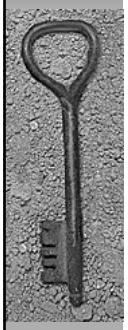
The Software Development Process

- Identifying the model assumptions for how a "Target of Evaluation" is generated.
- Understanding the gap between the model and the reality.
- Tools that are ready to hand.

600.436 High-Assurance Systems Fall 01/Introduction 1



Phases of SW Development

(Cheap)

Design

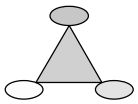
Develop

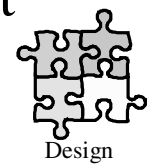
Deploy

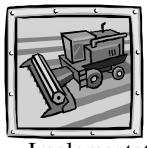
(Bankrupt)

Requirements (1x)

Rigorous Model (?)


Architecture (1ox)

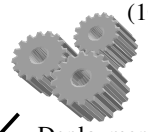

Design (100x)


Implementation (1,000x)

Validation/QA (10,000x)

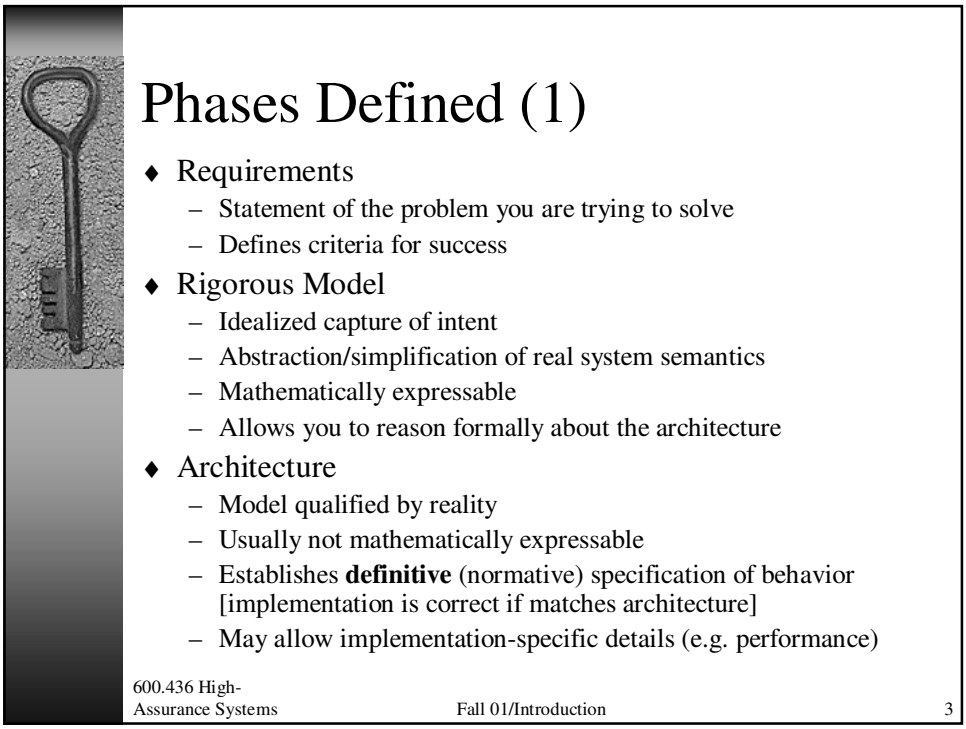
Delivery (100,000x)

Acceptance (1,000,000x)


Deployment (10,000,000x)

Release Feedback Cycle

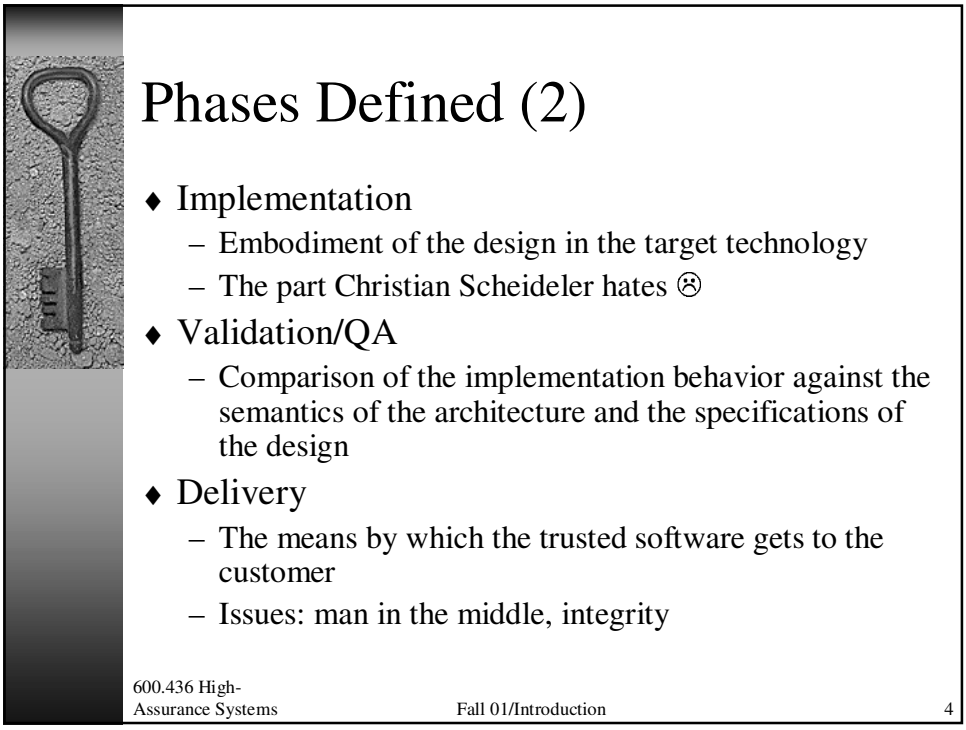
600.436 High-Assurance Systems Fall 01/Introduction 2



Phases Defined (1)

- Requirements
  - Statement of the problem you are trying to solve
  - Defines criteria for success
- Rigorous Model
  - Idealized capture of intent
  - Abstraction/simplification of real system semantics
  - Mathematically expressable
  - Allows you to reason formally about the architecture
- Architecture
  - Model qualified by reality
  - Usually not mathematically expressable
  - Establishes **definitive** (normative) specification of behavior [implementation is correct if matches architecture]
  - May allow implementation-specific details (e.g. performance)

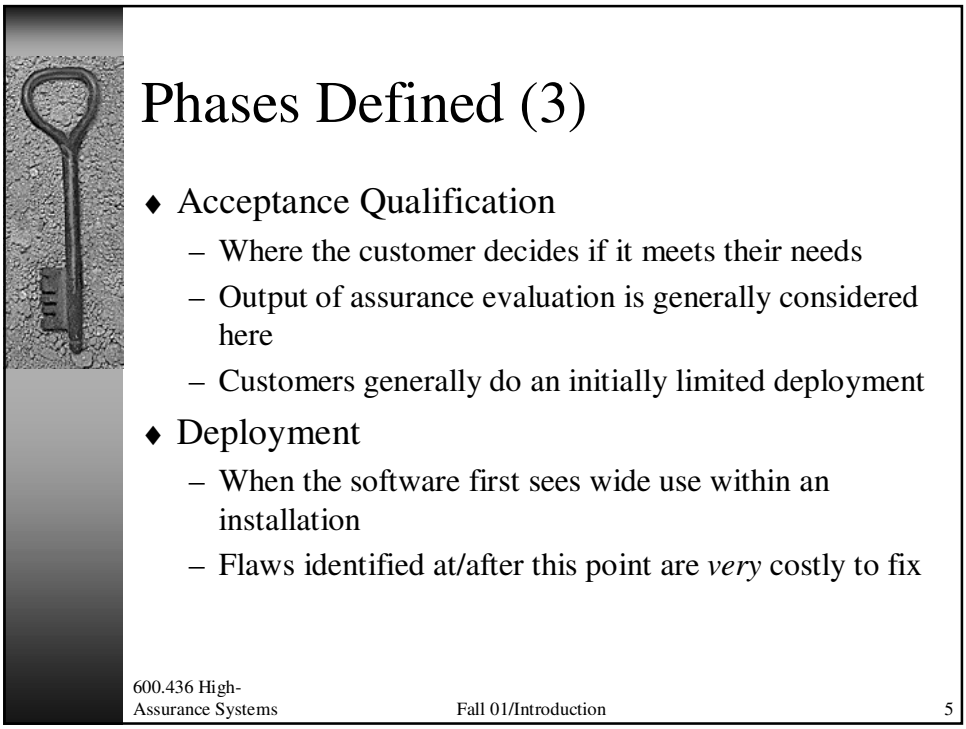
600.436 High-Assurance Systems Fall 01/Introduction 3



Phases Defined (2)

- Implementation
  - Embodiment of the design in the target technology
  - The part Christian Scheideler hates ☹
- Validation/QA
  - Comparison of the implementation behavior against the semantics of the architecture and the specifications of the design
- Delivery
  - The means by which the trusted software gets to the customer
  - Issues: man in the middle, integrity

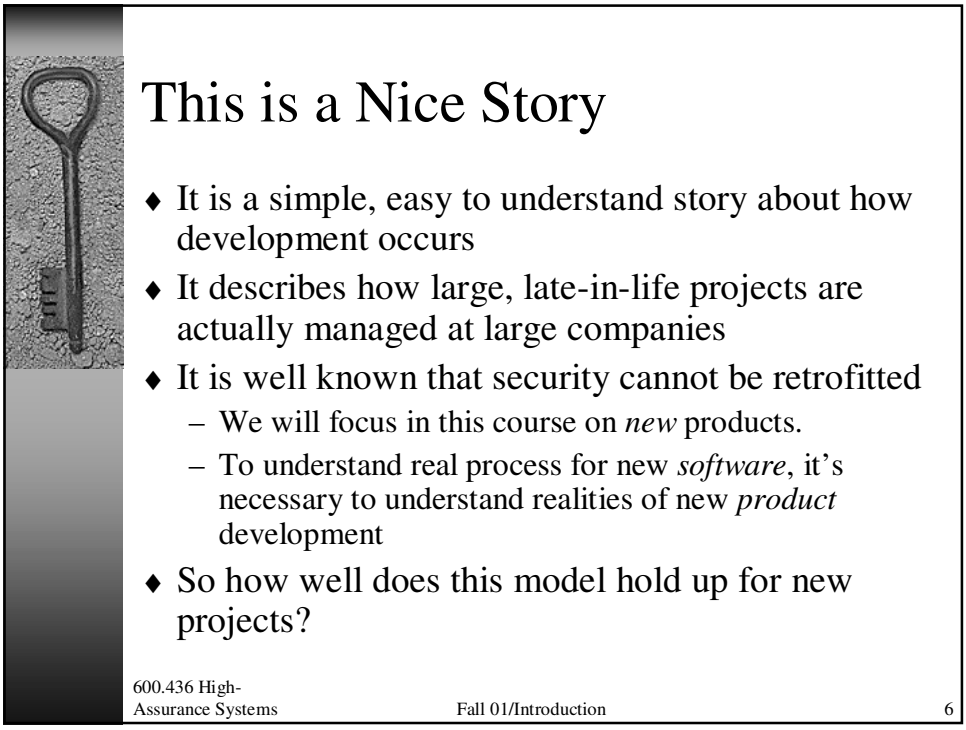
600.436 High-Assurance Systems Fall 01/Introduction 4



Phases Defined (3)

- Acceptance Qualification
  - Where the customer decides if it meets their needs
  - Output of assurance evaluation is generally considered here
  - Customers generally do an initially limited deployment
- Deployment
  - When the software first sees wide use within an installation
  - Flaws identified at/after this point are *very* costly to fix

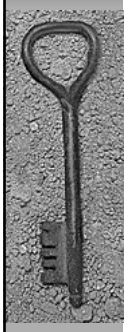
600.436 High-Assurance Systems Fall 01/Introduction 5



This is a Nice Story

- It is a simple, easy to understand story about how development occurs
- It describes how large, late-in-life projects are actually managed at large companies
- It is well known that security cannot be retrofitted
  - We will focus in this course on *new* products.
  - To understand real process for new *software*, it's necessary to understand realities of new *product* development
- So how well does this model hold up for new projects?

600.436 High-Assurance Systems Fall 01/Introduction 6



Great Technology != Great Product

"…when weighing the competitive strengths of Apple and IBM, we win in one key category: PRODUCT SUPERIORITY. It is out preemptive Lisa Technology at a recognizable price/performance advantage that will allow us to successfully compete with IBM **for the next 18/24 months**."

Macintosh Product Introduction Plan
October 7, 1983

"To **maintain** leadership in a mainstream market, you must at least keep the pace with the competition. It is no longer necessary to be the technology leader, nor is it necessary to have the very best product."

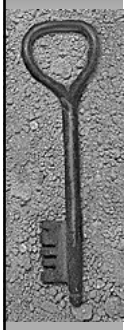
Crossing the Chasm, p. 49
Geoffrey A. Moore

"Disruptive technologies bring to market a very different value proposition than had been available previously… are first commercialized in emerging or insignificant markets … by and large [cannot be used by leading customers] … [and are] initially embraced by the least profitable customers in a market."

"Ironically, in each of the instances studied in this book it was disruptive technology that precipitated the leading firms' failure."

The Innovator's Dilemma, p. xv
Clayton M. Christensen

600.436 High-Assurance Systems Fall 01/Introduction 7

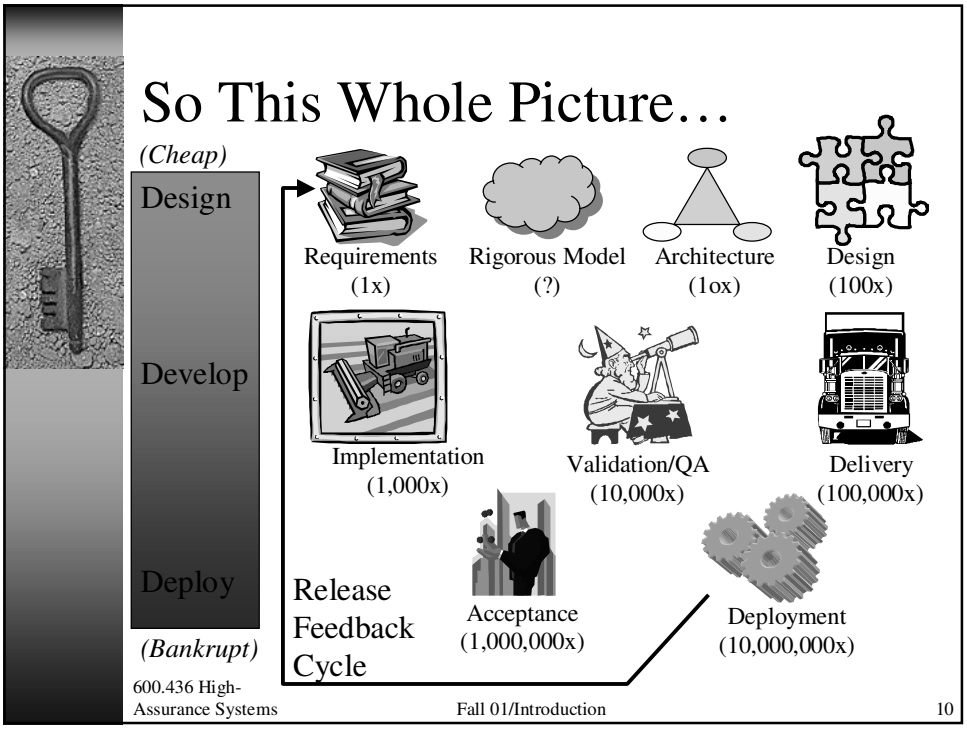


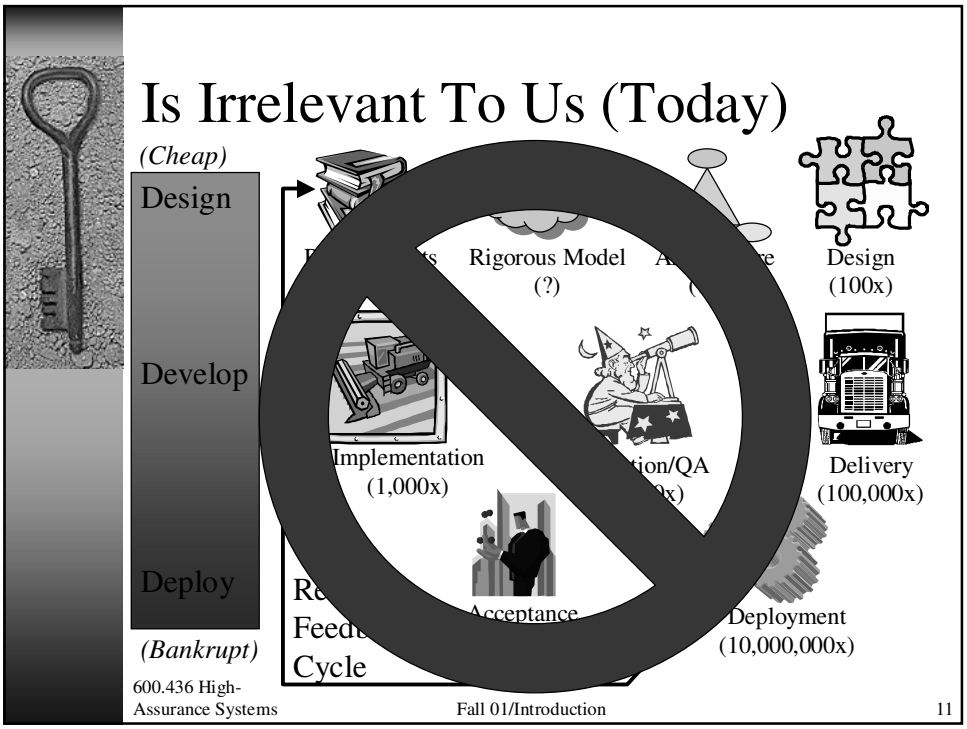
In Other Words…

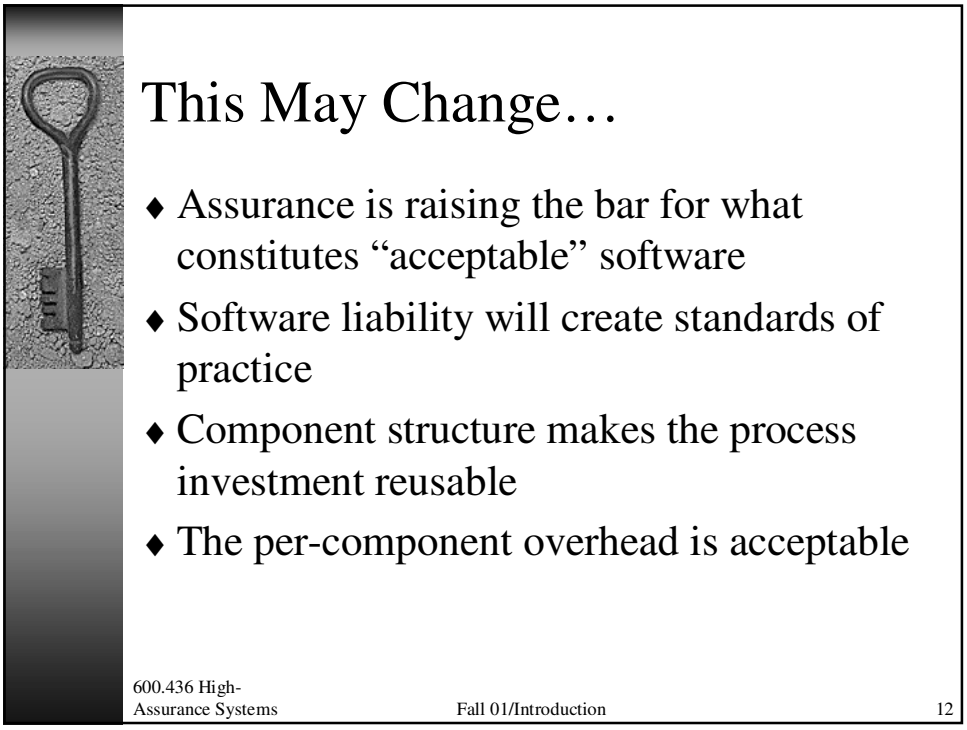
- Technology buys *entry* to a market, if it:
  1. Strikes from ambush (surprise)
  2. Attacks an unperceived/undefended position
  3. At lower cost
- But it doesn't keep you there!
  - Continued evolution, repackaging, and repositioning are required for each new market segment
  - Yesterday's segment is tomorrow's road kill
  - Reputation takes a long time to catch up with you
- Development Model != LifeCycle Model

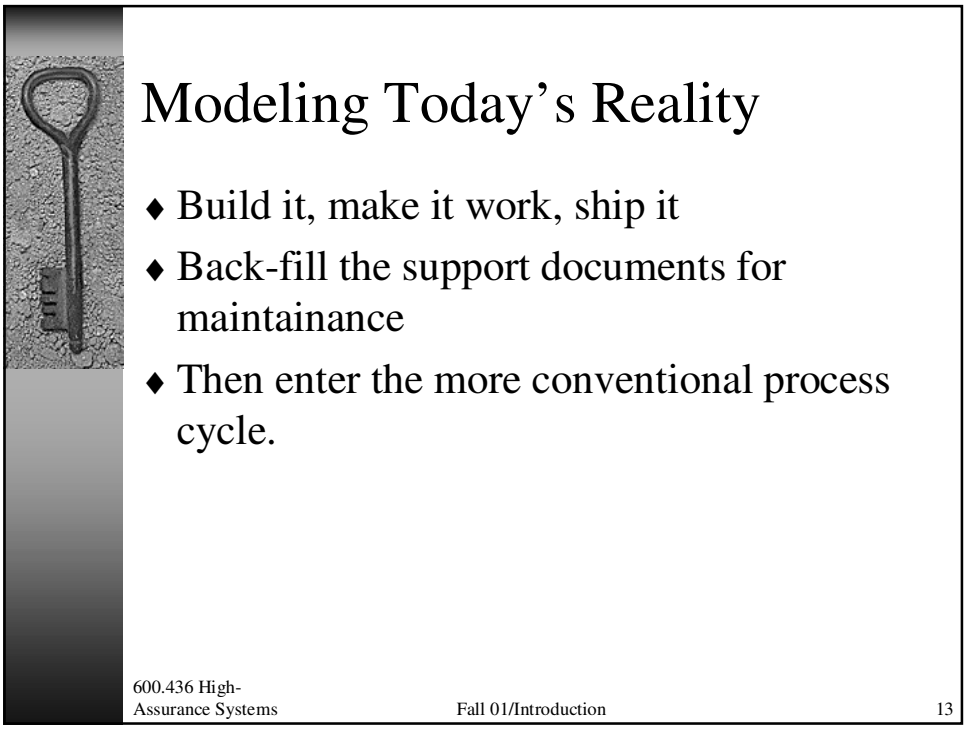
600.436 High-Assurance Systems Fall 01/Introduction 8

Successful New Companies…
♦ Understand "hit and run" marketing
♦ Are in a ***perpetual*** state of product transition
♦ Recognize that
– An acceptable product that is first to market *always* beats a good product that gets there after the first product is entrenched
– Market share is an exponential process. You don't have to be very late to be *too* late.
– Market growth is about managing *perceptions*, not realities.
♦ ***No time for long, linear processes!***
600.436 High-Assurance Systems
Fall 01/Introduction
9


So This Whole Picture…
(Cheap)
Design
Develop
Deploy
(Bankrupt)
Requirements (1x)
Rigorous Model (?)
Architecture (1ox)
Design (100x)
Implementation (1,000x)
Validation/QA (10,000x)
Delivery (100,000x)
Acceptance (1,000,000x)
Deployment (10,000,000x)
Release Feedback Cycle
600.436 High-Assurance Systems
Fall 01/Introduction
10


Is Irrelevant To Us (Today)
(Cheap)
Design
Develop
Deploy
(Bankrupt)
Rigorous Model
(?)
Design
(100x)
Implementation
(1,000x)
Delivery
(100,000x)
Acceptance
Deployment
(10,000,000x)
Cycle
600.436 High-Assurance Systems
Fall 01/Introduction
11

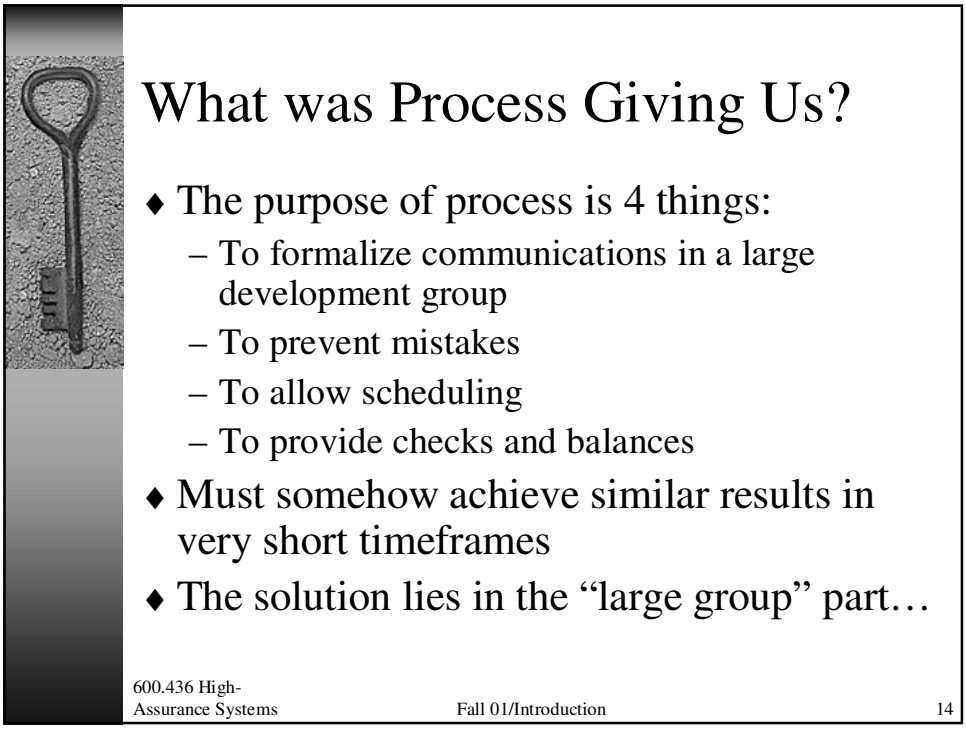

This May Change…
♦ Assurance is raising the bar for what constitutes “acceptable” software
♦ Software liability will create standards of practice
♦ Component structure makes the process investment reusable
♦ The per-component overhead is acceptable
600.436 High-Assurance Systems
Fall 01/Introduction
12



Modeling Today's Reality

- Build it, make it work, ship it
- Back-fill the support documents for maintainance
- Then enter the more conventional process cycle.

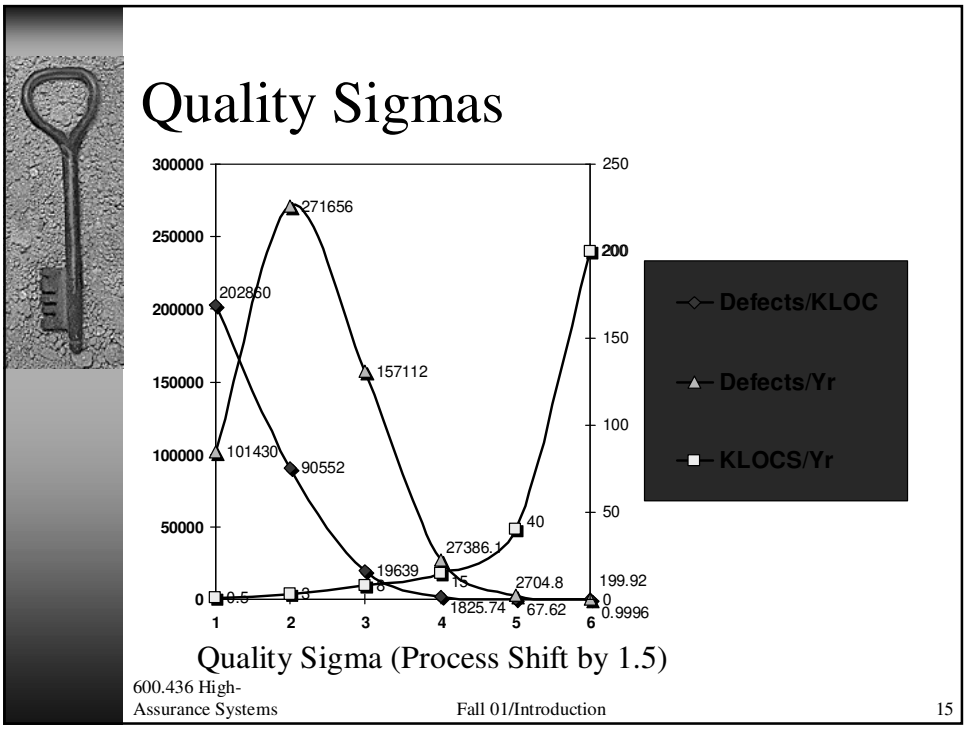
600.436 High-Assurance Systems Fall 01/Introduction 13

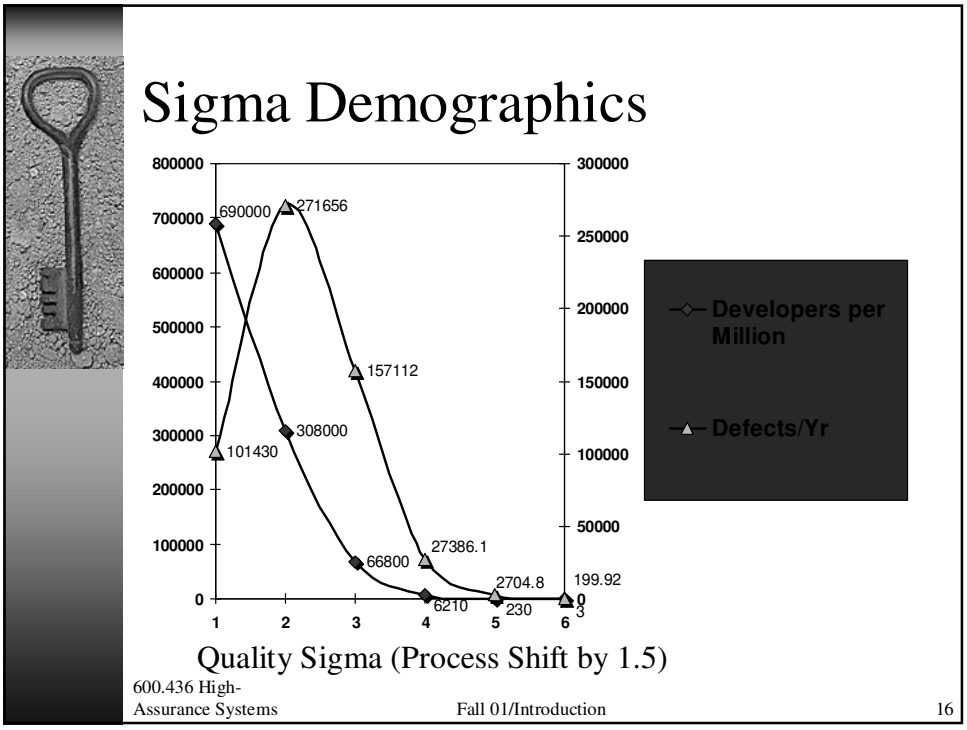


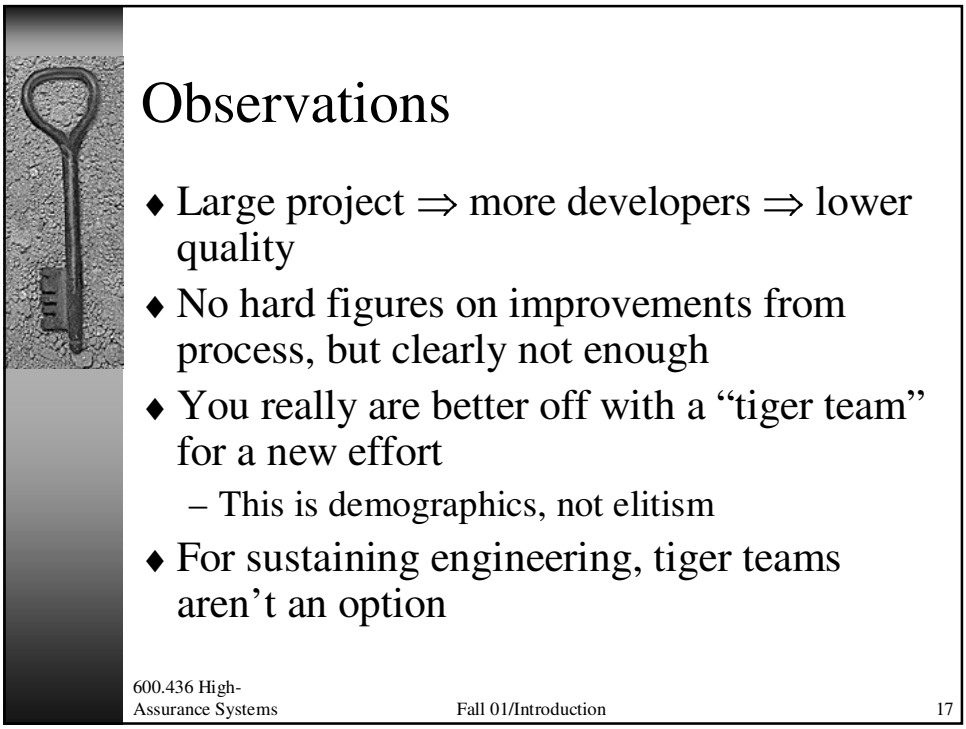
What was Process Giving Us?

- The purpose of process is 4 things:
  - To formalize communications in a large development group
  - To prevent mistakes
  - To allow scheduling
  - To provide checks and balances
- Must somehow achieve similar results in very short timeframes
- The solution lies in the "large group" part…

600.436 High-Assurance Systems Fall 01/Introduction 14


Quality Sigmas
300000
250000
200000
150000
100000
50000
0
250
200
150
100
50
271656
202860
157112
101430
90552
40
27386.1
19639
2704.8
199.92
1825.74
67.62
0.9996
1
2
3
4
5
6
Defects/KLOC
Defects/Yr
KLOCS/Yr
Quality Sigma (Process Shift by 1.5)
600.436 High-Assurance Systems
Fall 01/Introduction
15

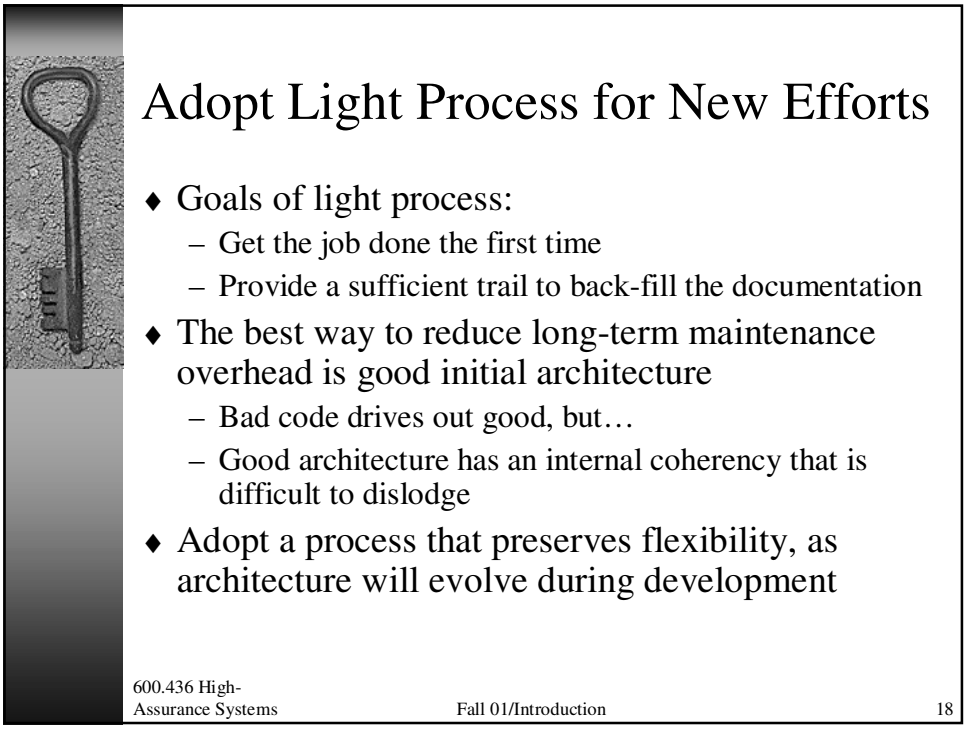

Sigma Demographics
800000
700000
600000
500000
400000
300000
200000
100000
0
300000
250000
200000
150000
100000
50000
690000
271656
157112
308000
101430
66800
27386.1
2704.8
199.92
6210
230
1
2
3
4
5
6
Developers per Million
Defects/Yr
Quality Sigma (Process Shift by 1.5)
600.436 High-Assurance Systems
Fall 01/Introduction
16



Observations

- Large project ⇒ more developers ⇒ lower quality
- No hard figures on improvements from process, but clearly not enough
- You really are better off with a "tiger team" for a new effort
  - This is demographics, not elitism
- For sustaining engineering, tiger teams aren't an option

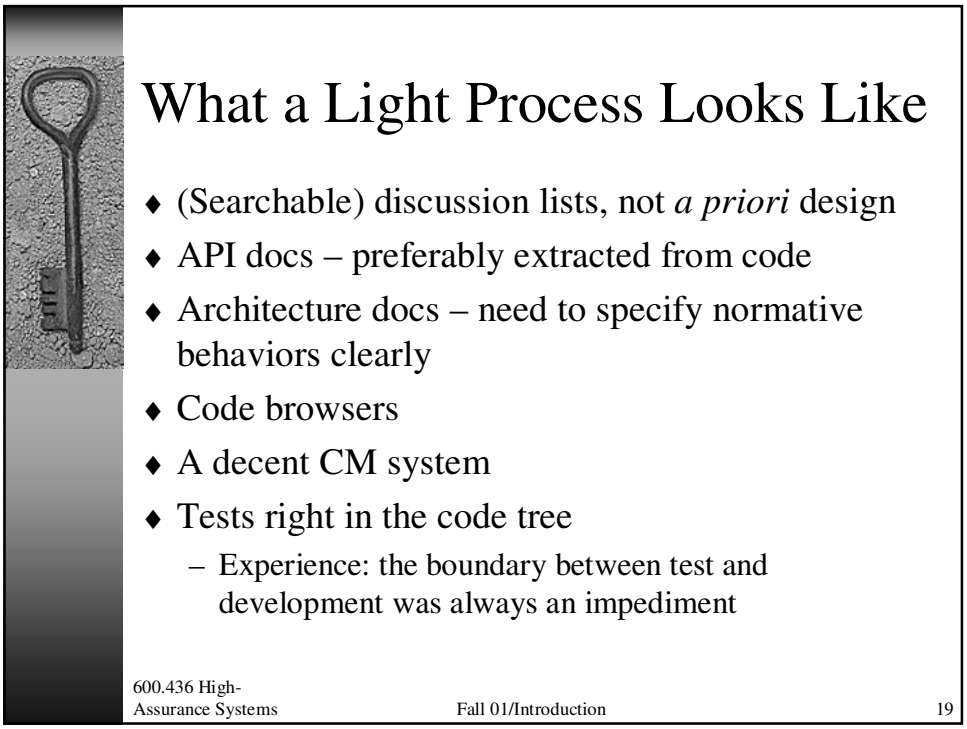
600.436 High-Assurance Systems Fall 01/Introduction 17



Adopt Light Process for New Efforts

- Goals of light process:
  - Get the job done the first time
  - Provide a sufficient trail to back-fill the documentation
- The best way to reduce long-term maintenance overhead is good initial architecture
  - Bad code drives out good, but…
  - Good architecture has an internal coherency that is difficult to dislodge
- Adopt a process that preserves flexibility, as architecture will evolve during development

600.436 High-Assurance Systems Fall 01/Introduction 18


What a Light Process Looks Like
♦ (Searchable) discussion lists, not *a priori* design
♦ API docs – preferably extracted from code
♦ Architecture docs – need to specify normative behaviors clearly
♦ Code browsers
♦ A decent CM system
♦ Tests right in the code tree
– Experience: the boundary between test and development was always an impediment
600.436 High-Assurance Systems
Fall 01/Introduction
19