
  - Core team (2-8 people)
    - Architecture, requirements, integration/build, release
  - Patch developers (10-100)
    - Patch (evolutive + corrective)
  - Bug providers (100 - 1000+)
    - Signal bugs
  - Others
    - Download and use

SOftEng http://softeng.polito.it

## Facts

- Limited documentation
  - no project plan, no quality plan, no requirement document
- Key points
  - Strict CVS
  - Simple effective tools for bug/change tracking
  - Hierarchy of roles and responsibilities
  - Top notch, motivated developers (especially in core team)

SOftEng http://softeng.polito.it

## Lucent

SOftEng
http://softeng.polito.it

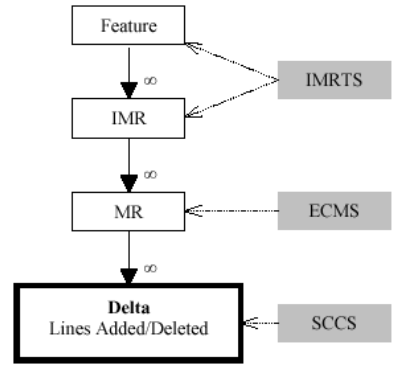
## Evolution – 5ESS change process

- 5ESS telephone switching system, Lucent
- 100MLoc, C, C++
- 50 subsystems
- 20 years
- 200 developers (peak) to 50

SOftEng
http://softeng.polito.it

# Change hierarchy

- Feature, composed of (many)
- IMR, initial modification request (logical), Implemented by (many)
- MR (functionally independent, one developer)




SOftEng

# Tools

- Features and IMRs: IMRTS
  - Description
- MRs: ECMS tool
  - Parent IMR, date, files affected, rationale for change
- Configuration management (deltas): SCCS

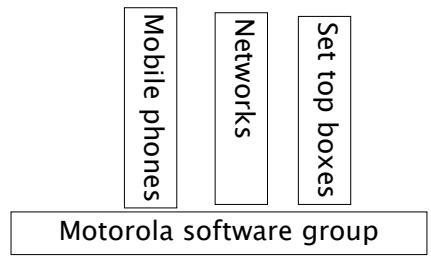
SOftEng

# Motorola GSG

SOftEng
http://softeng.polito.it

# Motorola software group

- Horizontal support (software development) to Motorola business units
- Multisite (16 worldwide)



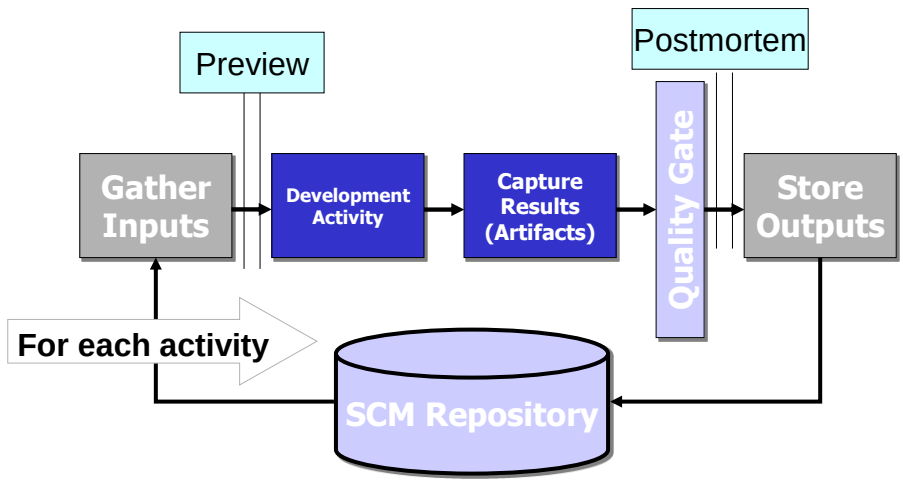

SOftEng
http://softeng.polito.it

# Principles

- CMMI framework
  - Goal: All centers level 5
- Inviolates
  - Project Planning & Tracking
  - Process Framework
  - Previews and Post Mortems
  - Records and Metrics
  - Quality Control (Review & Test)
  - Configuration Management

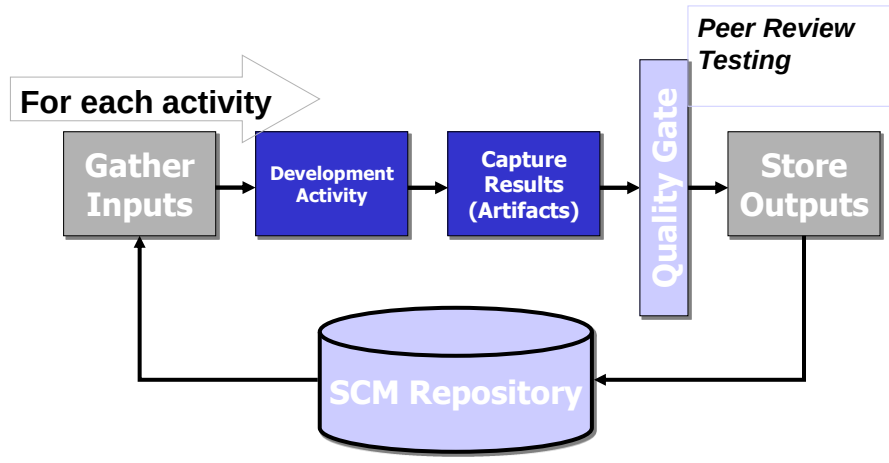
SOftEng

# Inviolate: Process framework



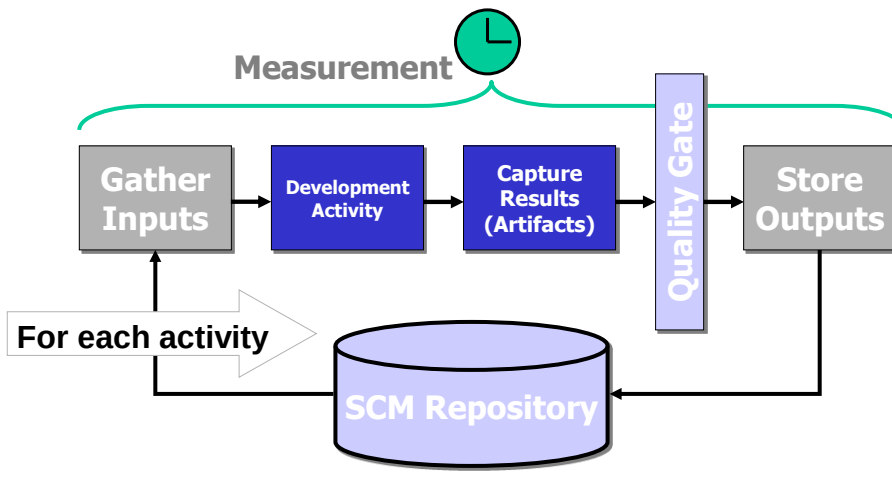

SOftEng

# Inviolate: Quality control




SOftEng

# Inviolate: record and metrics




SOftEng

## Synch and Stabilize

- Microsoft, 93-95



## Context

- Time to market essential
  - Requirements cant be fixed early on
- Complex products (MLocs), several interacting components,
  - design hard to devise and freeze early on

## Approach

- Iterations
  - changes to design and requirements
- Small teams (3-8 people) working in parallel
- Maintain hacker culture
- Synchronize frequently
- 1 tester : 1 developer

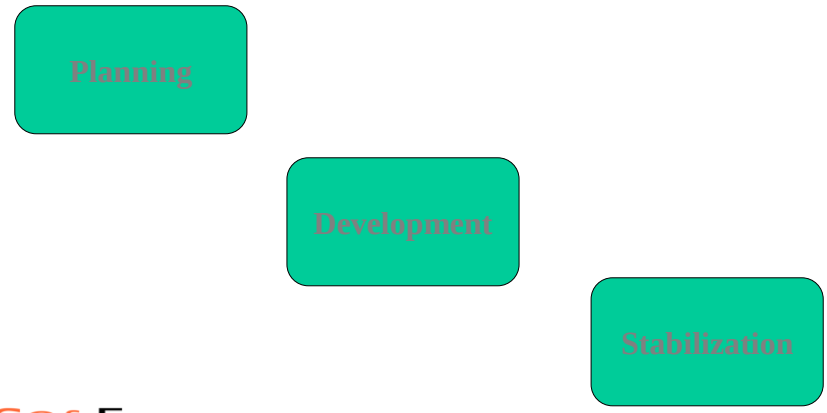
SOftEng
http://softeng.polito.it

## Synch-and-Stabilize

- Continually synchronize parallel teams
- Periodically stabilize the product in increments versus once at the end

SOftEng
http://softeng.polito.it

# 3 Phases




SOftEng
http://softeng.polito.it

# Planning Phase

Planning

- Vision Statement – Product Managers
  - Define goals for the new product
  - Priority-order user activities that need to be supported by product features
- Deliverables:
  - Specification document
  - Schedule and “feature team” formation
    - 1 program manager
    - 3-8 developers
    - 3-8 testers (1:1 ratio with developers)

SOftEng
http://softeng.polito.it

## Development Phase

Development

- Plan 3-4 subprojects (lasting 2-4 months each) (iterations)
- Buffer time between iterations
- Each subproject many feature teams
- Subprojects -- design, code, debug
  - starting with most critical features and shared components
  - feature set may change by 30% or more

SOftEng http://softeng.polito.it

## Subproject Development

- Feature teams go through the complete cycle of development, feature integration, testing and fixing problems
- Testers are paired with developers
- Feature teams synchronize work by building the product, finding and fixing errors on a daily and weekly basis
- At the end of a subproject, the product is stabilized

SOftEng http://softeng.polito.it

## Stabilization Phase

Stabilization

- Internal testing of complete product
- External testing
  - beta sites
  - ISVs
  - OEMs
  - end users
- Release preparation

SOftEng http://softeng.polito.it

## Five Principles

1. Divide large projects into multiple cycles with buffer time (20-50%)
2. Use a "vision statement" and outline feature specifications to guide projects
3. Base feature selection and priority order on user activities and data
4. Evolve a modular and horizontal design architecture
5. Control by individual commitments to small tasks and fixed project resources

SOftEng http://softeng.polito.it

## + Five Principles

1. Work in parallel teams but "synch up" and debug daily
2. Always have a product you can ship, with versions for every major platform and market
3. Speak a "common language" on a single development site
4. Continuously test the product as you build it
5. Use metric data to determine milestone completion and product release

SOftEng http://softeng.polito.it

## Rules for Coordination

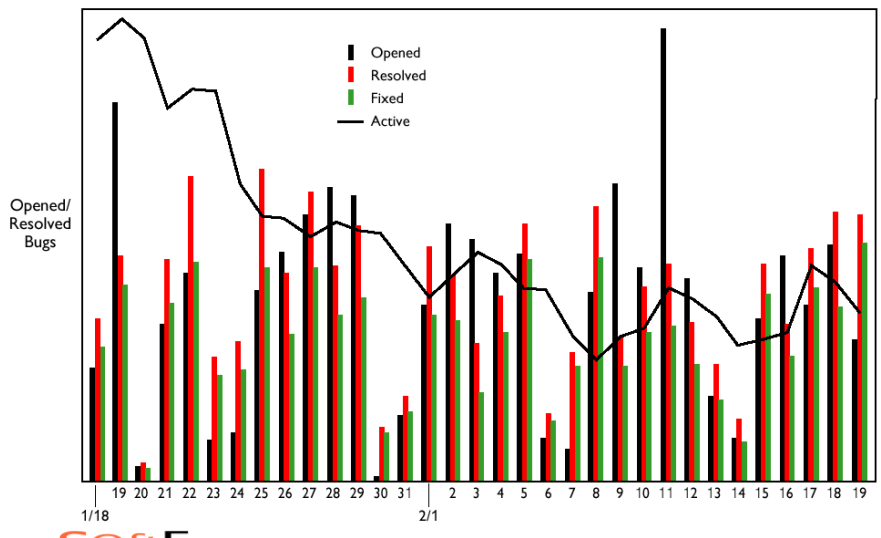
- Specific time to "check in" code
  - new build every day
  - immediate testing and debugging
- Code that "breaks" the build must be fixed immediately
- Daily builds generated for each platform and for each market

SOftEng http://softeng.polito.it

# Communication

- All developers at a single physical site
- Common development languages (C/C++)
- Standardized development tools
- Small set of quantitative metrics to decide when to move forward in a project
  - e.g. daily build progress is tracked by the number of new, resolved and active bugs

SOftEng http://softeng.polito.it




SOftEng http://softeng.polito.it

## The "Structured Hacker" Approach

- Retain hacker culture
- Add enough structure to make products reliable enough, powerful enough, and simple enough
- Support a competitive strategy of introducing products that are "good enough", and improving them by incrementally evolving features

SOftEng http://softeng.polito.it

## Comparison to Traditional

- Waterfall approach "freezes" product specification, then all components are designed, built and merged.
- Prevents
  - changing specifications
  - getting feedback from customers
  - continually testing components as the product evolves

SOftEng http://softeng.polito.it

## Advantages

- Lends itself to
  - shipping preliminary versions
  - adding features or in subsequent releases
  - easier integration of pieces of products that may still be years away

SOftEng http://softeng.polito.it

## Synch-&-Stabilize vs. Sequential

- Parallel development & testing
- Vision statement and evolving specification
- Features prioritized in subprojects
- Daily builds (synch), intermediate stabilizations

- Sequential development & testing
- Frozen specification
- All features built simultaneously
- One late, large integration and test phase at the end

SOftEng http://softeng.polito.it

## Synch-&-Stabilize vs Sequential

- Fixed, multiple release & ship dates
- Continuous customer feedback in development process
- Large teams work like small teams
- Aiming for perfection in each cycle
- Feedback after development as input for next project
- Individuals in separate functional departments work as a large group

SOftEng http://softeng.polito.it

## Weaknesses

- Microsoft needs more concentrated focus on its product architectures
- Microsoft needs a more rigorous approach to design & code reviews
- S-&-S process may not be suitable for all new products
  - e.g., Video on demand components have real-time constraints that require precise mathematical models
- Does not focus on defect prevention

SOftEng http://softeng.polito.it

## Benefits

- Breaks down large projects into manageable pieces (priority-ordered sets of features)
- Projects proceed systematically even when it's impossible to complete a stable design at the outset
- Large teams work like small teams

SOftEng http://softeng.polito.it

## More Benefits

- Provides a mechanism for customer inputs, setting priorities, completing most important parts first
- Allows responsiveness to the marketplace by "always" having a product ready to ship and having an accurate assessment of which features are completed

SOftEng http://softeng.polito.it

Cyber physical ecosystems (wikipedia, OSM, Google, Twitter..)

SOftEng http://softeng.polito.it

## The 'city' model [Kazman 2014]

- No clear boundary
  - APIs and mash ups (ex API for Google Maps)
- Peer content / code production
  - Requirements never known, emerge from individual contributions
- Continuous evolution
  - No releases, no planning

SOftEng http://softeng.polito.it

- Continuous operation
  - No development – maintenance
  - Always on and always changing
- Open teams, unstable resources
- Sufficient correctness
  - Always ‘beta’
- Emergent behaviours
  - Unplanned behaviours and functions
  - (Twitter as coordination tool of masses)

SOftEng http://softeng.polito.it

## The structure

- Core/Kernel
  - Horizontal functions
  - Always on, top reliability, closed, slow to change
- Periphery
  - End user functions and content
  - Fast changing, open

SOftEng http://softeng.polito.it

# Bifurcation

- Core
  - Core requirements / architecture well defined
  - Controlled by core team
  - Reliability: high
- Periphery
  - Unclear, instable req and design
  - No control, crowdsourcing
  - Reliability: variable

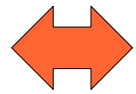
SOftEng
http://softeng.polito.it

SOftEng
http://softeng.polito.it

## Process selection

SOftEng
http://softeng.polito.it

## Process selection

Product  Process

SOftEng
http://softeng.polito.it

# Process attributes

- Sequential vs parallel activities
  - (i.e. documents can be modified in parallel or not)
- Iterations vs no iterations
  - Duration of iteration
- Time framed vs not
- Colocation of staff
- Emphasis on documents (need for certification, compliance)

SOftEng http://softeng.polito.it

# Process comparison

| | Agile | 26262 | RUP | Synch Stab |
|---|---|---|---|---|
| Iteration number | many | One+ | few | few |
| Iteration duration | weeks | long | months | months |
| Parallel activities | yes | no | yes | Yes |
| Documents | few | many | many | few |
| Colocation staff | yes | Yes or not | - | Yes |
| | | | | |

SOftEng http://softeng.polito.it

## Product attributes

- Criticality
- Size
- Domain
- Lifespan
- Lifecycle phase
- Bespoke / market driven
- Ownership
- Relationship user – developer

SOftEng http://softeng.polito.it

## Criticality

- Criticality
  - Safety critical, mission critical, other
  - More criticality, more emphasis on: reliability, safety, security
    - Norms and laws applied, legal responsibility issues
    - ex ISO IEC 61508 for safety critical functions
    - Ex Iso 26262 for safety critical functions in automotive

SOftEng http://softeng.polito.it

## Size

- Size
  - LOCs (Lines of Code)
  - Duration of development, team size
  - Number of subcontractors
  - Effect on coordination and communication during development and maintenance

SOftEng http://softeng.polito.it

## Domain

- Aerospace, medical, automotive, industry, banking, insurance, ...
  - Effect on norms and laws applied:
    - ex Basel 3 in finance, 26262 in automotive, Do178B in aerospace
  - Effect on responsibility of developer, operator, maintainer

SOftEng http://softeng.polito.it

## Lifespan, lifecycle phase

- Expected lifespan
  - Months, years
  - Effect on tradeoff development cost vs maintenance cost
- Lifecycle phase
  - New development
  - Maintenance
    - Cyclic management of change requests, effects of legacy

SOftEng http://softeng.polito.it

## Bespoke / market driven

- Bespoke: one customer / end user
- Market driven: many customers / end users
- Effect on time to market, requirement collection and ranking, cost structure and business model

SOftEng http://softeng.polito.it

## Ownership

- Full property
- Copyright
- Copyleft
  - Effect on capability of modifying code vs parameterization/adaptation

SOftEng http://softeng.polito.it

## Relationship with developer

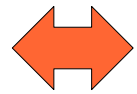
- Same as end user
- Internal department of company
- External company

SOftEng http://softeng.polito.it

# Process selection

**Product attributes**

- Criticality
- Size
- Lifetime
- Bespoke /market driven



**Process attributes**

- Parallel activities
- Iterations
- Time framed
- Colocation of staff
- Documents based

SOftEng http://softeng.polito.it

# Process selection

- Understand product attributes
- Rank –ilities
- Apply rules of thumb

SOftEng http://softeng.polito.it

## Rules of thumb

- Reliability, safety
  - Waterfall -like (document-based, few iterations, no parallel activities)
- Market driven (Time to market)
  - Frequent iterations
- Colocation of staff
  - No: document-based, no parallel activities

SOftEng http://softeng.polito.it

## Rules of thumb 2

- Size
  - The bigger, the more documents and activities
- Lifetime
  - Long: documents

SOftEng http://softeng.polito.it

## Relationship developer / user

SOftEng http://softeng.polito.it

## Typical scenario

1 Bespoke, external, property, *new*

- Company (ex Polito) needs custom software product (ex Polito APP)
- Company SW develops it
  - Inception, for requirement analysis and contract negotiation
  - Contract signature
  - Development, delivery

SOftEng http://softeng.polito.it

# Typical scenario

2 Bespoke, external, property, *maintenance*

- Company (ex Polito) needs maintenance work on owned software product P (Ex POLITO App)
- Company SW performs maintenance work
  - Similar to case 1, but typically contract is per year and involves a fixed amount of work (or dedicated staff) in the year (ex 400 p-days) and not an amount of functionality (as in scenario 1)

SOftEng http://softeng.polito.it

# Typical scenarios

1-a, 2-a

same as above, but internal

- typical of bank, insurance,
- internal IT department instead of external vendor
- No legal contract, but similar negociation of functionality / effort requested between user department and IT department

SOftEng http://softeng.polito.it

# Typical scenarios

3 COTS, external, copyright

- Company (ex Microsoft) develops and sells a mass market product (ex Windows)
  - Fixes and patches on the current release (maintenance thread)
  - Work on next release (new development thread)

SOftEng
http://softeng.polito.it