1.264 Lecture 4

Software Process: CMM
Unified Modeling Language (UML)

Capability Maturity Model for Software

- Developed at Software Engineering Institute (SEI), Carnegie-Mellon University (www.sei.cmu.edu)
- De facto standard for software process assessment
- Five level model
  - 1: Initial
  - 2: Repeatable
  - 3: Defined
  - 4: Managed
  - 5: Optimized
- Predictability, effectiveness and control of software processes improve as organization moves up these levels

CMM Motivation

- 20 years of unfulfilled promises about productivity and quality gains from new software technology
- Organizations realized fundamental problem is the inability to manage the software process
- CMM provides guidance on how to evolve toward a culture of software engineering and rational management

CMM Level 1: Initial

- Ad hoc, occasionally chaotic
- Few processes defined
- Success depends on individual effort and heroics

CMM Level 2: Repeatable

- Basic project management processes established to track cost, schedule, functionality
- Discipline in place to repeat earlier successes on projects with similar applications
- Key processes focus on basic project management controls
  - Requirements management
  - Software project planning
  - Software project tracking and oversight
  - Software subcontract management
  - Software quality assurance
  - Software configuration management
- At level 2, you can measure what’s going on, and that helps understand future projects

CMM Level 3: Defined

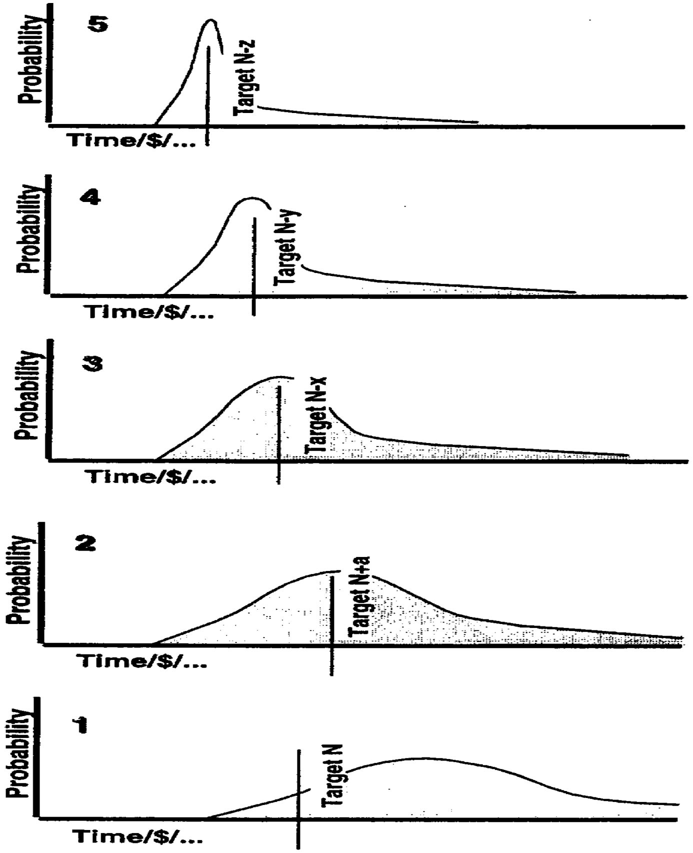
- Software process for management and development is documented, standardized and integrated into an overall process for the organization
- All projects use approved, tailored version of standard process
- Key process areas focus on institutionalizing effective process
  - Organization process focus
  - Organization process definition
  - Training program
  - Integrated software management
  - Software product engineering
  - Intergroup coordination
  - Peer reviews
- At level 3, you begin to have some control; you can actually project times/costs and make some choices

CMM Level 4: Managed

- Detailed measures of software process and product quality are collected
- Process and products are quantitatively understood and controlled
- Key processes focus on quantitative understanding of process
  - Quantitative process management
  - Software quality management
- At level 4 you have real control: you can measure and manage all aspects of the project

CMM Level 5: Optimizing

- Continuous process improvement through quantitative feedback
- Piloting innovative technology and ideas
- Key process areas focus on continual process improvement
  - Defect prevention
  - Technology change management
  - Process change management
- At level 5, you not only have control but are efficient



Performance continuously improves in Level 5 organizations

Based on quantitative understanding of process and product, performance continues to improve in Level 4 organizations

With well-defined processes, performance improves in Level 3 organizations

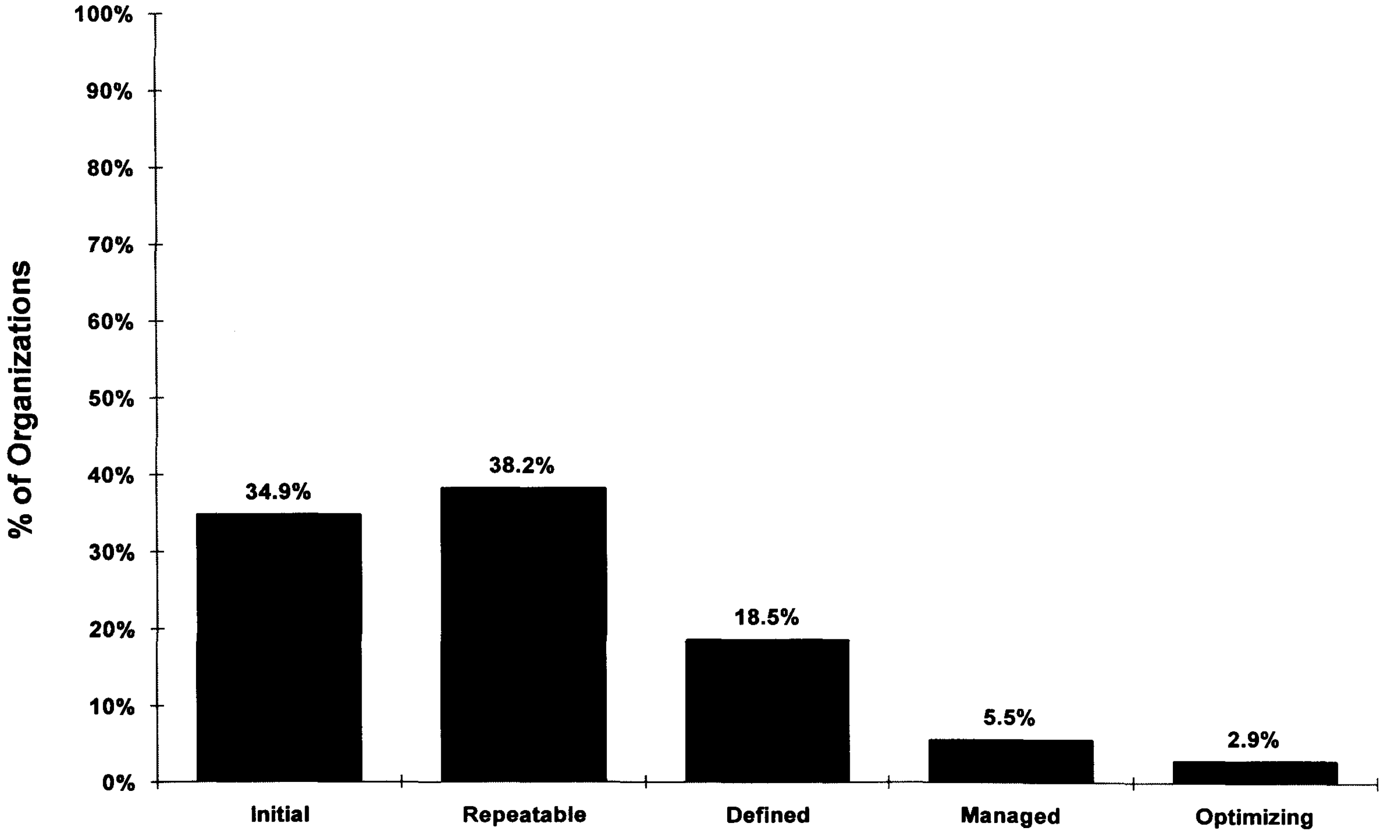
Plans based on past performance are more realistic in Level 2 organizations

Schedule and cost targets are typically overrun by Level 1 organizations.

Figure 2.4 Process Capability as Indicated by Maturity Level

Organization Maturity Profile

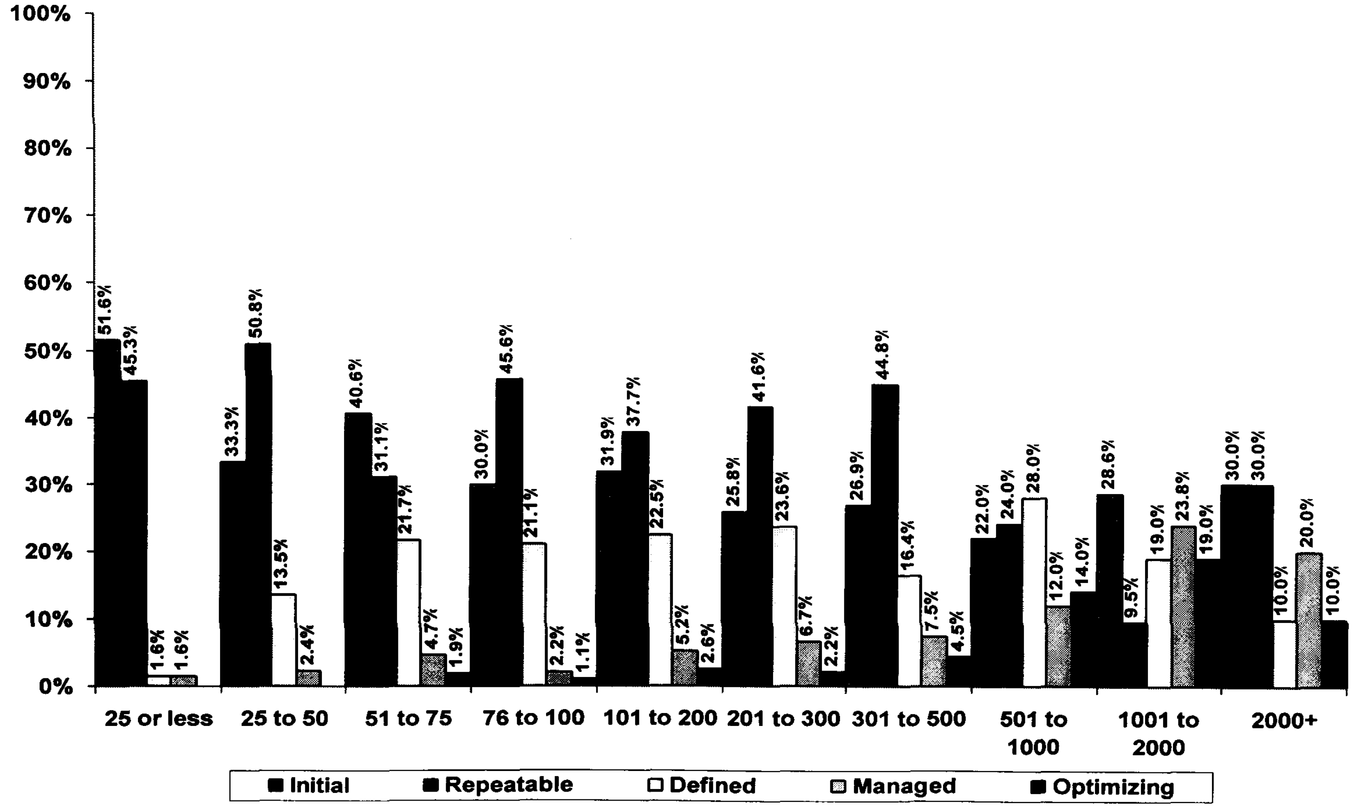
August 2000



Based on most recent assessment, since 1996, of 901 organizations. For a perspective, please see page 18.

Maturity Profile by Organization Size

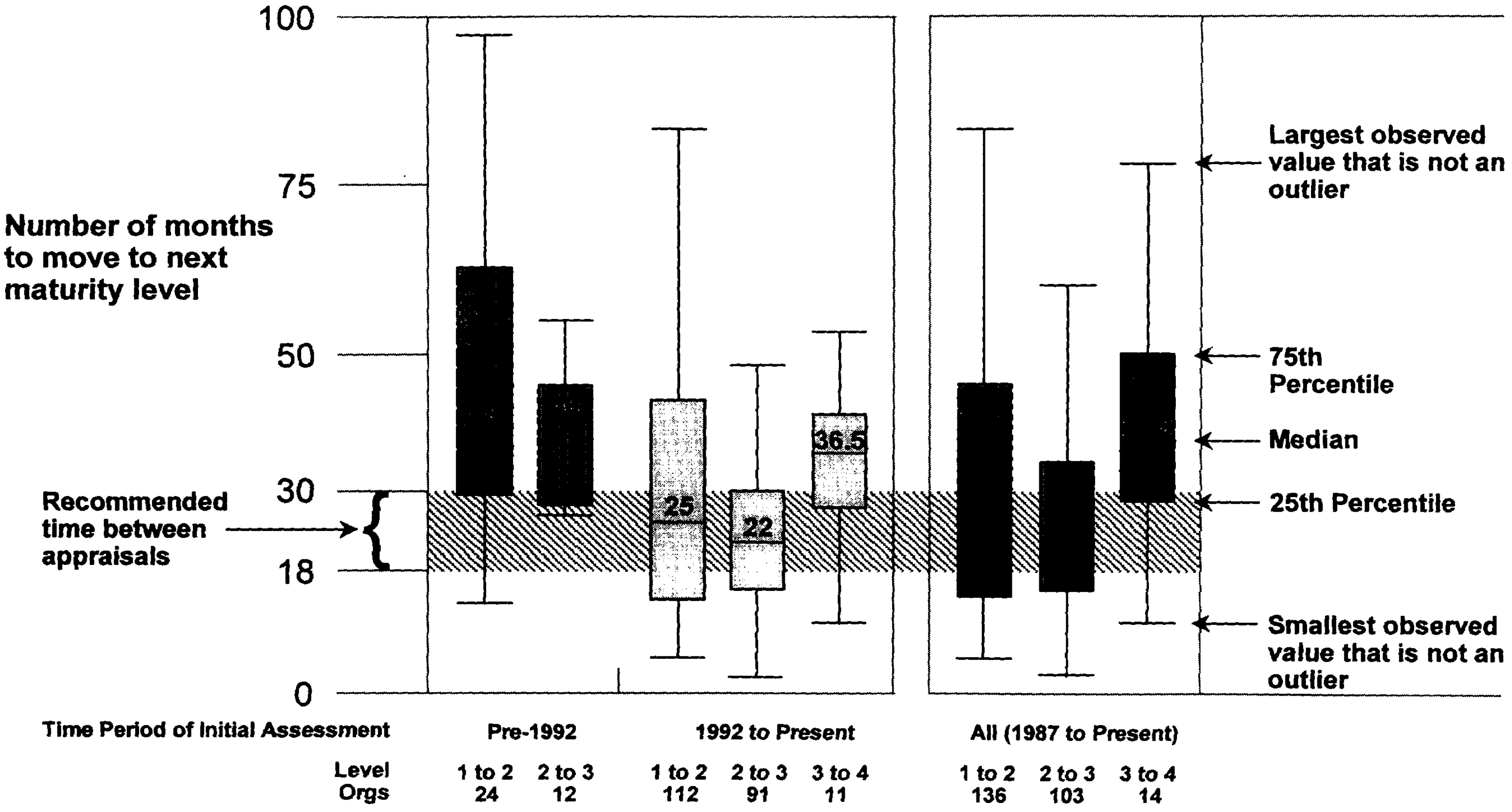
Based on the total number of employees primarily engaged in software development/maintenance in the assessed organization



The 1001 to 2000 and 2000+ categories are of a small percentage which will inflate the maturity level bars. Please see page 9 and take this into account. The purpose of this chart is to indicate that all size categories contain most, if not all, maturity levels.

Based on 814 organizations reporting size data

Time to Move Up



ISO 9000

- International Standards Organization (ISO)
  - National standards bodies from 100+ countries
- ISO 9000
  - Family of QA standards for framework, models, specifications for quality management systems
  - Best practices applied to production
- ISO 9001
  - QA standards for design, development and service organizations
  - Guidelines for software are ISO 9000-3
  - 20 requirements must be met
  - Documenting and standardizing processes to develop end product
  - ISO 9001 does not standardize products

ISO 9000 cont

- ISO 9000 certification mandatory to do business in Europe, and becoming so in Pacific Rim and eventually Americas
- General approach
  - Develop a quality team
  - Say what you do: document processes, usually via flow diagrams
  - Do what you say
  - Prove it: annual audits
- After completing these four tasks, independent firm audits and grants certification
- Often, no real change results from ISO 9000

Unified Modeling Language

- Object-oriented modeling language, migrated from relational database modeling
  - Standard managed by Object Mgt Group (CORBA)
  - Rational Rose, from Rational Corp, is a common implementation of UML. Many other vendors available now.
- Combines previously competing approaches
  - Rumbaugh Object Modeling Technique (OMT)
  - Shlaer-Mellor method
  - Booch method
- Modest level of use currently, becoming common

Unified Modeling Language, p.2

- Why is UML coming into wider use?
  - Speed up requirements process
  - Lessen information loss between requirements and design processes, and between design and implementation
  - Communication: clearer than natural language, provides a level of precision, but avoids details
  - Supports iterative development (i.e., spiral model)
    - Supports both high level requirements/design in early spirals and detailed requirements/design later
- UML is just the modeling language
  - Rational Unified Process (RUP) is a recommended process, based on using UML
    - Inception (requirements)
    - Elaboration (design)
    - Construction (development): “extreme programming” fashionable
    - Transition (test, implementation)

Unified Modeling Language, p.3

- Used in requirements:
  - Deployment diagram, component diagram to show high level view of system
  - Use cases, which are very structured scenarios used to define system requirements
    - Good basic approach, but needs narrative to support
- Used in design:
  - Data models (not strictly part of UML) are done in conjunction with class (object) models and correspond closely
    - Often done area by area and then consolidated
  - Activity diagrams, used to model workflows, to find duplicate processes that can be eliminated
  - Prototyping used for risky, critical, difficult parts of system

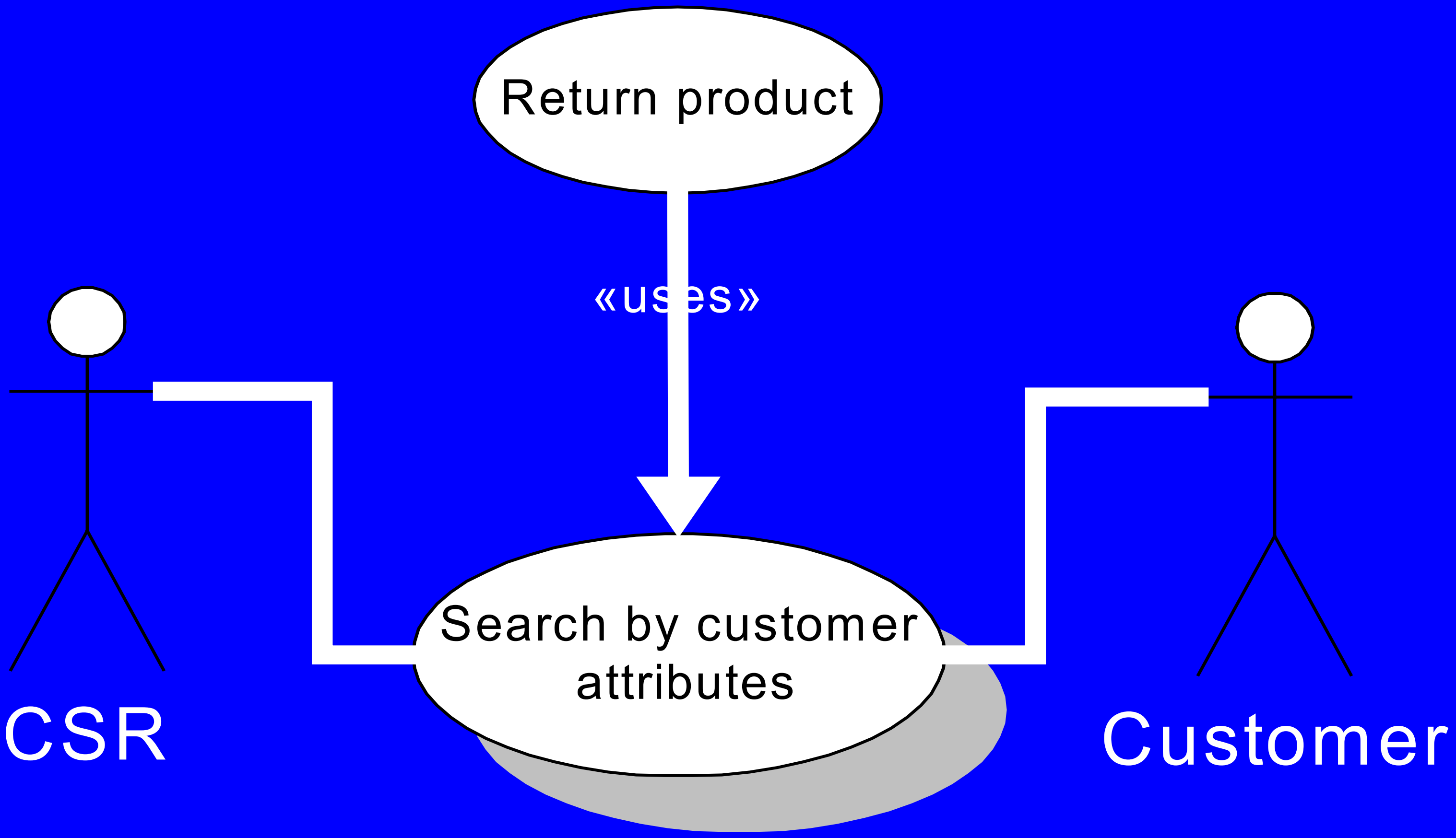
UML static model diagrams

- Use case diagram
  - Drawings and structured descriptions of steps in workflows
- Class diagram
  - Internal structure of system, extension of entity-relationship diagram.
  - Three elements in each entity: name, attributes, methods
- Deployment diagram
  - Physical components: processors, workstations, network
- Package or Component diagram
  - High level model of physical software architecture
  - Consists of modules, which are grouped in packages
  - Packages contains definition of group of classes (entities, methods)

Use Case Example

Use Case Name	Locate Orders and Products Using Order Number and Zip
Summary	Allow user to locate their customer and order info for the items to be returned by entering an order number (usually located on the invoice) and zip code.
Iteration	Filled
Events	1. System displays search screen 2. User enters search criteria. The search criteria in this case are: a. Order Number (usually located on the invoice) b. Billing Zip Code 3. System retrieves and displays customer's order and order details
Exception Paths	1. If no search results were located then: a. The system will display an error message that this information was not found b. The system will redisplay the search using order number and zip code screen c. The user will enter the search attributes d. The system retrieves and displays customer's order and order details 2. If the customer's search fails three times consecutively then: a. The system will display an error message apologizing for not finding the order and suggesting that the user call customer service b. The user can select to try the search again or go elsewhere on the site
Trigger	Customer has logged in or called in, is identified and chooses 'new return' option
Assumptions	Orders and product data are correct and current through previous day. Customer data is current through previous day. (All must be real-time for in-store returns)
Preconditions	1. User has selected this command from the retailer's customer service page 2. The user has purchased a product from this retailer ***3.*** The user has the order # (from e-mail receipts, the invoice, or other means)
Postconditions	1. The system located and displayed the order and order details (products)
Related Business Rules	1. Note an annoyance: zip codes change. Check old and new during transition period 2. What if a customer wants to return items from two orders? Ok; allow multiple selection.
Notes	--
History	Todd Clarke – 3/15/01 - Façade iteration Todd Clarke – 3/16/01 - Filled iteration

Use Case Example, p.2

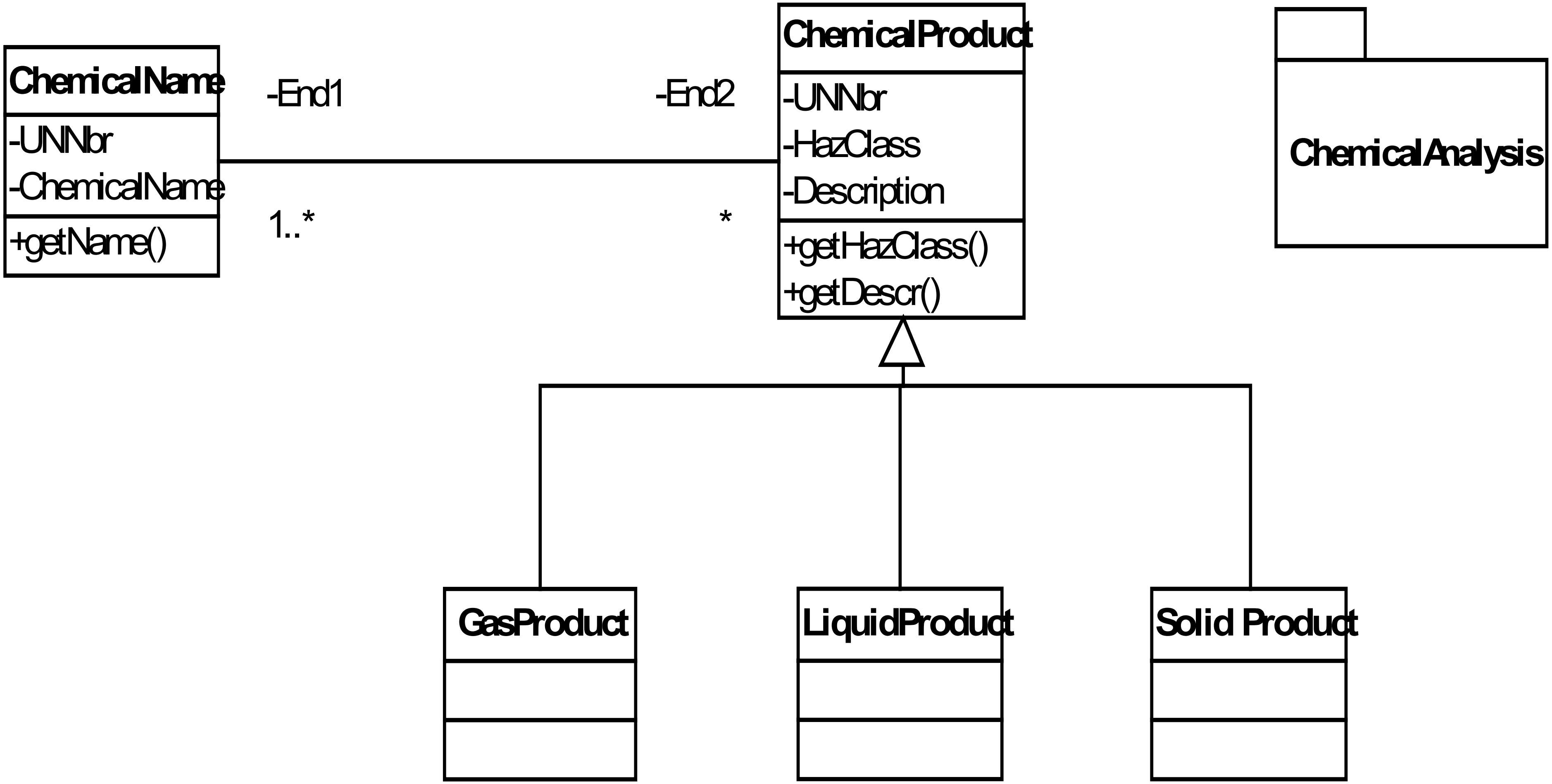


Actors (stick figures): CSR, Customer
Use Cases (ovals): Search, return
Relationship(line, arrow)

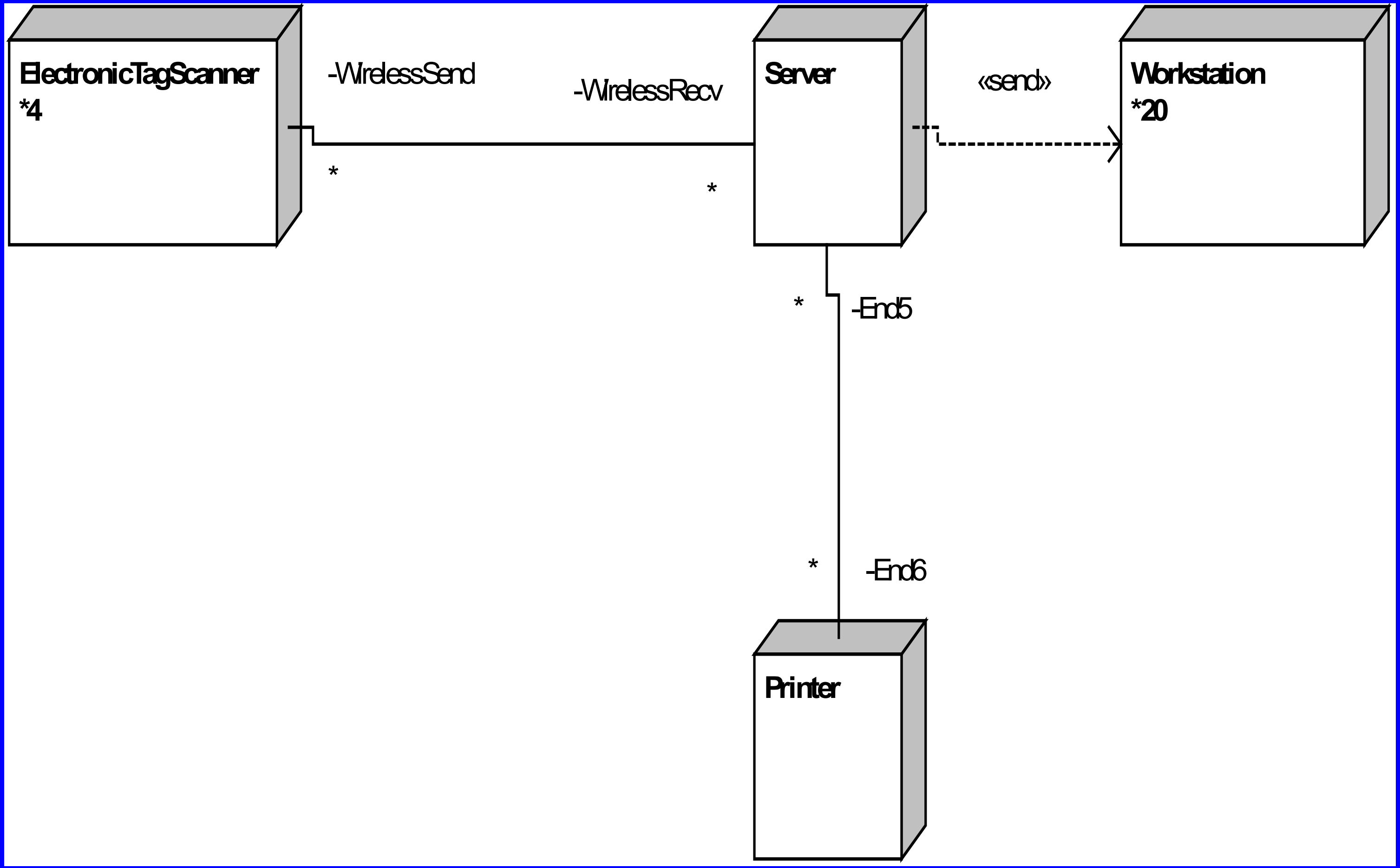
Class Diagrams

- Used in requirements, design and implementation:
  - Conceptual, to represent general entities in system
  - Specification, where we specify what each entity (class) will do (but not how)
    - List the methods/actions
  - Implementation
    - Detailed class diagram of actual software (Java or C++)
- List attributes, same as data model
- List methods/operations/functions
  - Activities naturally associated with the data in the entity
- Also model constraints, preconditions, postconditions, etc. that are laid out in the use cases
- We often don’t model everything—too hard to read
  - Focus on key parts of system

Class Diagram



Deployment diagram

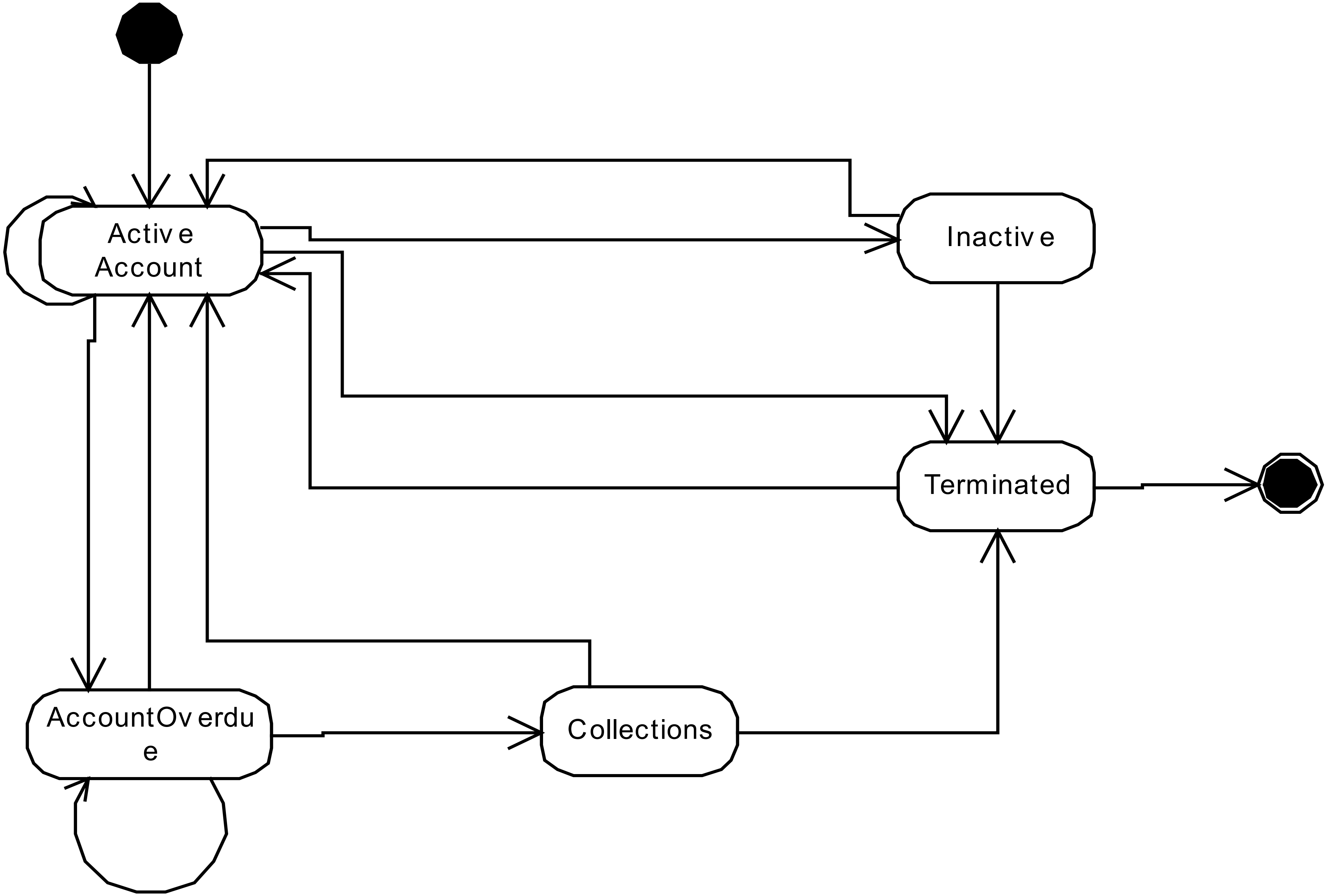


Package diagram similar but more detailed

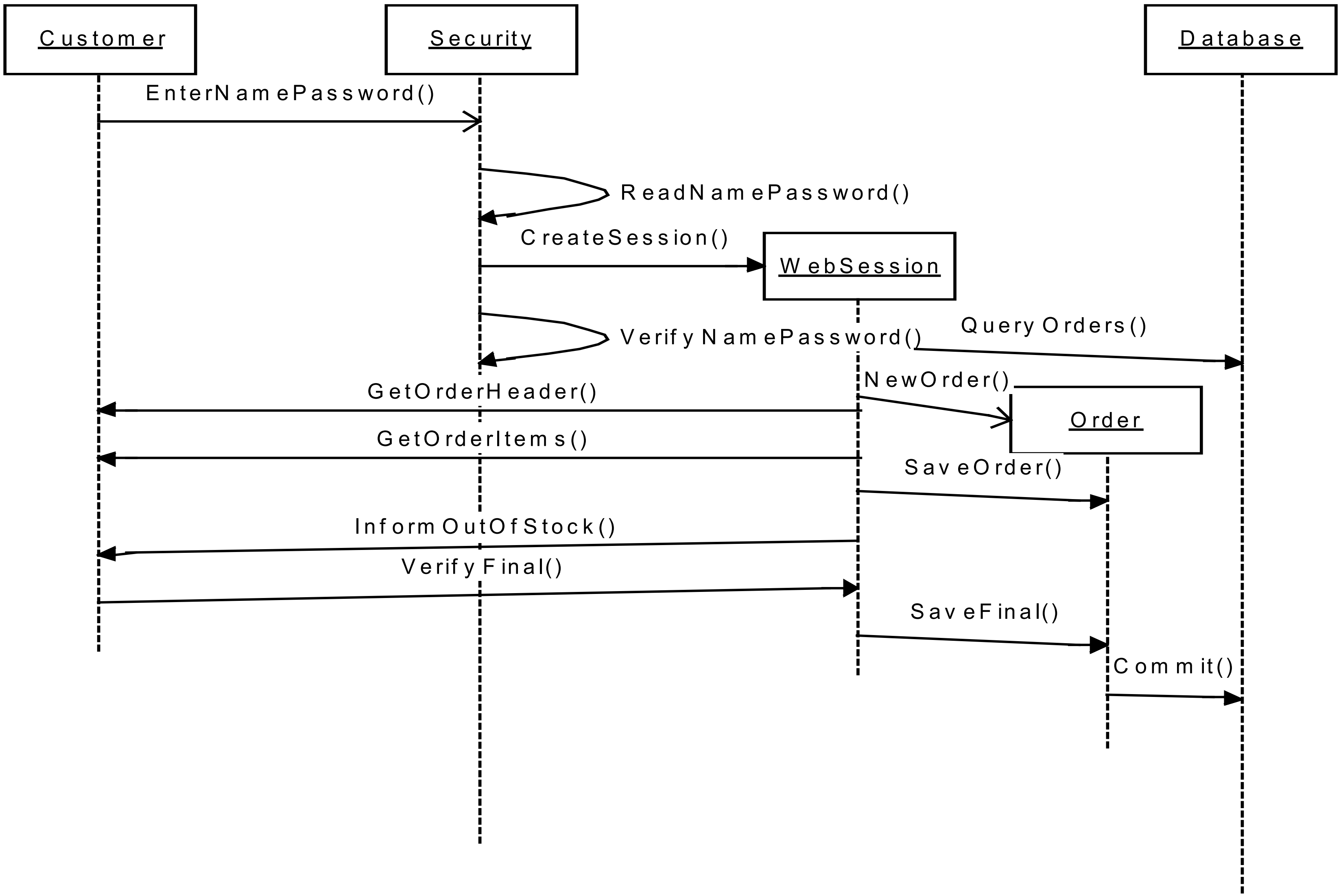
Dynamic models

- While static models are done for the system as a whole, dynamic models are done only for key components
- State diagram
  - Specifies behavior of an object (entity)
- Sequence diagram (or ladder diagram)
  - Shows details of scenario and messages that flow between objects over time
  - Heavily used in standards
- Collaboration diagram
  - Shows flow of messages as a graph.

State diagram



Sequence diagram



UML Summary

- Visio demo (Software->UML Model has all types)
- Use UML after:
  - Writing scenarios and narratives as an initial requirements document
    - Refine them into use cases
  - Preparing the initial data model
    - Add operations/methods to the entities, after understanding the data, to create a class diagram
- Use UML package and component models to give overview of the system, in requirements
- Use UML state, collaboration, sequence models selectively in complex parts of the system
- UML is becoming a ‘universal’ language: new staff coming to a project can read it, and this reduces the learning curve very substantially