The Lambda Protocol for Synthesizing Trustworthy Requirements

Larry Bernstein
Stevens Institute of Technology
Castle Point, Hoboken, NJ 07030
USA

QSE Lambda Protocol

- Prospectus
- Measurable Operational Value
- Prototyping or Modeling
- sQFD
- Schedule, Staffing, Quality Estimates
- ICED-T
- Trade-off Analysis

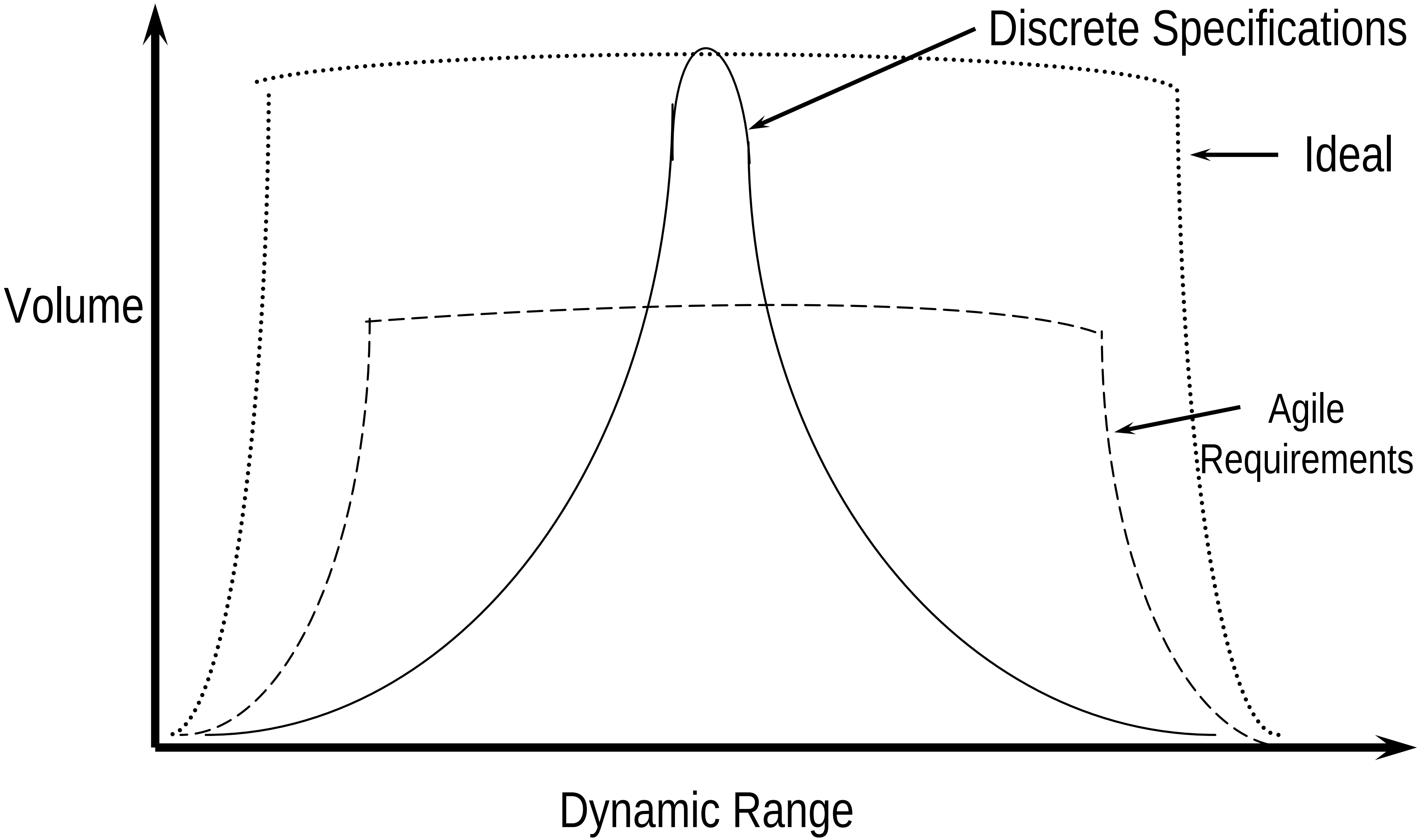
Trustworthy Software is:

- Safe: Does no harm
- Reliable: No crash or hang.
- Secure: No Hacking Possible

What is a Requirement?

- A property that must be exhibited by a system to solve some problem.
- Requirements may be
  - – Functional providing product capabilities
  - – Non-Functional constraining the implementation

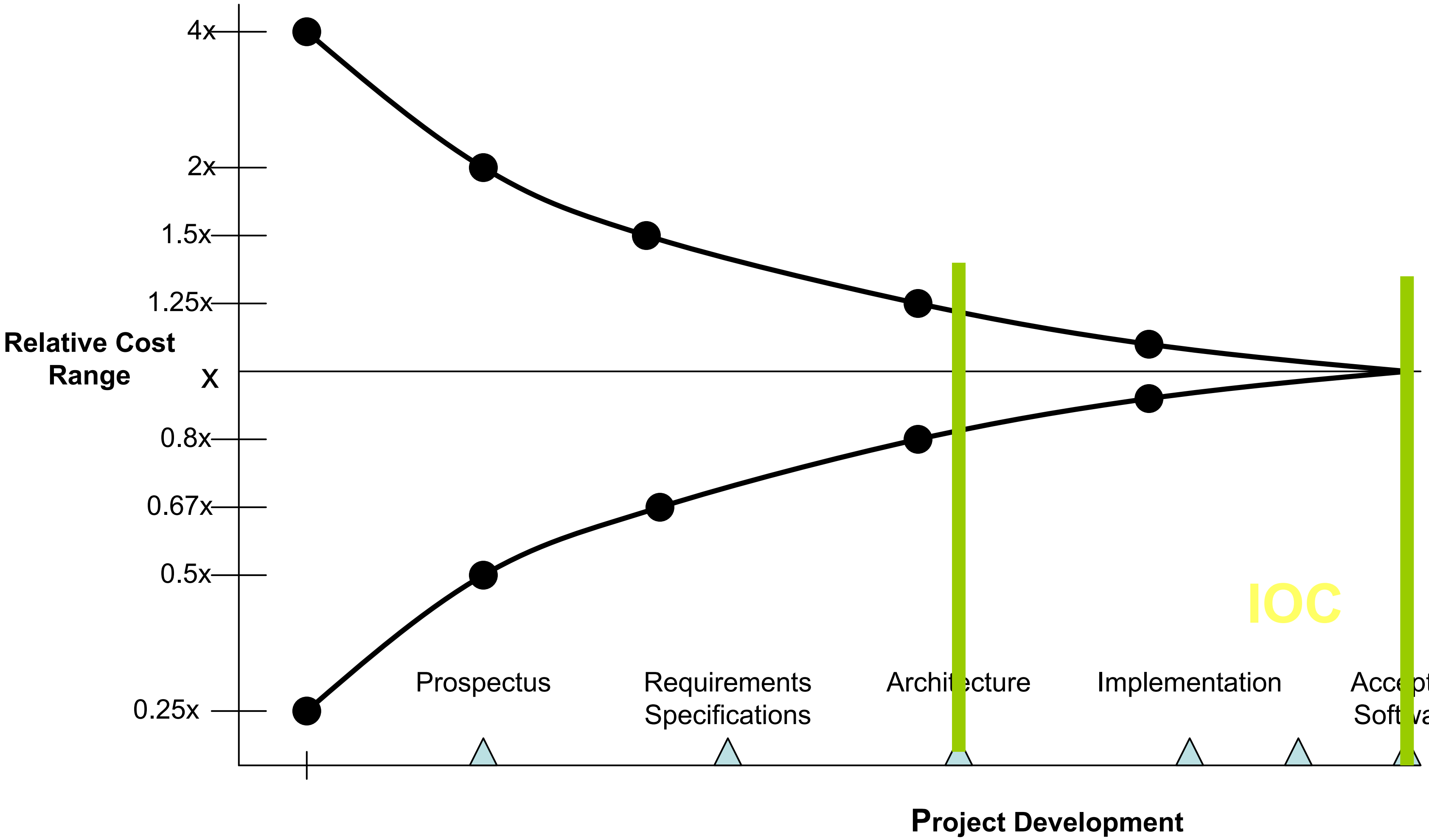
System Performance Resulting from Robust Requirements vs. Discrete Specifications



Top Ten Software Risk Items

Category	Risk Item
People	1. Personnel Shortfalls
	2. Unrealistic Schedules and Budgets
Requirements	**3. Developing the Wrong Software Functions**
	4. Developing the Wrong User Interface
	5. Gold Plating
	6. Continuing Stream of Requirements Changes
Externalities	7. Shortfalls in Externally-Furnished Component
	8. Shortfalls in Externally-Performed Tasks
Technology	9. Real-Time Performance Shortfalls
	10. Straining Computer Science Capabilities

Costs Cone of Uncertainty



QSE Characteristics

- Solving the right problem the right way
- Tested against requirements.
- Certified against problem
- Bounded execution domain
- Industrial Strength Requirements for Software Intensive Systems-of-Systems

Universal Software Engineering Equation

$$\mathit{Reliability}\ (t) = \mathrm{e}^{-k\,\lambda t}$$

when the error rate is constant and where k is a normalizing constant for your software shop and

λ = Complexity/ [effectiveness x staffing]

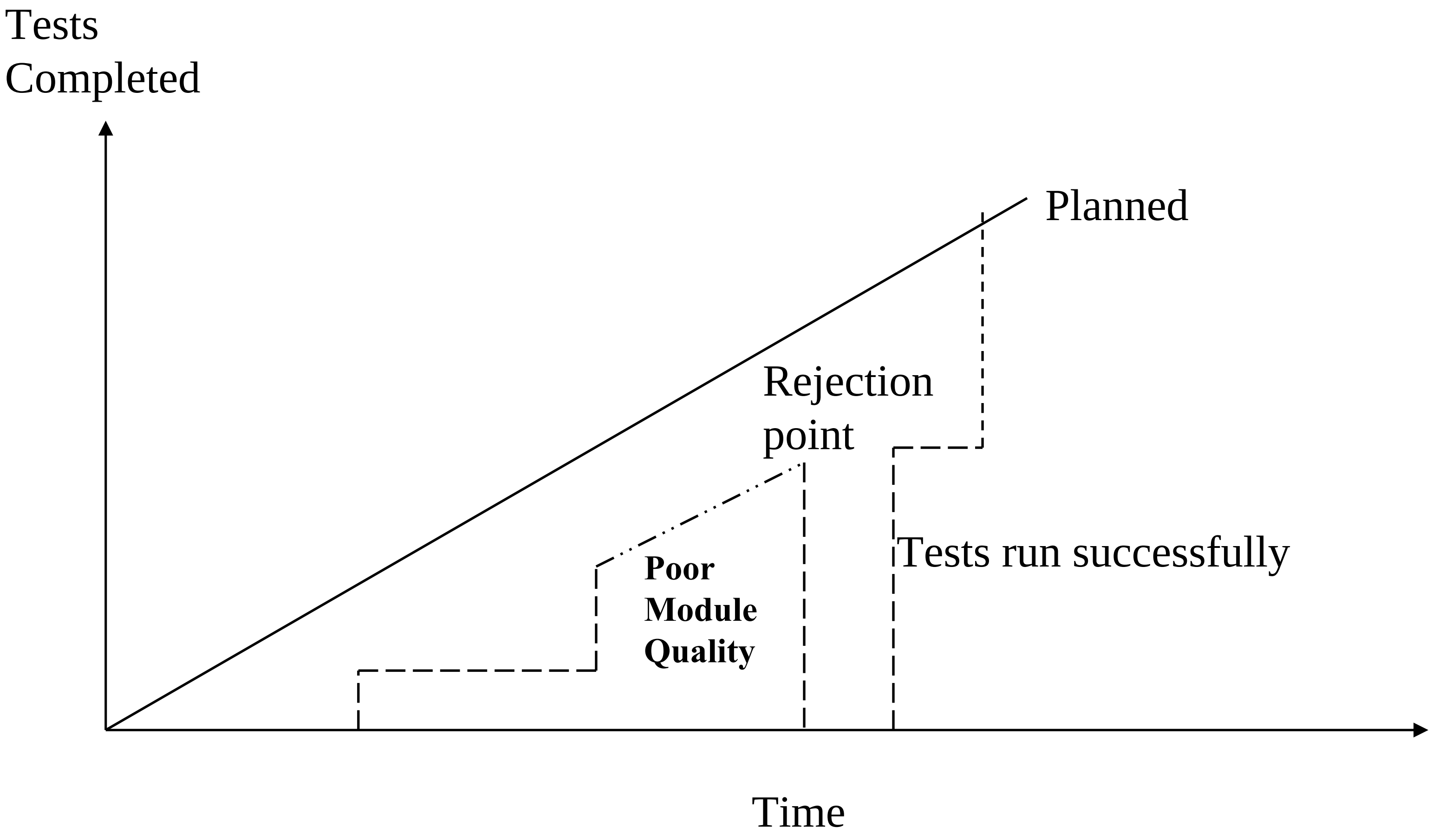
Boundary Conditions

$$\mathit{Reliability}\ (\mathbf{0}) = \mathbf{1}$$

$$\mathit{Reliability}\ (\mathbf{T}) = e^{-k\,\lambda \mathbf{T}}$$

$$\mathit{Reliability}\ (\infty) = \mathbf{0}$$

Software Testing Footprint



QSE Lambda Protocol

- Prospectus
- Measurable Operational Value
- Prototyping or Modeling
- sQFD
- Schedule, Staffing, Quality Estimates
- ICED-T
- Trade-off Analysis

Prospectus

- Description of the problem domain
- Scope of solution
- Specific project goals
- Constraints on the behavior or structure of the software:
  - For example, Trustworthiness

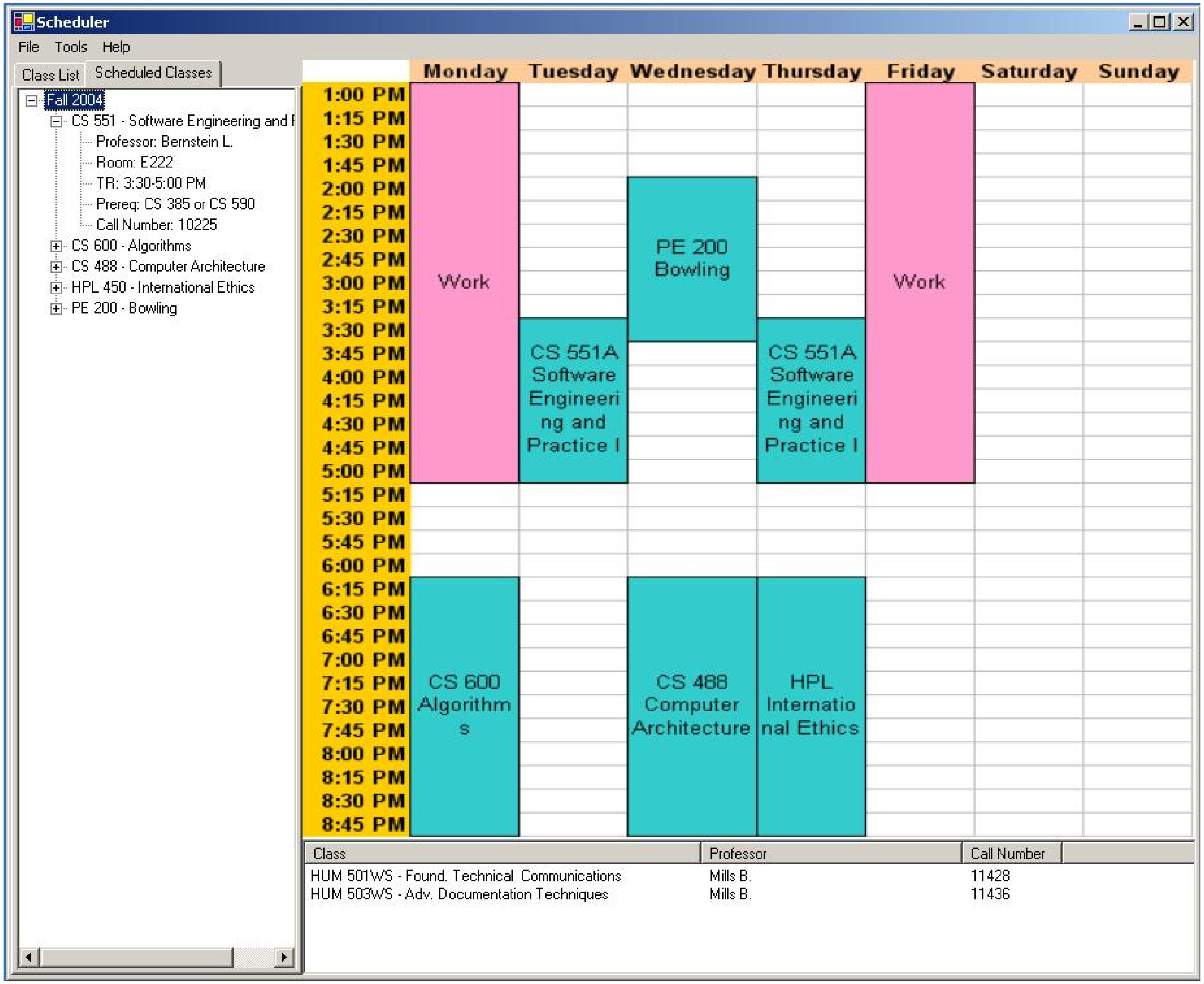
Case Study: SchedulerPro Prospectus

User friendly, efficient interface for students to create and modify class schedules.

Features:

- Visual schedule creation and editing
- Schedule suggestion
- Schedule comparison view
- Monitor closed-out sections

SchedulerPro Prototype Screen

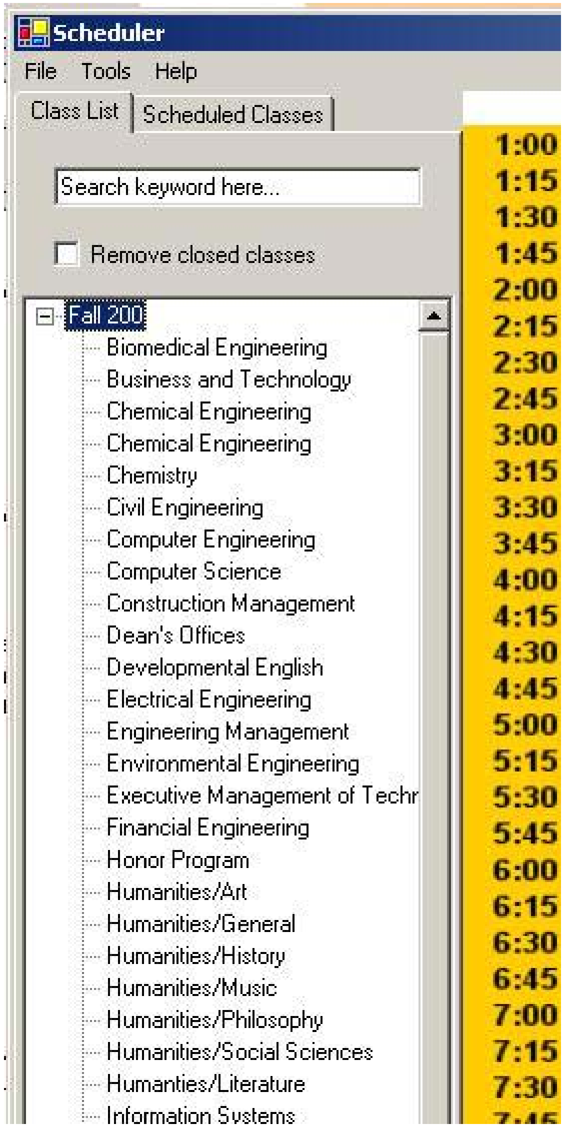


- Fall 2004
  - CS 551 - Software Engineering and
    - Professor: Bernstein L.
    - Room: E222
    - TR: 3:30-5:00 PM
    - Prereq: CS 385 or CS 590
    - Call Number: 10225
  - CS 600 - Algorithms
  - CS 488 - Computer Architecture
  - HPL 450 - International Ethics
  - PE 200 - Bowling

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
1:00 PM	Work				Work		
1:15 PM							
1:30 PM							
1:45 PM							
2:00 PM			PE 200 Bowling				
2:15 PM							
2:30 PM							
2:45 PM							
3:00 PM							
3:15 PM							
3:30 PM		CS 551A Software Engineering and Practice I		CS 551A Software Engineering and Practice I			
3:45 PM							
4:00 PM							
4:15 PM							
4:30 PM							
4:45 PM							
5:00 PM							
5:15 PM							
5:30 PM							
5:45 PM							
6:00 PM							
6:15 PM	CS 600 Algorithms		CS 488 Computer Architecture	HPL International Ethics			
6:30 PM							
6:45 PM							
7:00 PM							
7:15 PM							
7:30 PM							
7:45 PM							
8:00 PM							
8:15 PM							
8:30 PM							
8:45 PM							

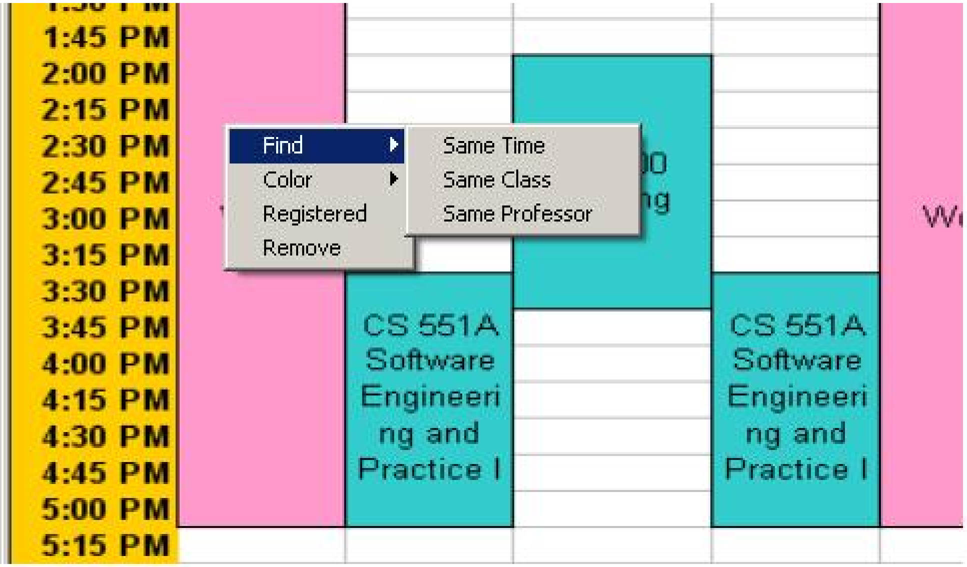
Class	Professor	Call Number
HUM 501WS - Found. Technical Communications	Mills B.	11428
HUM 503WS - Adv. Documentation Techniques	Mills B.	11436

SchedulerPro Prototype Screen



□ Remove closed classes

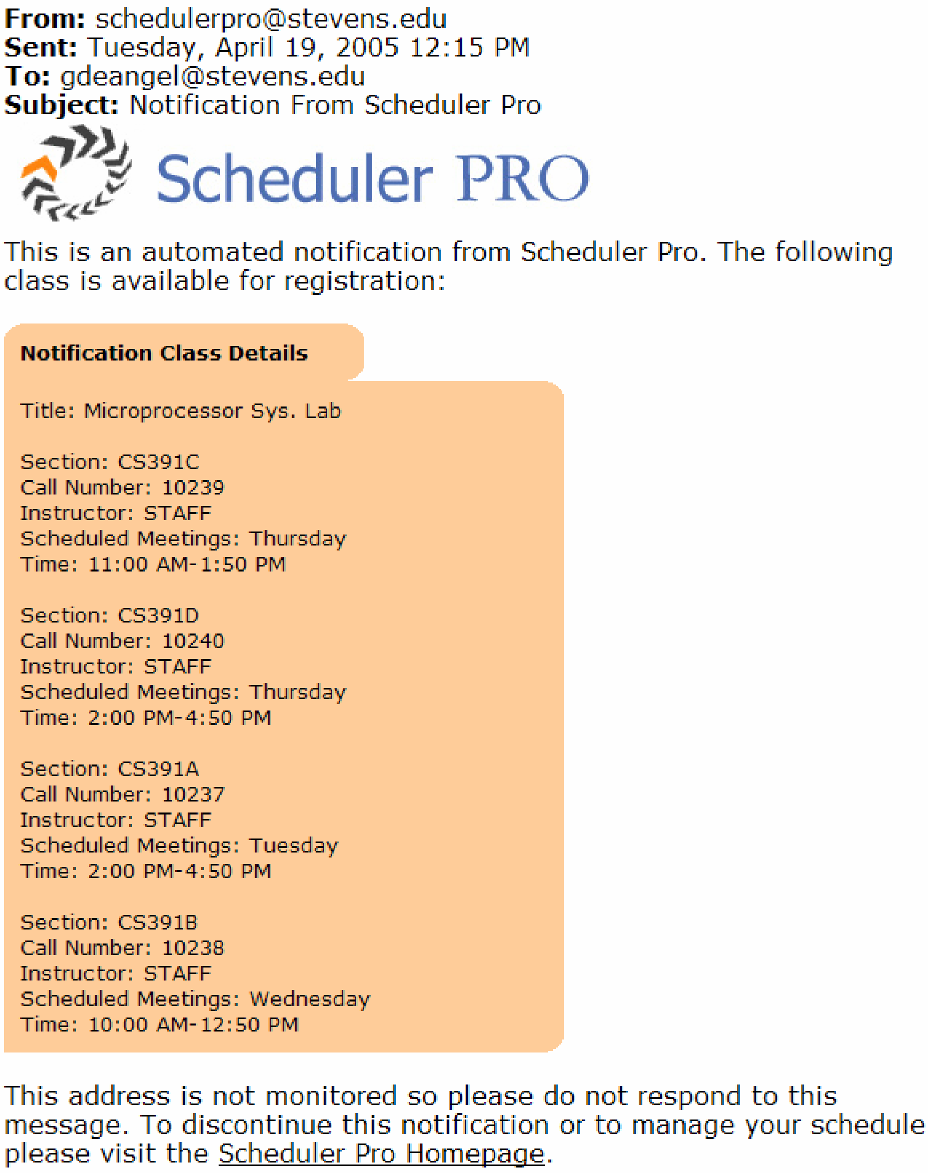
⊟ Fall 200
- Biomedical Engineering
- Business and Technology
- Chemical Engineering
- Chemical Engineering
- Chemistry
- Civil Engineering
- Computer Engineering
- Computer Science
- Construction Management
- Dean's Offices
- Developmental English
- Electrical Engineering
- Engineering Management
- Environmental Engineering
- Executive Management of Techr
- Financial Engineering
- Honor Program
- Humanities/Art
- Humanities/General
- Humanities/History
- Humanities/Music
- Humanities/Philosophy
- Humanities/Social Sciences
- Humanties/Literature
- Information Systems



CS 551A Software Engineering and Practice I

CS 551A Software Engineering and Practice I

SchedulerPro Notification Emails



From: schedulerpro@stevens.edu
Sent: Tuesday, April 19, 2005 12:15 PM
To: gdeangel@stevens.edu
Subject: Notification From Scheduler Pro

Scheduler PRO

This is an automated notification from Scheduler Pro. The following class is available for registration:

Notification Class Details

Title: Microprocessor Sys. Lab

Section: CS391C
Call Number: 10239
Instructor: STAFF
Scheduled Meetings: Thursday
Time: 11:00 AM-1:50 PM

Section: CS391D
Call Number: 10240
Instructor: STAFF
Scheduled Meetings: Thursday
Time: 2:00 PM-4:50 PM

Section: CS391A
Call Number: 10237
Instructor: STAFF
Scheduled Meetings: Tuesday
Time: 2:00 PM-4:50 PM

Section: CS391B
Call Number: 10238
Instructor: STAFF
Scheduled Meetings: Wednesday
Time: 10:00 AM-12:50 PM

This address is not monitored so please do not respond to this message. To discontinue this notification or to manage your schedule please visit the Scheduler Pro Homepage.

Measurable Operational Value SchedulerPro MOV

Reduce student withdrawals by 20%

SchedulerPro Functional Goals

Schedule Classes and Personal Time

→ Searching

→ Course Placement

→ Course Detail Viewing

→ Course Removal

→ Scheduling Personal Blocks

→ Notification (optional)

→ Course Suggestions (optional)

Student Directed Features

- Search available classes by:
  - ✓Same professor
  - ✓Similar time
  - ✓Same or equivalent class but different sections
- Register and track registrations
- Color classes and arbitrary time-blocks by user choice

SchedulerPro Nonfunctional Requirements

- Integrate with “Web for Students’ and existing authentication systems and avoid incompatibilities
- Allow schedules to be saved/accessed from a server or local file
- Provide a scaled time-accurate visual representation of the schedule

More Non-functional requirements

- Make schedules available even if the application is down, provided an internet connection is available
- Perform some functions without a live connection to the ‘Web for Students’ registrar web site
- Make compatible with all popular browsers
- Display section states and print schedules without loss of detail

sQFD

Functions/ Features	Class Filters	Allocate non-class time	Long term information availability	Authenticate	
Makes scheduling classes easier	8	3	6	2	19
Makes scheduling a semester easier	7	9	8	2	26
Find schedules in one place	1	1	5	7	14
Total	**16**	**13**	**19**	**11**	**59**

SchedulerPro Product Reliability

- Two hours of unavailability allows for daily backups, service, and reboots of the system
- Connections to server are minimized, reducing overall activity on the server

SchedulerPro
Estimate of Reliability

$R(t) = 1 - F(t)$ $F(t) = P(T \leq t)$

- During load testing, we discovered the test server can support 1500 user queries a minute.
- P(failures/query) = 55/1500 = 0.036
- Thus, F(t) = 3.6%, which means the software is **96.4% reliable**

SchedulerPro Reliability Estimate

$1/\lambda = \text{MTTF} = \varepsilon E/kC$

k = scaling constant = 1

C is complexity = 2.78

E is the development effort = 36.4

ε is the expansion factor = 1.5

$\lambda = 0.05$

t is the continuous execution time for the software

$R(t) =$ **95.12%**

Complexity Chart - Client

- Project Type: online transaction

- Problem Domain: 2
- Architecture Complexity: 3
- Logic Design – Data: 2
- Logic Design – Code: 3
    - Total Score: 10
    - Complexity = (10/18) * 5 = 2.78

Complexity Chart - Server

- Project Type: online transaction

- Problem Domain: 1
- Architecture Complexity: 2
- Logic Design – Data: 2
- Logic Design – Code: 2
    - Total Score: 7
    - Complexity = (7/18) * 5 = 1.94

Complexity Chart - Overall

- Project Type: client/server
- Problem Domain: 2
- Architecture Complexity: 3
- Logic Design – Data: 2
- Logic Design – Code: 3
    - Total Score: 10
    - Complexity = (10/18) * 5 = 2.78

Jan. Function Point Est.

Function	**Low (L)**	**Average (A)**	**High (H)**	**Total**
Outputs	1	3	0	19
Inquiries	8	4	1	49
Inputs	5	7	1	41
Internal Files	3	2	0	24
External Interfaces	2	1	0	10
Total UFP				**143**
Adjustment Factor				0.99
Total AFP				**141**

April Function Points Est.

Function	**Low**	**Average**	**High**	**Total**
Outputs	1	0	1	9
Inquiries	3	0	0	9
Inputs	2	3	0	18
Internal Files	3	1	0	31
External Interfaces	1	1	0	12
Total UFP				**79**
AFP				**82**

History of Function Points

Date	AFP	Project Length*	Projected Finish*
January 27	141	19.7 staff months	August 2006
February 24	104	14.4 staff months	March 2006
April 17	82	8.5 staff months	May 2006

*Using COCOMO Model

ICED-T

Scheduling by:	**Intuitive**	**Consistent**	**Efficient**	**Durable**	**Thoughtful**
Paper	3	2	2	2	3
Excel	3	2	3	3	3
School Scheduler	3	4	4	3	4
SchedulerPro	4	4	5	4	5

Missing: An Installation Plan

Installation

1. Third Party Software Required

Scheduler Pro requires the following products to be already installed on the target machine. Please consult the documentation of each product for installation instructions specific to each.

- Windows 2000, XP, or 2003 Server
- Microsoft IIS, version 5.0 or higher
- Microsoft .NET, version 1.1
- Microsoft SQL Server 2000
- Message Queuing Service (Windows component)
- ASP.NET State Service

Software Requirements Process

- Requirements Elicitation
- Requirements Analysis
- Use Cases
- Requirements Specification
- Prototype/Modeling
- Requirements Management

Creeping Featurism

- Endemic to the Software Industry
  - Occurs on more than 70% of all applications of over 1000 function points
- From a 60 project sample
  - Average creep was 35%
  - Maximum observed was 200%
  - Creeping requirements change about 1% per month
    - For a 3 year project, 1/3 of the delivered requirements would have been added after requirements were initially defined
- Rate of Requirements change is higher than for other forms of engineering (electrical, mechanical, civil)

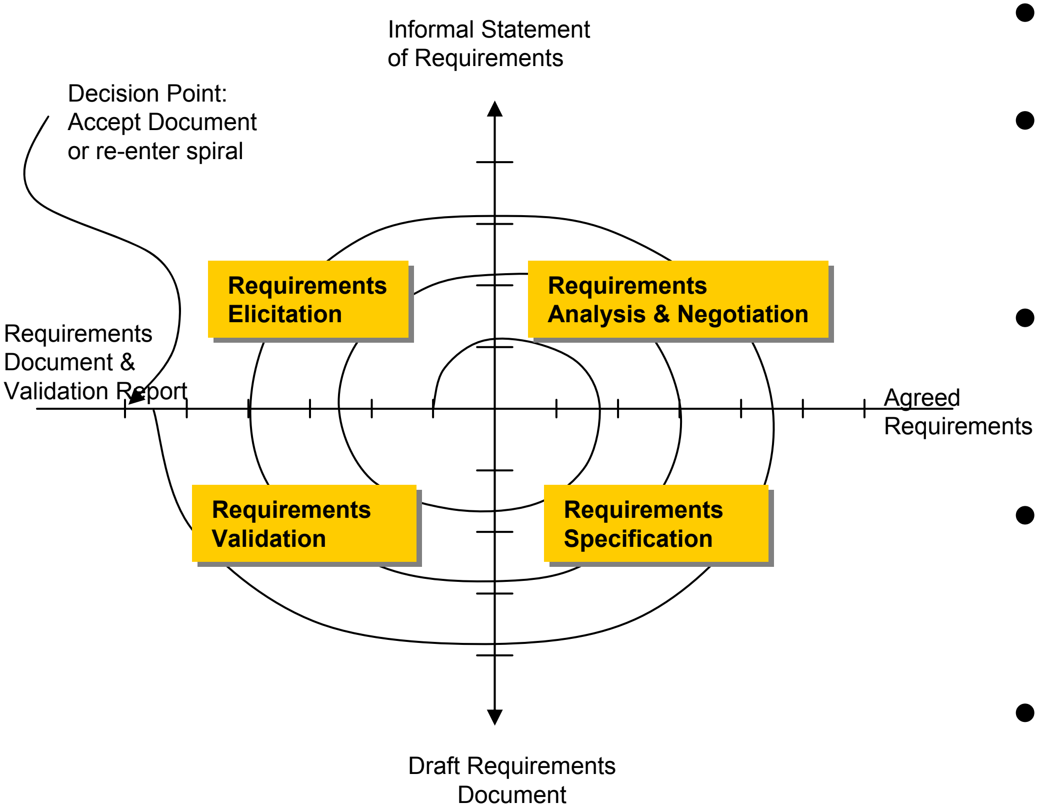
Root Causes of Creeping Requirements

- Uncertainty in resolving true user needs
- For multi-year projects, changes in normal business environment
- Failure to adopt methodologies that limit the risk associated with creeping requirements
- Primitive fundamental technologies for exploring and modeling requirements
- Failure to use technology to measure the impact of creeping requirements
- Engineering trade-off analysis is impossible

Requirements Management

- Establish and maintain a business case to support funding
- Strategic linkages to business and technology organizations –AVOID SHELFWARE
- Continuous customer agreement on requirements
- Requirements agreement used as a basis for estimating, planning, implementing and tracking
- FORMAL COMMITMENT PROCESS

Requirements Engineering Process

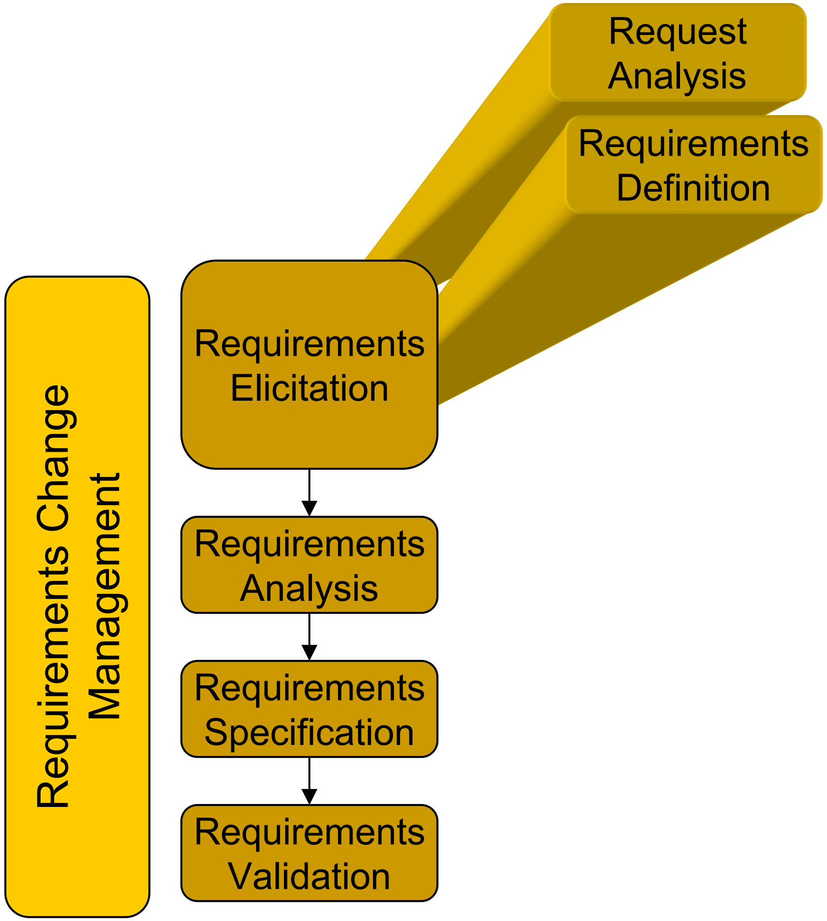

Informal Statement of Requirements
Decision Point: Accept Document or re-enter spiral
Requirements Elicitation
Requirements Analysis & Negotiation
Requirements Document & Validation Report
Agreed Requirements
Requirements Validation
Requirements Specification
Draft Requirements Document

- Process Models
- Process Actors and Stakeholders
- Process Support and Management
- Process Quality and Improvements
- Relationship to the Business Decision

Real-time Requirements

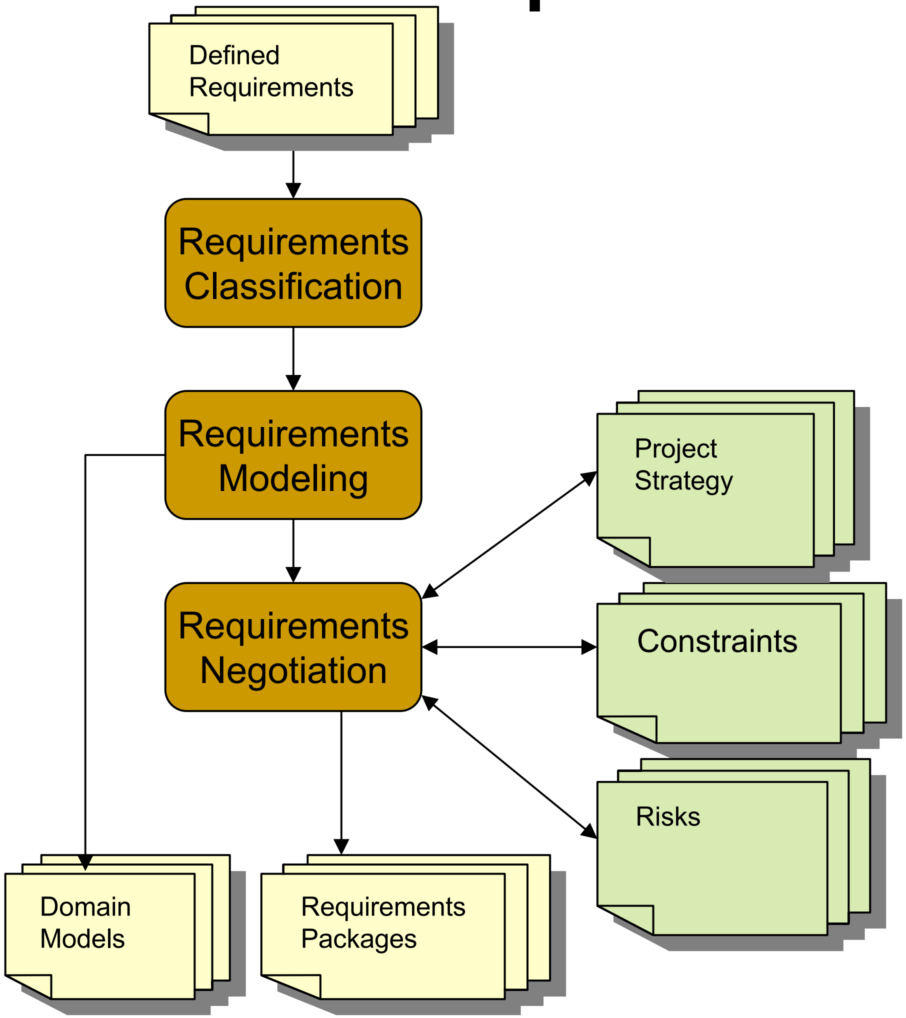
- Computer uses only past and present data
- Data is sampled at a constant rate, the pulse repetition rate of the radar,
- The calculations are completed in time to adjust the radar for the next sample
- The equations are stable

Requirements Process



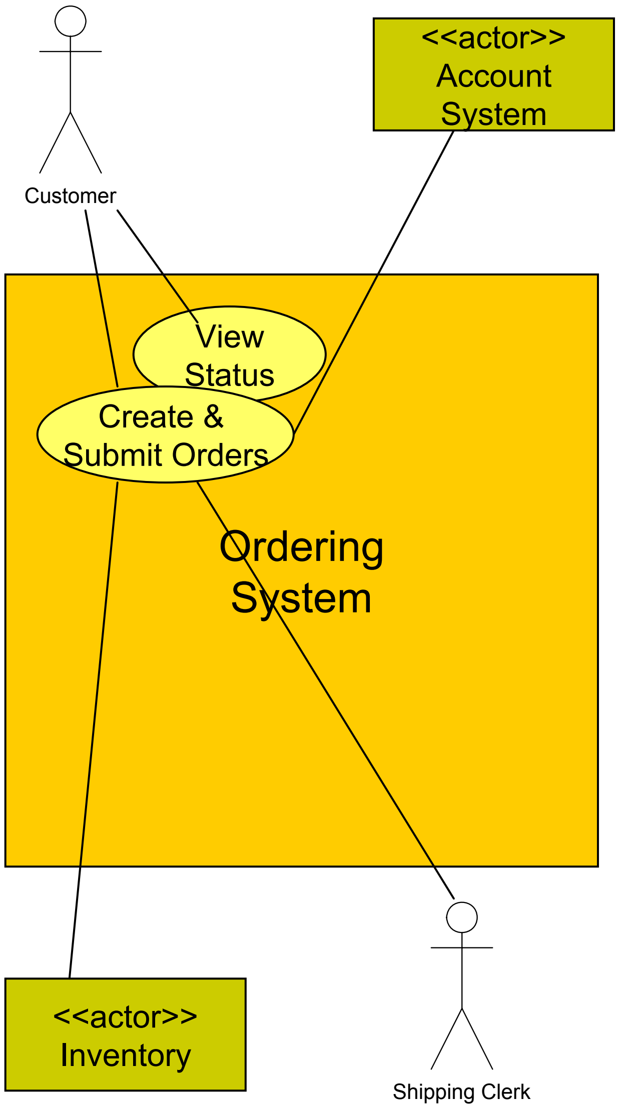
- Elicitation
  - Request Analysis
    - Sourcing & Screening
  - Definition
    - Purposeful
    - Understand value
- Analysis
  - Interrelationships
  - Prioritization
  - Risk & Cost Assessment
- Specification
  - Modeling
- Validation
  - Agreement
- Change Management

Requirements Analysis



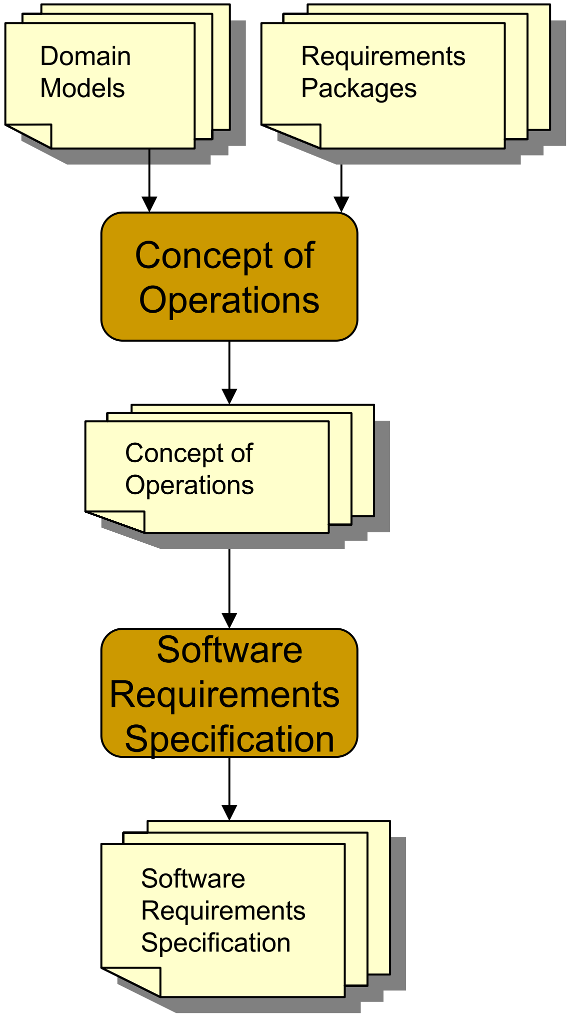
- Requirements Classification
  - Product/Process
  - Priority/Risk
  - Scope/Allocation
  - Volatility/Stability
- Conceptual Modeling
  - Understanding & Communication
  - Functional Architecture
- Requirements Negotiation
  - Trade Offs
  - Consensus with Stakeholder

Example



- Develop Use Cases
  - Focus on Goals
  - Identify Actors
  - Identify Main Tasks
- Use Case Concept
  - Complete, orthogonal, externally visible functionality
  - Initiated by an actor
  - Identifiable value to the actor

Software Requirements Spec.

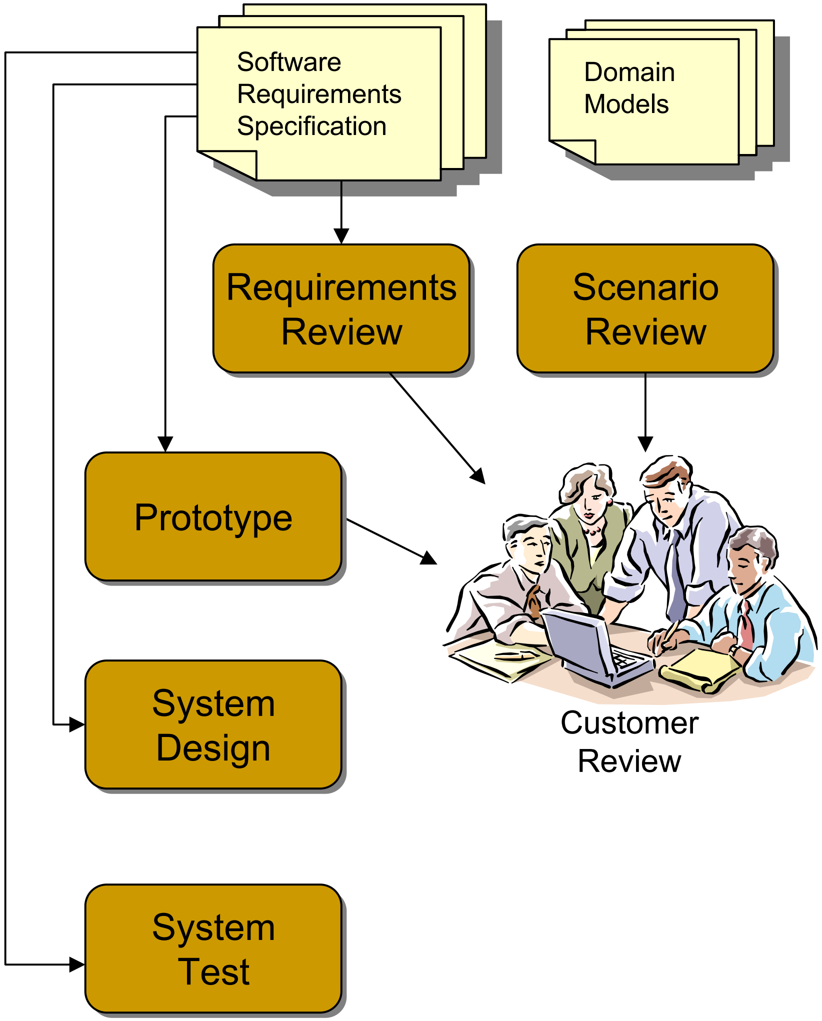


- Concept of Operations
  - System Characteristics
  - User Operational Needs
  - Domain Perspective
  - Constraints
  - Trade-Off Analysis
- Software Requirements Specification
  - Basis for Agreement
  - Reduce Development
  - Provide Basis for Estimation
  - Baseline for Validation & Verification
  - Basis for Enhancement

Requirements Specification Spec

1. Project Title, Revision Number and Author
2. Scope and Purpose of the system
3. Measurable Operational Value
4. Description
5. Feature List including ICED T and Simplified QFD analysis
6. Interfaces
7. Constraints
8. Change Log and Expected Changes
9. Responses to the unexpected
10. Measurements
11. Glossary
12. References

Requirements Validation



Software Requirements Specification

Domain Models

Requirements Review

Scenario Review

Prototype

Customer Review

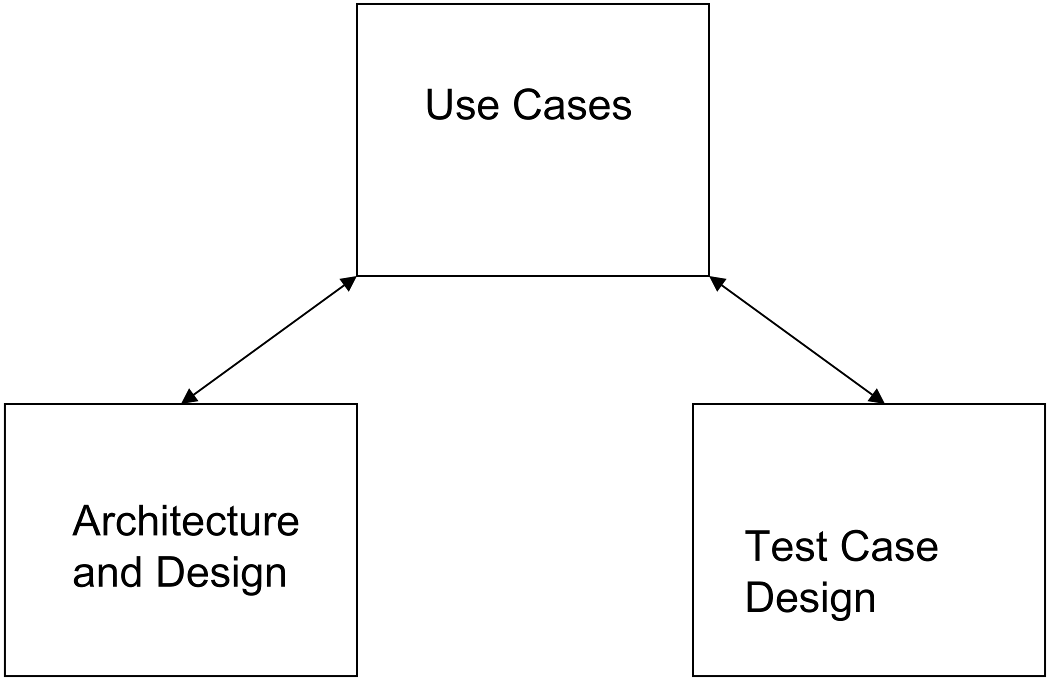
System Design

System Test

- Requirements Reviews
  - Formal
  - Customer Representative
- Prototyping
- Model Validation
  - Scenario Reviews with Customers
  - Model Consistency
- Acceptance Tests
  - Verifiable Requirements

As of
9/9/04

Use Cases Drive Development



Use Cases

Architecture and Design

Test Case Design

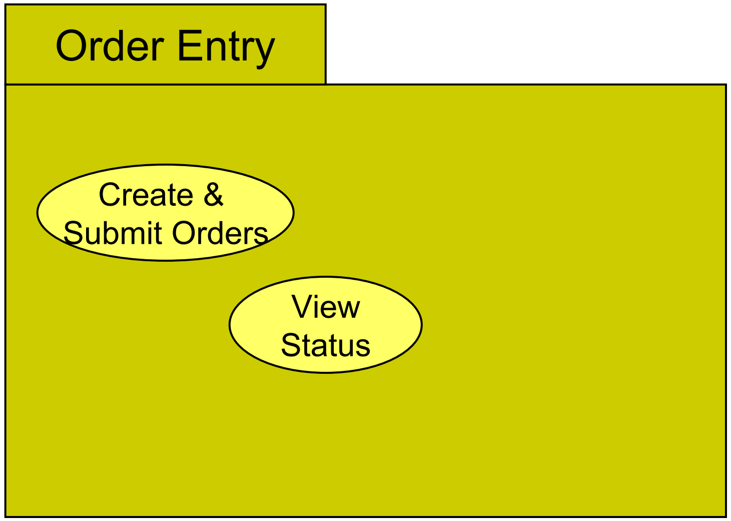
Use Case Documentation

Feature	Use Case
The customer can order on the web.	UC 1
The customer builds the order by selecting items from the on-line catalog and specifying a quantity.	UC 1
Only customers that have an account can create an order.	UC 1
At any time during the process of creating an order, the customer can determine the current price of the order.	UC 1
The customer signifies that the order is complete by submitting the order. When an order is submitted, it is assigned an order number.	UC 1
Customers with the priority privilege may designate an order as priority.	UC 1a
The customer can view the status of an order at any time by logging on to web site and requesting status on all open orders.	UC 2
Once an order is submitted, it is checked to see if it is pre-paid or whether the customer has an account in good standing. If these conditions are not met, the order is held until the conditions are met or the order is cancelled.	UC 1
…	

Use Case Documentation

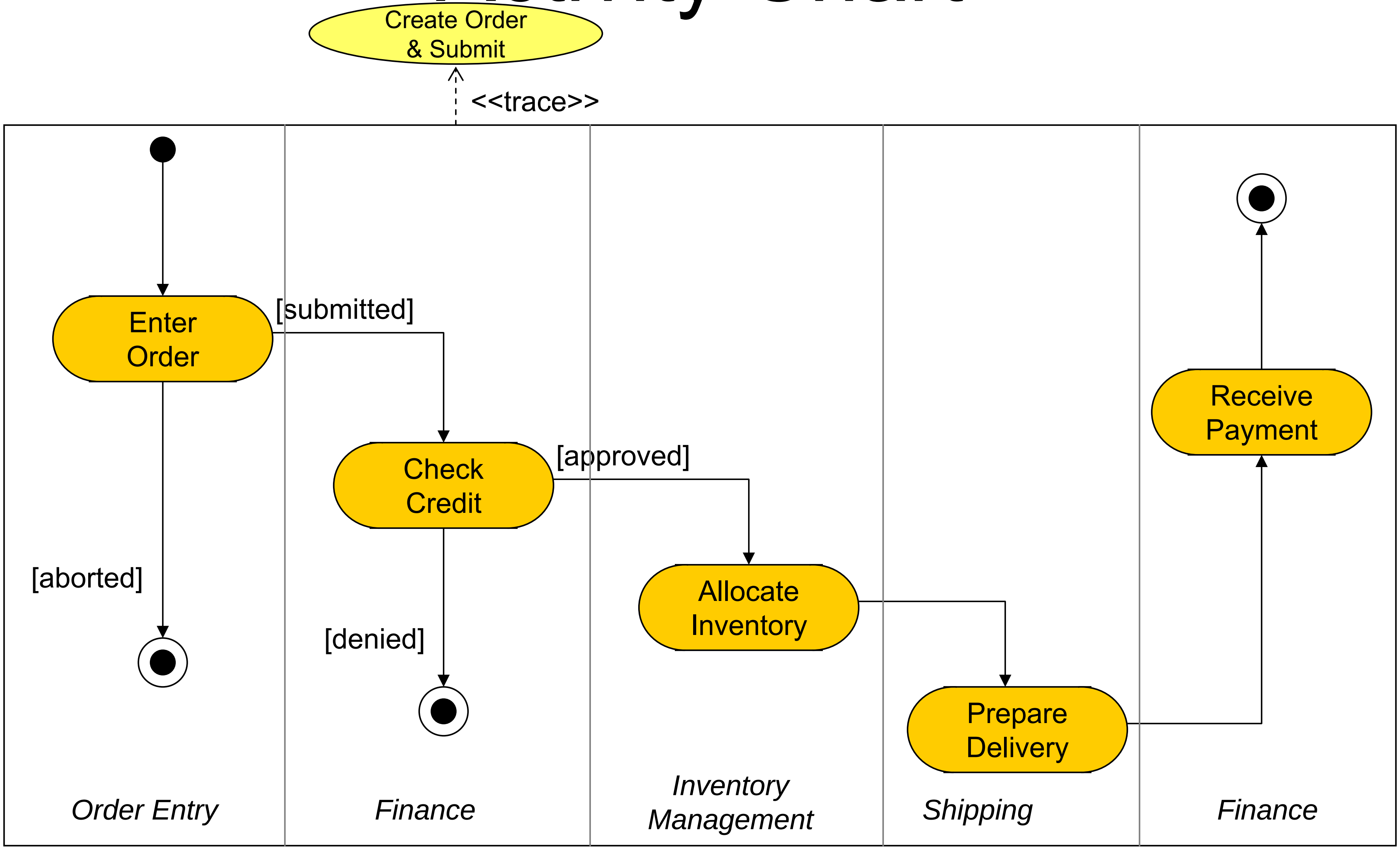
Use Case 1	Create Order & Submit
Brief Description	A customer wishes to order. Provided that the customer has a non-delinquent account or has pre-paid, the product is removed from inventory and delivered to the customer.
Actors	Customer, Inventory, Shipping Clerk, Account System
Trigger	Customer visits web site & creates an order.
Preconditions	Customer has established and account. Customer email address is known. Customers are pre-designated to enter priority orders.
Main flow	Customer visits web site, signs on and is validated. Customer selects items from the online catalog and builds an order. Customer is appraised of current cost of order. Customer may denote that the order is a priority Customer submits order when done. A customer order number is assigned and the customer's credit and account status are checked. If credit is OK or the account shows pre-payment, then the order is sent to the inventory system. …..
Alternative flows	Priority Order Account is delinquent. Action taken ? Cancelled ? Changes to or cancellation of the order? Order cannot be fulfilled ?
Postconditions	Order has been created and is either been cancelled or been fulfilled.

Package Diagram

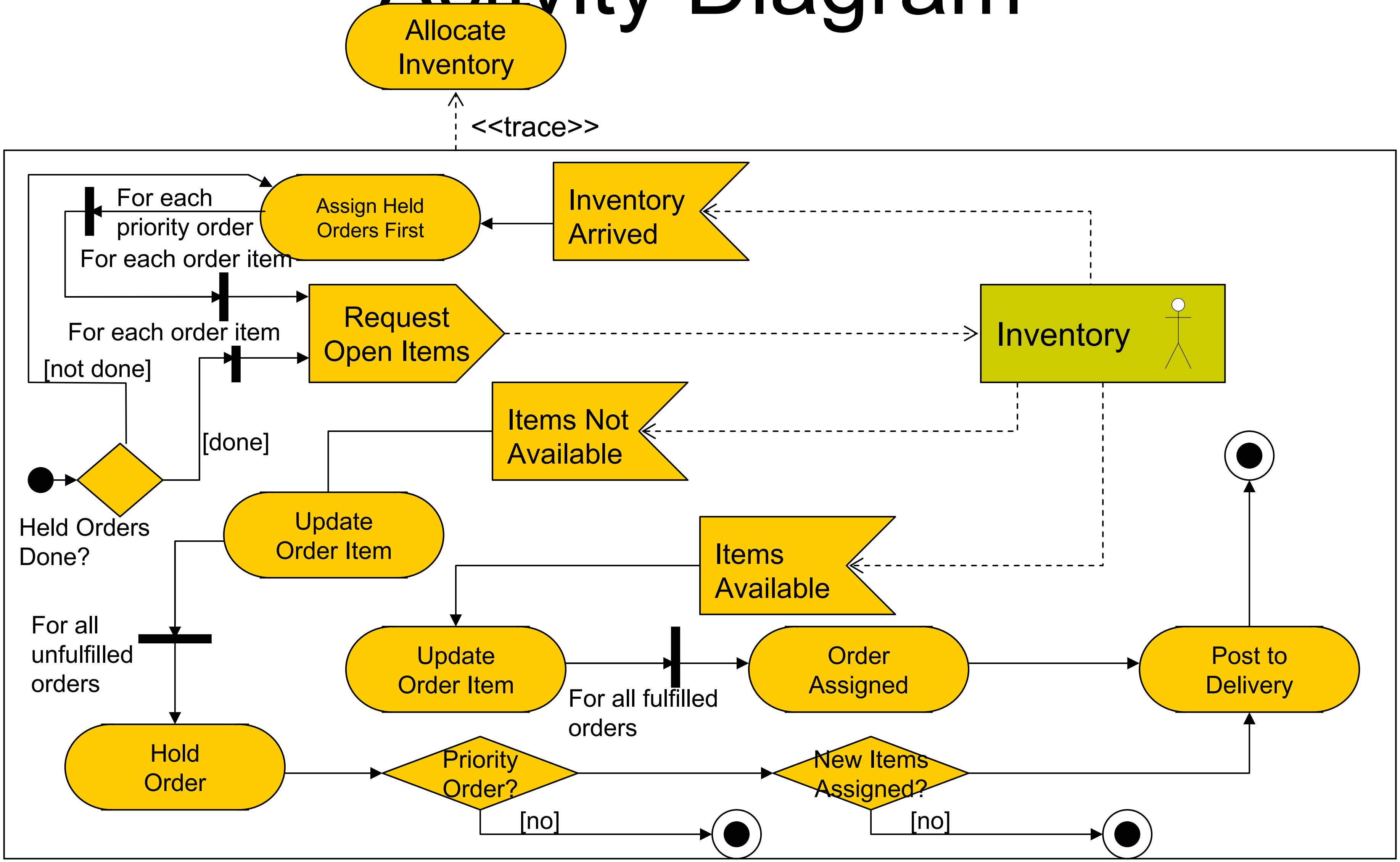


- Groups related use cases
- Forms basis for a functional partitioning from the users point of view.
- Shorthand for tracking within the project

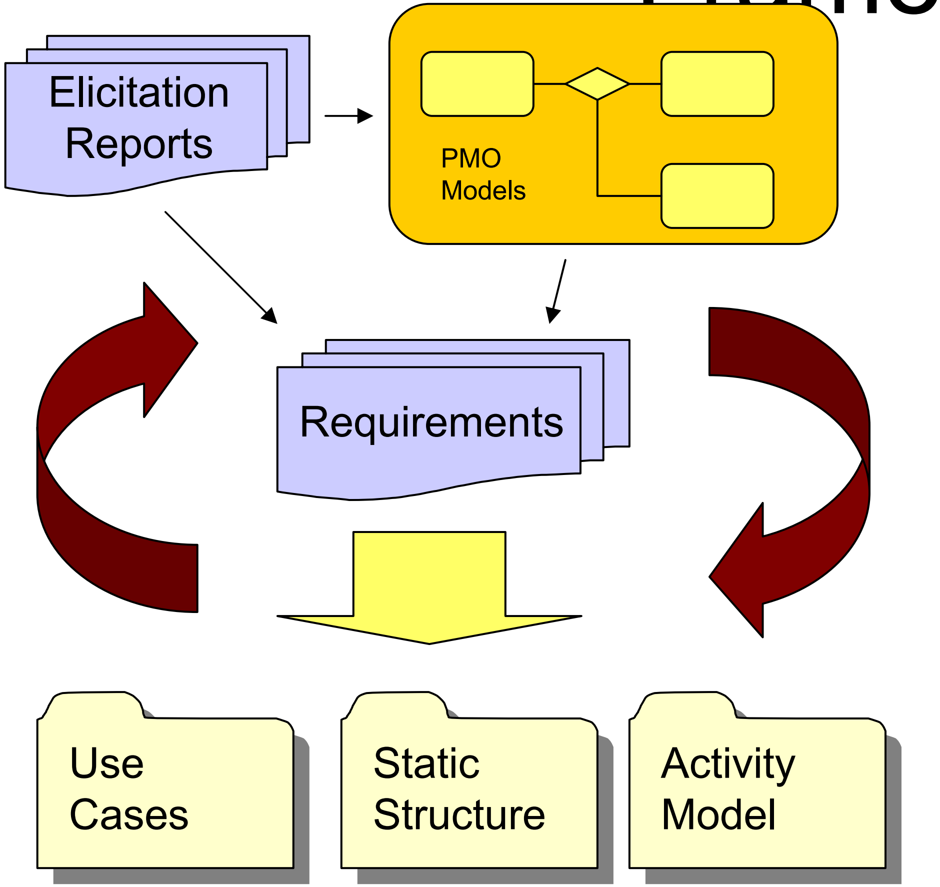
Activity Chart



Activity Diagram



Mapping Requirements to a Framework



- ICED T
  - Intuition
  - Consistent
  - Efficient
  - Durable
  - Thoughtful
- UML Framework
  - Use Cases
  - Structure
  - Business Rules

Case History: Cardiac Data Analysis

Propectus: Create a graphical interface that displays a time series graph with **selected** points of inflection, and allows for user modification of points.

MOV

Background: Drs. determine points manually taking 20-30 minutes, or with tools that take 2 – 10 minutes.

MOV: Our software allows points to be chosen, on average, **4 times** faster than previous available tools with 80% accuracy

Function Points

Siemens Unadjusted Function Point Analysis

Updated 2/15/06

Use Cases	Transactions	Type	Complexity	UFP
Tool 1:				
Input data file	1	I	4	4
User point modification	1	I	6	6
Load User Point Changes	1	I	3	3
Screenshot	1	O	4	4
Save User Point Changes	1	O	4	4
Tool 2:				
Curve Fitting Algorithm	1	I	6	6
Find/Send points to tool 1	1	O	7	7
Point Selecting Algorithm	7	N	15	105
Tool 3:				
Data from tool 1	1	I	4	4
Rotating image (user control)	2	I	6	12
Snapshot	1	O	4	4
Coloration of Image logic	1	N	15	15
3-D imaging/rotation logic	1	N	15	15
Total Unadjusted Function Points				189

Simplifications

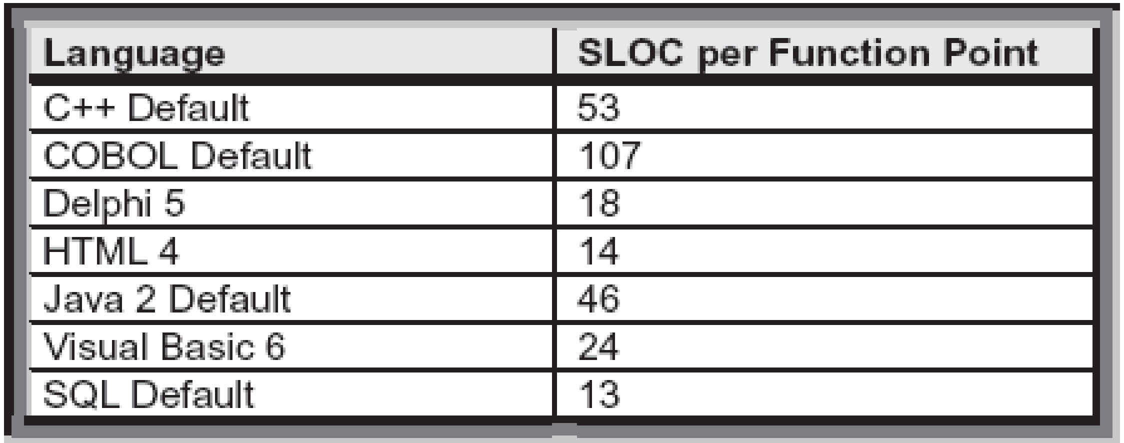
- Narrowing of the requirements to only consider data from ‘healthy hearts.’
- Open source code: *NTGraph*.
- Before simplifications

  Unadjusted Function Points were 356

  now they are 189.

Function Points to LOC

- This conversion cart is shown below



Language	SLOC per Function Point
C++ Default	53
COBOL Default	107
Delphi 5	18
HTML 4	14
Java 2 Default	46
Visual Basic 6	24
SQL Default	13

- Thus for our system using the conversion factor of **53 LOC/FP** since we will be programming in C++ we can find the estimated LOC for our system through the following formula:

LOC = 53 * UFP

- Thus we can solve this equation to find the LOC estimated for our system.

LOC = 53 * UFP, where **UFP = 189**

LOC = 53 * 189 = **10,017 LOC**

From http://www.stsc.hill.af.mil/crosstalk/2005/04/0504Roetzheim.html

COCOMO

Effort/Staff Hours = A*(KNCSLOC)**B

Where KNCSLOC ≡ thousands of new and changed lines of code,

A ≡ small project productivity,

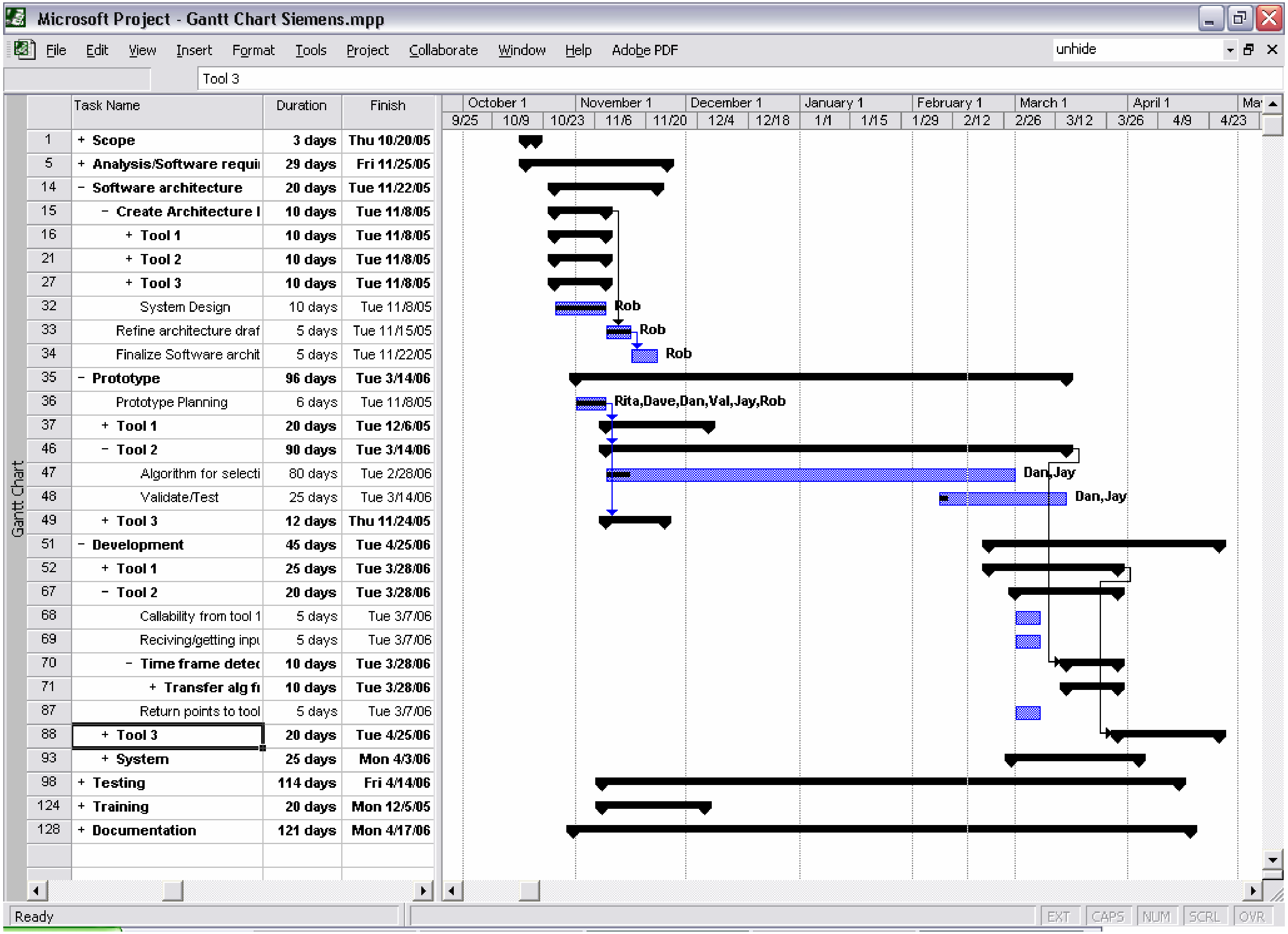
B≡ complexity factor

We use:

- Semidetached: A=3.0 B=1.12
- KNCSLOC = 10

Effort = $3.0*(10)^{1.12}$ **= 39.623 ≈ 40 staff months**

Gantt Chart



	Task Name	Duration	Finish
1	+ **Scope**	**3 days**	**Thu 10/20/05**
5	+ **Analysis/Software requi**	**29 days**	**Fri 11/25/05**
14	− **Software architecture**	**20 days**	**Tue 11/22/05**
15	− **Create Architecture I**	**10 days**	**Tue 11/8/05**
16	+ **Tool 1**	**10 days**	**Tue 11/8/05**
21	+ **Tool 2**	**10 days**	**Tue 11/8/05**
27	+ **Tool 3**	**10 days**	**Tue 11/8/05**
32	System Design	10 days	Tue 11/8/05
33	Refine architecture draf	5 days	Tue 11/15/05
34	Finalize Software archit	5 days	Tue 11/22/05
35	− **Prototype**	**96 days**	**Tue 3/14/06**
36	Prototype Planning	6 days	Tue 11/8/05
37	+ **Tool 1**	**20 days**	**Tue 12/6/05**
46	− **Tool 2**	**90 days**	**Tue 3/14/06**
47	Algorithm for selecti	80 days	Tue 2/28/06
48	Validate/Test	25 days	Tue 3/14/06
49	+ **Tool 3**	**12 days**	**Thu 11/24/05**
51	− **Development**	**45 days**	**Tue 4/25/06**
52	+ **Tool 1**	**25 days**	**Tue 3/28/06**
67	− **Tool 2**	**20 days**	**Tue 3/28/06**
68	Callability from tool 1	5 days	Tue 3/7/06
69	Reciving/getting inpu	5 days	Tue 3/7/06
70	− **Time frame detec**	**10 days**	**Tue 3/28/06**
71	+ **Transfer alg fi**	**10 days**	**Tue 3/28/06**
87	Return points to tool	5 days	Tue 3/7/06
88	+ **Tool 3**	**20 days**	**Tue 4/25/06**
93	+ **System**	**25 days**	**Mon 4/3/06**
98	+ **Testing**	**114 days**	**Fri 4/14/06**
124	+ **Training**	**20 days**	**Mon 12/5/05**
128	+ **Documentation**	**121 days**	**Mon 4/17/06**

ICED-T

ICED-T					
Build Metric	**Requirements**	**Architecture**	**Prototype**	**Development**	**Final**
Intuitive	2	3	3	1	3
Consistent	3	4	2	4	4
Efficient	3	4	3	2	4
Durable	5	4	2	5	5
Thoughtful	4	5	4	4	4

Reliability Requirement

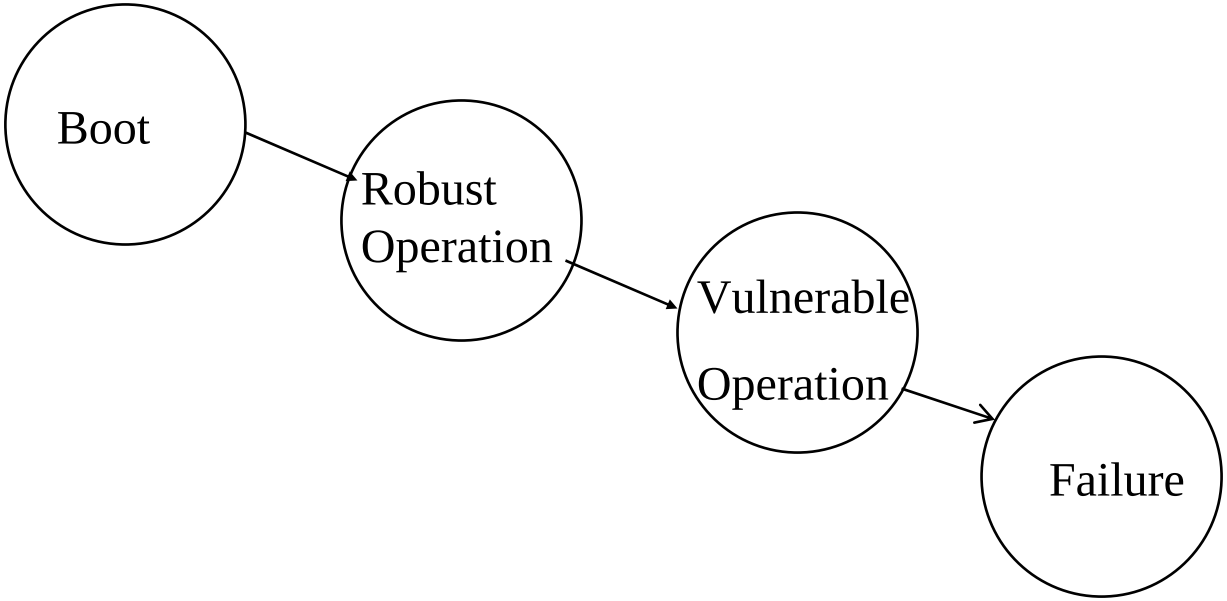
Heisenbugs

Latent faults causing gradual deterioration a software process with respect to the use of some resource resulting in a failure.

Case Study: Pluto Express

- Duplicated computers for reliability.
- One computer runs at a time to minimize power drain.
- Hardware detects computer failure and switches to backup.
- Assume Prob. of unsuccessful switchover = 10^{-8}

Case Study: Pluto Express



Boot

Robust Operation

Vulnerable Operation

Failure

Case Study: Pluto Express

Let the rate of going from Robust State to Vulnerable State be: 10^{-3}

Let the rate of going from the Vulnerable State to Failure be: 10^{-4}

Then using Rejuvenation with a 6 week period increases system reliability by a factor of 10

Case Study: Pluto Express

If the failures double and the Rejuvenation interval is halved, system reliability with Rejuvenation is about100 times more reliable then systems without Rejuvenation.

Parnas reliability checklist

Response to all failures in communication, secondary storage, memory, or any hardware that may interrupt a transaction:

- ➢The SQL Server DBMS will not commit incomplete transactions. User will be notified of the error, and will have to redo the transaction.

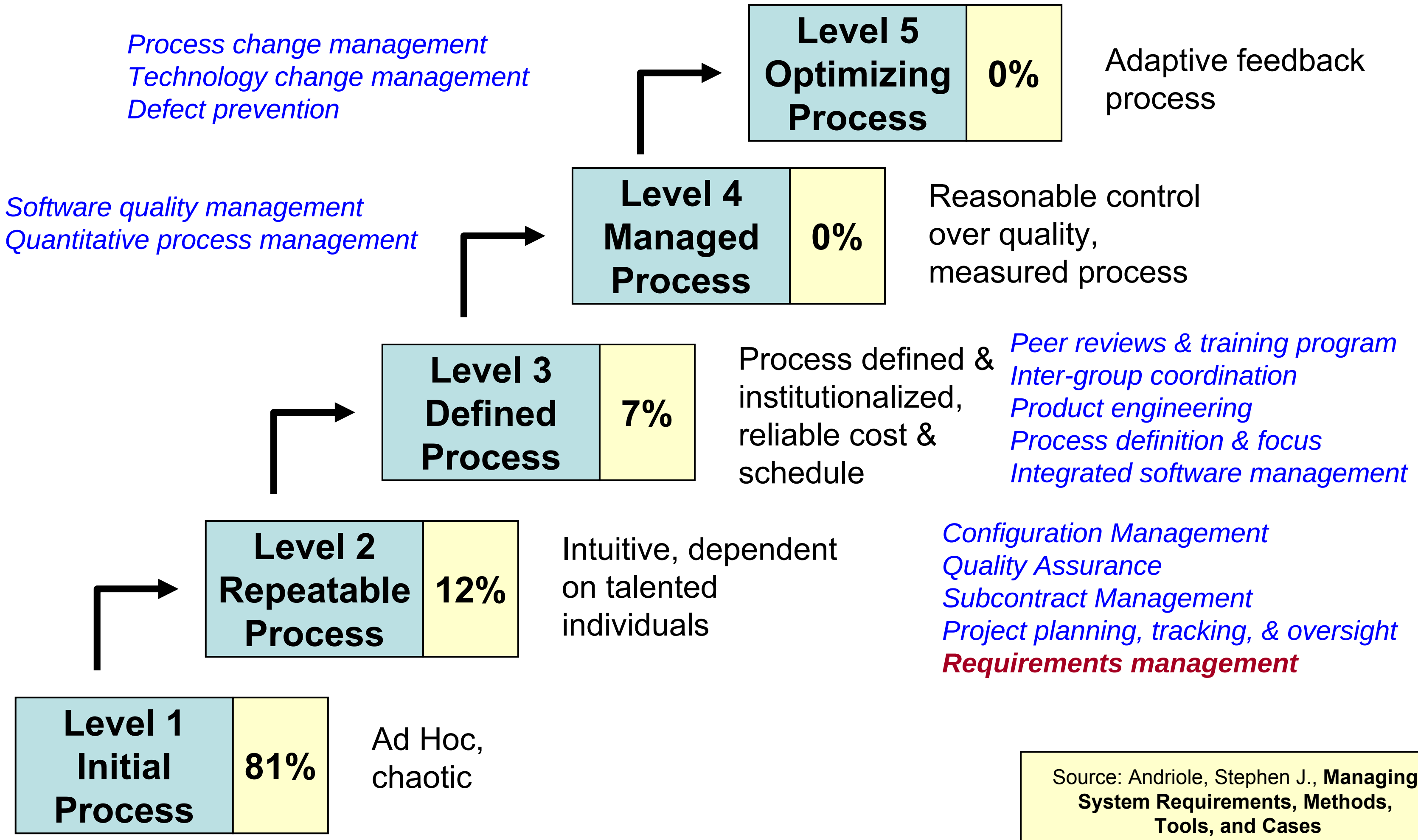
- Operator errors:
  - ➢Important operations are confirmed before they are completed to avoid large accidental errors.

Conditions That Cause Unreliability

- Poor Algorithms
- Missing Deadlines
- Roundoff Error Build Up
- Memory Leaks
- Broken Pointers

SEI Capability Model

Key Process Areas



Process change management
Technology change management
Defect prevention

Level 5 Optimizing Process **0%**

Adaptive feedback process

Software quality management
Quantitative process management

Level 4 Managed Process **0%**

Reasonable control over quality, measured process

Level 3 Defined Process **7%**

Process defined & institutionalized, reliable cost & schedule

Peer reviews & training program
Inter-group coordination
Product engineering
Process definition & focus
Integrated software management

Level 2 Repeatable Process **12%**

Intuitive, dependent on talented individuals

Configuration Management
Quality Assurance
Subcontract Management
Project planning, tracking, & oversight
Requirements management

Level 1 Initial Process **81%**

Ad Hoc, chaotic

Source: Andriole, Stephen J., **Managing System Requirements, Methods, Tools, and Cases**
McGraw-Hill, 1996

People

Software Trustworthiness depends on people:

I propose that customers insist that software products identify a Software Architect and Software Project Manager in their contracts

Software Architect:

- Affirms that the software product solves the customer’s problem
- Affirms that the software product is suitably reliable, easy-to-use, extendible, not harmful and robust. That it is trustworthy.
- Affirms that the requirements are valid.

Software Project Manager:

- Affirms that the software was successfully tested against the requirements.
- Affirms and identifies the good software engineering processes were used in the software development and integration.
- Affirms that the project is within budget, on-time and performs satisfactorily.

People

Software Trustworthiness depends on people:

I propose that customers insist that software products identify a Software Architect and Software Project Manager in their contracts

Software Architect:

- Affirms that the software product solves the customer’s problem
- Affirms that the software product is suitably reliable, easy-to-use, extendible, not harmful and robust. That it is trustworthy.
- Affirms that the requirements are valid.

Software Project Manager:

- Affirms that the software was successfully tested against the requirements.
- Affirms and identifies the good software engineering processes were used in the software development and integration.
- Affirms that the project is within budget, on-time and performs satisfactorily.

Systems Engineering

Systems Engineering

“An interdisciplinary approach and means to enable the realization of successful systems.”

– INCOSE (The International Council on Systems Engineering)

System:

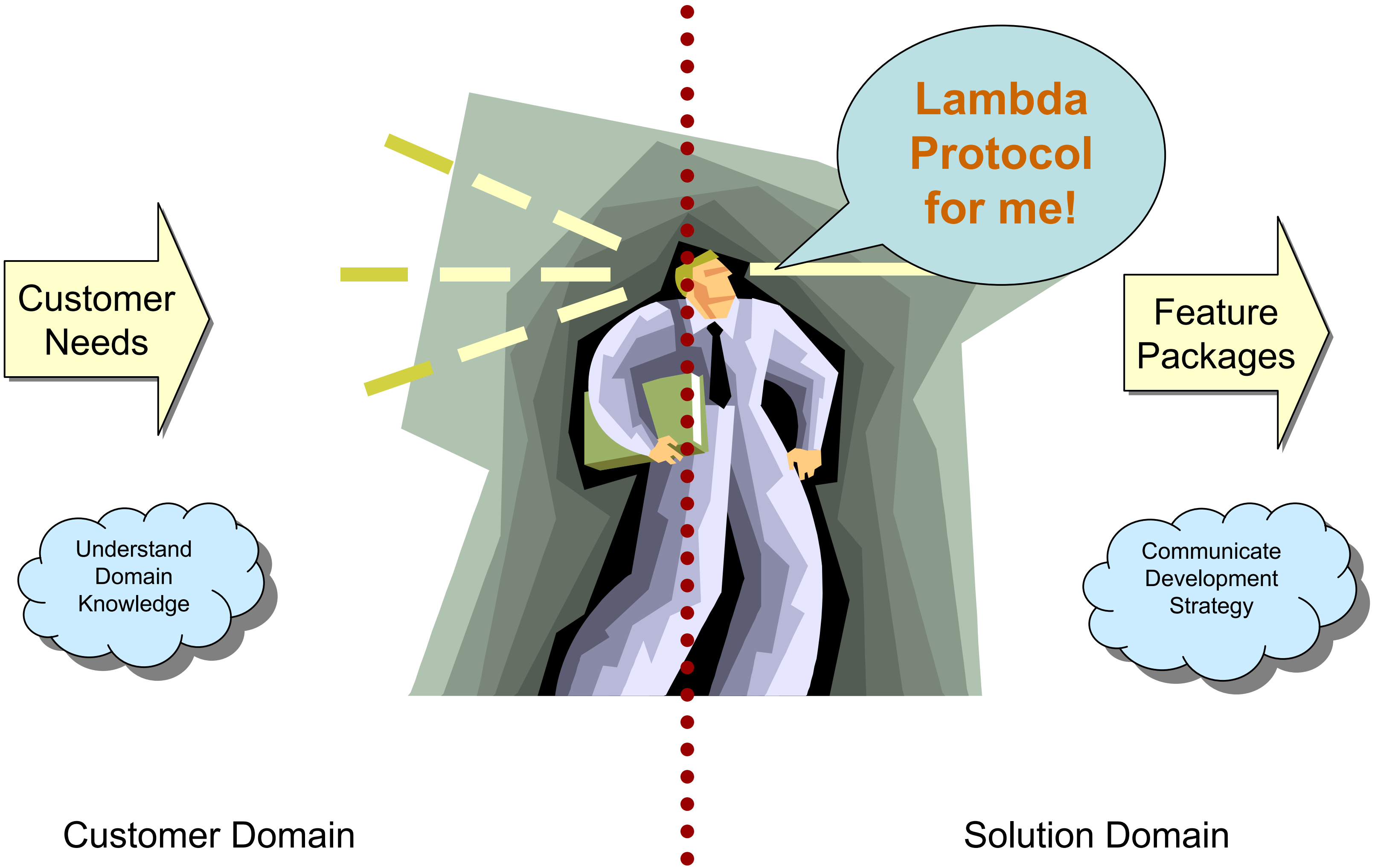
“A group of interacting, interrelated, or interdependent elements that together form a complex whole.”

– NGE Project (Next Generation Education Project)

QSE Lambda Protocol

- Prospectus
- Measurable Operational Value
- Prototyping or Modeling
- sQFD
- Schedule, Staffing, Quality Estimates
- ICED-T
- Trade-off Analysis

Requirements Engineer



Customer Needs

Lambda Protocol for me!

Feature Packages

Understand Domain Knowledge

Communicate Development Strategy

Customer Domain

Solution Domain