Models [5]

Prototypes [2.3.1]
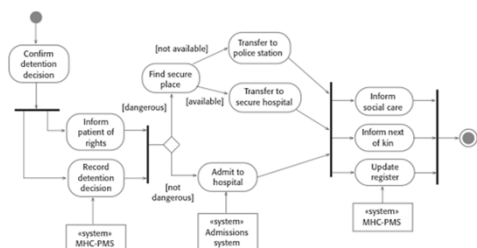
Process repetition [2.3.2, 2.4]

Agile processes [3]

Models

- Abstraction
  - remove detail
  - focus on viewpoint
- Dynamic
  - interaction with environment
  - control flow
  - data flow
- Static
  - physical
  - database schema
  - object structure

Software Engineering   47

Figure 5.2 Process model of involuntary detention

UML activity diagram

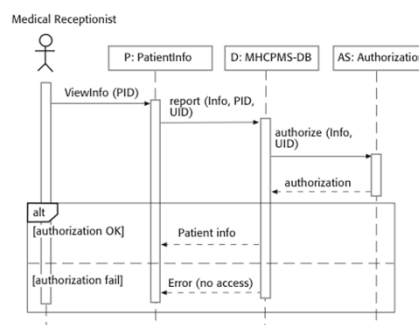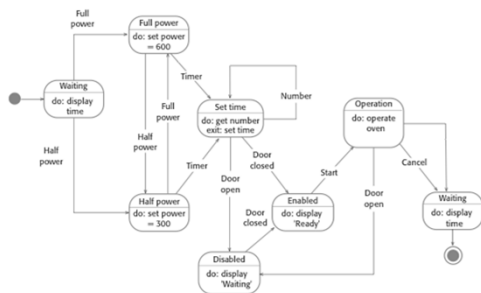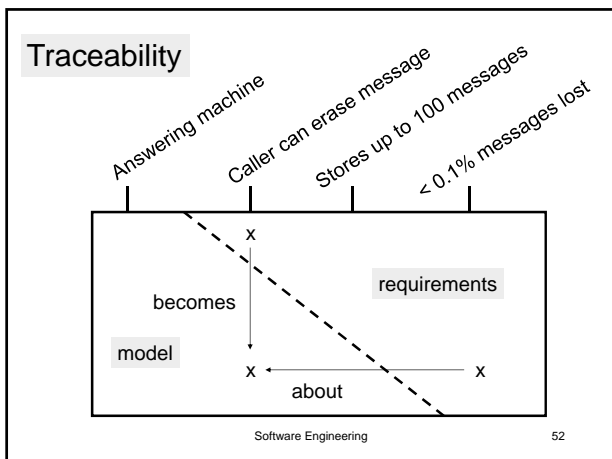Figure 5.6 Sequence diagram for View patient information

Figure 5.16 State diagram of a microwave oven

Models

- Why?
  - Improve understanding
  - all viewpoints
  - "coathanger" - requirements refer to it
  - validation
- Warnings
  - try to avoid design
  - avoid massive documentation

Software Engineering   51

Traceability

Answering machine
Caller can erase message
Stores up to 100 messages
< 0.1% messages lost

x

requirements

becomes

model

x ← x

about

Software Engineering   52

Prototypes

- Used for
  - requirements elicitation
  - requirements validation
  - proof of concept   } Risk reduction

  - user training
  - back to back testing

Software Engineering   53

Warning

- A prototype is not the product!
  It may differ in
  - functionality
  - performance
  - reliability
  - scalability
  - maintainability
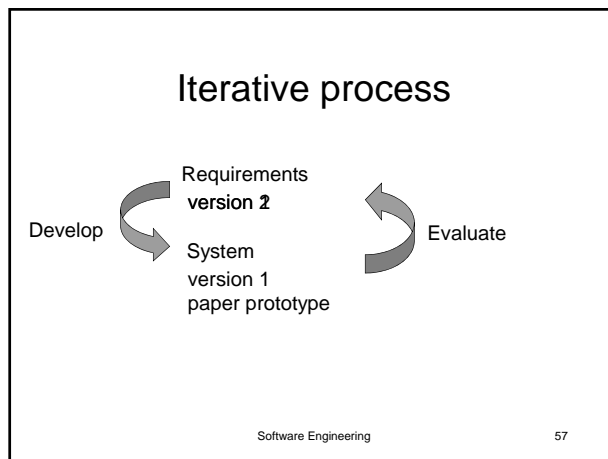 - user-interface, "finish"
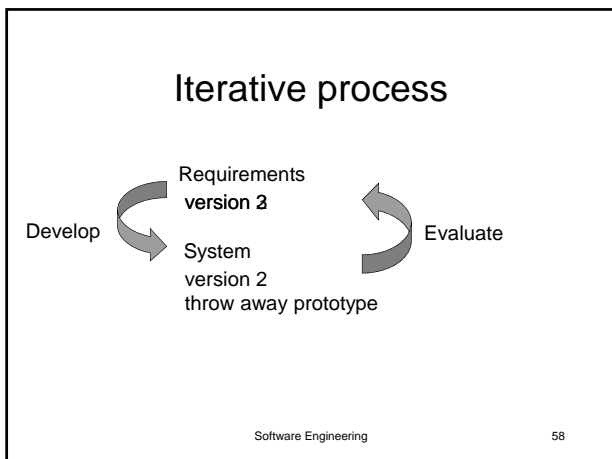
Software Engineering   54

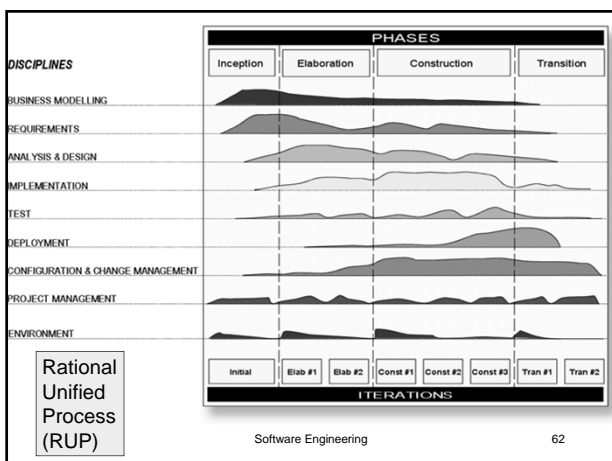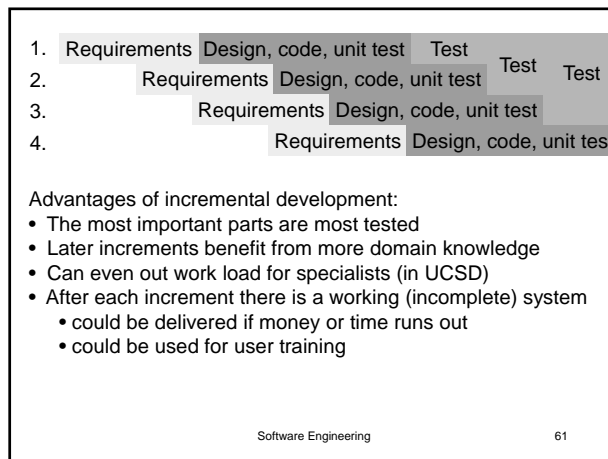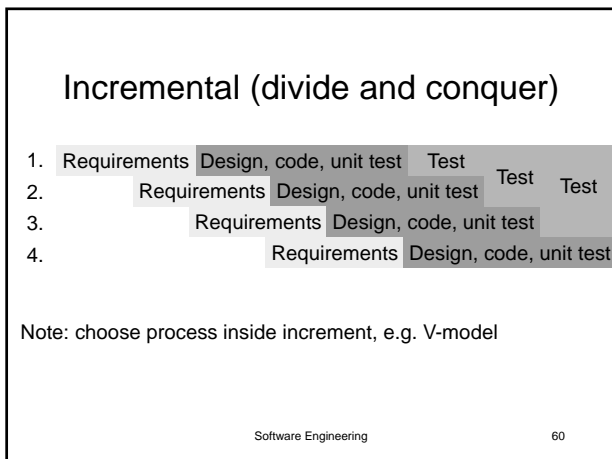|	Throw away prototype	Exploratory development
Goal	specification validation, risk	working system
Start	uncertain parts	known parts
Role	evaluate and throw away	grow into the system
Quality	as low as useful	product
Risks	- not thrown away - too low quality prevents evaluation	- bad structure - low process visibility - contractual problems

Software Engineering   55

Repetition in processes

Iterative
Incremental
Evolutionary

Iterative process

Requirements
version 2

Develop

System
version 1
paper prototype

Evaluate

Software Engineering   57

Iterative process

Requirements
version 2 ~~3~~

Develop

System
version 2
throw away prototype

Evaluate

Iterative process

- Each iteration
  - functionality can be added
  - requirements can be changed

- Problem: when does it end?
  - budget
  - contract

Incremental (divide and conquer)

1. Requirements | Design, code, unit test | Test
2. Requirements | Design, code, unit test | Test | Test
3. Requirements | Design, code, unit test
4. Requirements | Design, code, unit test

Note: choose process inside increment, e.g. V-model

1. Requirements | Design, code, unit test | Test
2. Requirements | Design, code, unit test | Test | Test
3. Requirements | Design, code, unit test
4. Requirements | Design, code, unit test

Advantages of incremental development:
- The most important parts are most tested
- Later increments benefit from more domain knowledge
- Can even out work load for specialists (in UCSD)
- After each increment there is a working (incomplete) system
  - could be delivered if money or time runs out
  - could be used for user training

Rational
Unified
Process
(RUP)

Top-down vs. Bottom-up

- Traditional "from scratch" development
- Hierarchical system

- "The Cathedral"

- Reuse based development
- "network" system component-based

- "The Bazaar"

2012-01-30

Traditional vs. Agile .

- Follow a plan
- Change costs
- Frozen requirements contract
- Documentation
- Deliverables at a deadline
  - *incremental*

- People
- Embrace change
- User stories, tests, customer involvement
- Working software
- Time-boxed
- smaller increments

Software Engineering   64

Agile principle	Scrum	Extreme Programming
Incremental planning and development	Sprints Sprint backlog Planning poker	Implement user stories Story cards Planning poker
Customer involvement	Product owner Demo at end of sprint	Customer representative in development team
People, not process	Scrum meetings Sustainable pace (time-boxed)	Pair programming Collective ownership of code Sustainable pace
Embrace change	Change occurs from one sprint to the next	Continuous integration and release Test-first development
Maintain simplicity	Refactoring No anticipation of future requirements	Refactoring No anticipation of future requirements