CSC 408F/CSC2105F Lecture Notes

These lecture notes are provided for the personal use of students taking CSC 408H/CSC 2105H in the Fall term 2004/2005 at the University of Toronto.

Phase A. Requirements Analysis

- Pick 4 to 5 partners to form a project team
- Describe the people in the team, allocation of tasks
- Pick one editor (VIM or Eclipse) to study, document the discussions why and how you choose the editor
- Discover why we shall use Web Services to bridge hetergeneous editors
- Describe the architecture of the OmniEditor project
- Plan the project as an iterative process and proposes a test plan

Deliverables after Phase A by Oct. 7

- Documents[a]
  - 30% marks: Requirements specification of the OmniEditor
  - 20%: Understanding of legacy editor: identify reusable components
  - 20%: Test plan of the project
  - 10%: Project plan of the OmniEditor
  - 10%: Risk analysis of the project
  - 10%: Team organization of the project
- Project team
  - Who are the team members and Who is the team leader
  - Skills and Preferences
  - Tasks and allocations
  - Team meeting schedules

[a] No lower or upper bounds to the document length

Requirements for requirements specifications

Reference to John Mylopoulos's CSC340S course:

http://www.cs.toronto.edu/∼jm/340S/

- A list of functional (behavioral) requirements
  - Goal of the requirement
  - Inputs, Outputs
  - Preconditions on Input, Postconditions on Output, Exceptions
- A list of non-functional (quality) requirements
  - Softgoal of the quality attribute
  - Metric of the attribute
  - Satisfaction criteria

Goal-oriented requirements engineering (GORE)

Goal: the desired state, often described in terms of predicate

GORE: identify logical dependency relationships between requirements (goal).

- Abstraction/Refinements: asking Why and How
- Completeness (obstacle analysis): asking What-if?
- Contributions and Correlations: asking How much?
- Domain analysis: asking who, what and where?

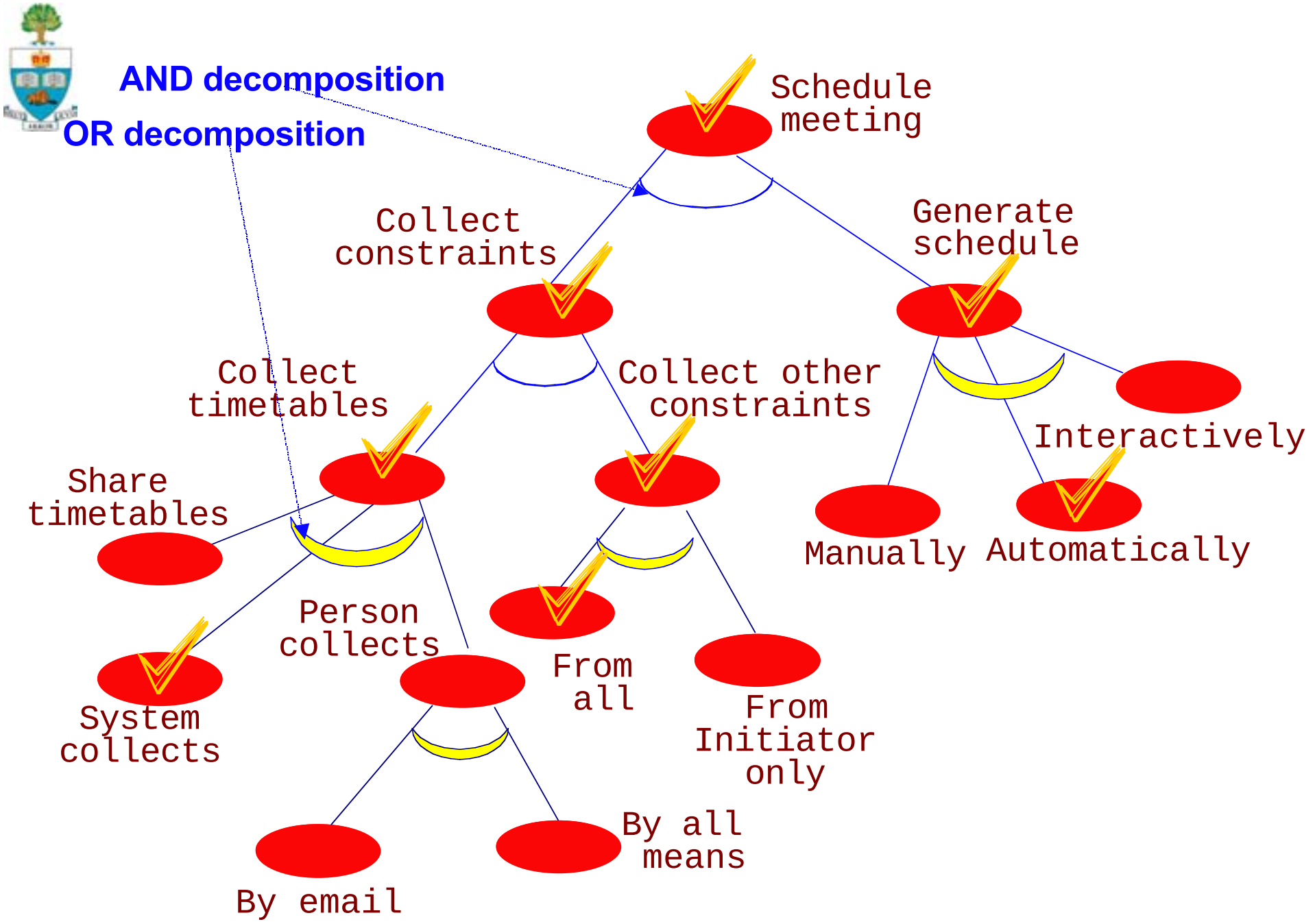
Use natural languages, Req. Spec. do not include project requirements, designs and quality assurance plans.

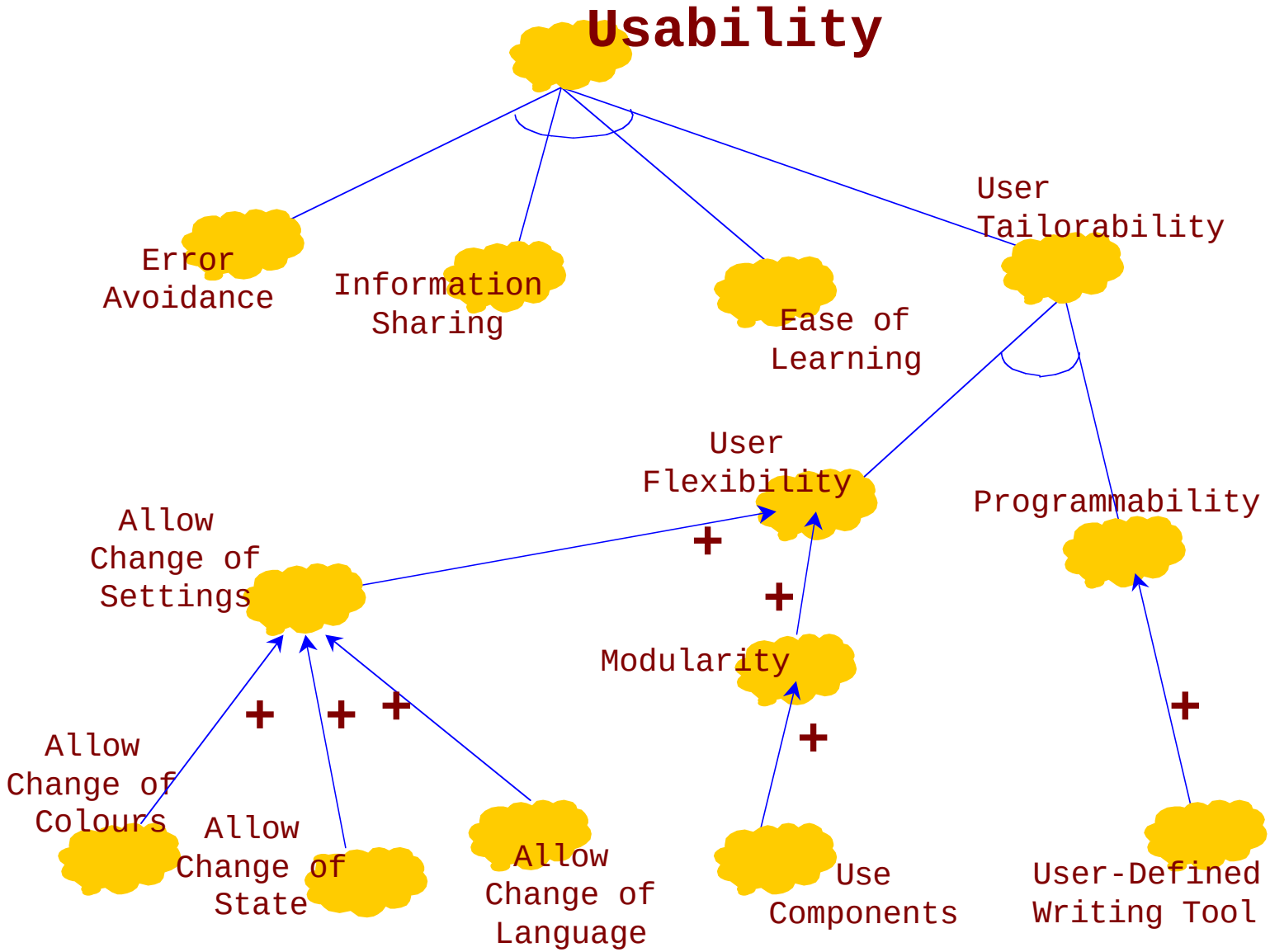
To edit the goal dependencies, try this tool for Windows:

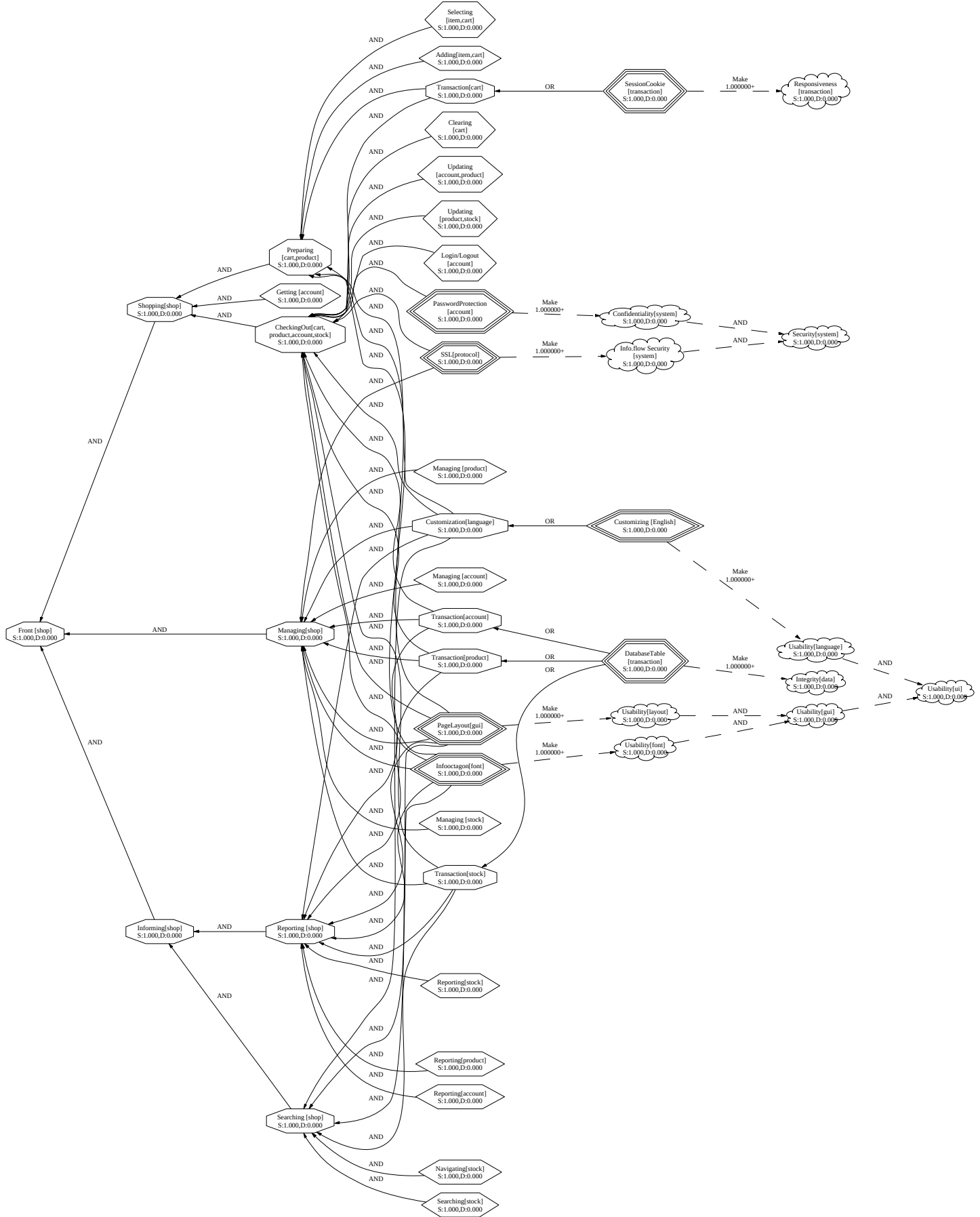
http://www.cs.toronto.edu/∼yijun/OpenOME.html

An example requirements goal model, see

http://www.cs.toronto.edu/∼yijun/slides/yu04re.ppt


AND decomposition
OR decomposition
Schedule meeting
Collect constraints
Generate schedule
Collect timetables
Collect other constraints
Interactively
Share timetables
Manually
Automatically
Person collects
From all
System collects
From Initiator only
By all means
By email


Usability
Error Avoidance
Information Sharing
Ease of Learning
User Tailorability
User Flexibility
Programmability
Allow Change of Settings
Modularity
Allow Change of Colours
Allow Change of State
Allow Change of Language
Use Components
User-Defined Writing Tool
+
+
+
+
+
+
+


Selecting [item,cart] S:1.000,D:0.000
Adding[item,cart] S:1.000,D:0.000
Transaction[cart] S:1.000,D:0.000
Clearing [cart] S:1.000,D:0.000
Updating [account,product] S:1.000,D:0.000
Updating [product,stock] S:1.000,D:0.000
Login/Logout [account] S:1.000,D:0.000
PasswordProtection [account] S:1.000,D:0.000
SSL[protocol] S:1.000,D:0.000
Preparing [cart,product] S:1.000,D:0.000
Getting [account] S:1.000,D:0.000
CheckingOut[cart, product,account,stock] S:1.000,D:0.000
Shopping[shop] S:1.000,D:0.000
Front [shop] S:1.000,D:0.000
Managing [product] S:1.000,D:0.000
Customization[language] S:1.000,D:0.000
Managing [account] S:1.000,D:0.000
Transaction[account] S:1.000,D:0.000
Transaction[product] S:1.000,D:0.000
PageLayout[gui] S:1.000,D:0.000
Infooctagon[font] S:1.000,D:0.000
Managing[shop] S:1.000,D:0.000
Managing [stock] S:1.000,D:0.000
Transaction[stock] S:1.000,D:0.000
Informing[shop] S:1.000,D:0.000
Reporting [shop] S:1.000,D:0.000
Reporting[stock] S:1.000,D:0.000
Reporting[product] S:1.000,D:0.000
Reporting[account] S:1.000,D:0.000
Searching [shop] S:1.000,D:0.000
Navigating[stock] S:1.000,D:0.000
Searching[stock] S:1.000,D:0.000
SessionCookie [transaction] S:1.000,D:0.000
Responsiveness [transaction] S:1.000,D:0.000
Confidentiality[system] S:1.000,D:0.000
Info.flow Security [system] S:1.000,D:0.000
Security[system] S:1.000,D:0.000
Customizing [English] S:1.000,D:0.000
DatabaseTable [transaction] S:1.000,D:0.000
Usability[language] S:1.000,D:0.000
Integrity[data] S:1.000,D:0.000
Usability[layout] S:1.000,D:0.000
Usability[font] S:1.000,D:0.000
Usability[gui] S:1.000,D:0.000
Usability[ui] S:1.000,D:0.000
AND
OR
Make 1.000000+

Example of requirements specifications

- A functional requirement
  - Goal: query [stock price]
  - Inputs: stock quote [string]
  - Outputs: stock price [float]
  - Precondition: stock quote is not empty
  - Postcondition: stock price ≥ 0 if stock quote is found
    stock price $= -1$ if stock quote is not found
- An alternative requirement
  - Inputs: stock name [string]
  - Precondition: stock name is not empty
  - Postcondition: stock price ≥ 0 if stock name is found and unique
    stock price $= -1$ if stock name is not unique
    stock price $= -2$ if stock name is not found

- A non-functional requirement
  - Softgoal: responsiveness [query]
  - Metric: response time
  - Satisfaction criteria: response time $<$ 1 second

Goal Elicitation

- Why I need this requirement? To get return of investment.
- How is this requirement done? Use web service.
- How much can this requirement be achieved? 100%.
- Who, What, Where? A web service deployed in U of T CDF machine run by the user.

Testing Plan

- How to test functional requirements
  - Test cases
  - Given input, expected output
  - Given unexpected input, expected error handling
  - Test Coverage
- How to test non-functional requirements
  - How to measure the metrics directly?
  - Qualitative vs. Quantitative measurements?
  - How to measure the metrics indirectly? E.g., logical reasonings on the goal dependencies
  - How to guarantee the statistical significance? E.g., average, errors and variations.

http://www.cs.toronto.edu/∼yijun/csc408h/handouts/testing.pdf

Understanding Legacy Editor

- Describe its (non-)functionalities (features) related to your task;
- Understand its architecture (components and connections);
- Install and configure it to make sure you can use it;
- Compare similar systems and give a detailed selection criteria in terms of (1) features and (2) reusability for your task.

Project plan, Risk control, Team organization

follows ...

Reading assignments Chapter 2, 3 and 8

Software Engineering Management

- Software development projects require active management and planning if they are to be successful.
- Many software project failures can be attributed to inadequate or faulty project planning and management.
- The managers of a software project must be aware of the larger context in which the software will be used. van Vliet calls this **information planning**.
- A *Project Plan* is a detailed description of *all* the activities that will take place during the project.
- The Project Plan is used for resource allocation, scheduling and to track progress during the project.

Elements of a Software Project Plan

Introduction Context for the project including background, history, aims, deliverables, key personnel, summary.

Process Model A description of the *process model* that will be used for the project. May include activities, milestones, deadlines, identification of critical paths.

Project Organization Relationship of the to the larger organization. Identification of project personnel and how they are organized into teams.

Standards, Guidelines, Procedures Identify software development standards and guidelines to be used. Specify software development procedures.

Management Activities A description of what project management will be doing during the project. Usually includes resource allocation, monitoring, status reporting.

Risks Description of potential risks that might affect the success of the project. Plans for dealing with these risks.

Staffing Plan for allocating personnel to the project including personnel schedule and cost estimates

Methods and Techniques Description of the tools and techniques that will be used during design, implementation and testing. For example, CASE design tools, version control tool, testing tools.

Quality Assurance Description of the steps that will be taken to assure the *quality* of the project software.

Work Packages A detailed list of the individual activities to be undertaking during the project. May include personnel allocation, schedule and constraint information.

Resources Plan for allocating computers and other physical resources required during the project. Usually includes schedule and costs.

Budget & Schedule Detailed budget for the project including allocation of costs to various activities. Detailed schedule for all of the project activities.

Change Management Plan for dealing with changes (e.g. change in requirements) that may (*will*) occur *during* the project.

Delivery Plan for delivery of the software to the user. May include interim deliveries of partial systems.

Software Engineering Processes

- A software engineering *process* is the collection of major activities that take place during the lifetime of a software system.

  Examples: Requirements, Design, Testing, Maintenance

- We study the software engineering process in order to
  - understand how the software is being built
  - find ways to improve the quality of the software
  - find ways to build the software more efficiently
  - reduce the overall *life cycle costs* of the software.

Some Software Engineering Processes

- The Wild West Approach
- Waterfall Model
- Prototyping
- Incremental Development
- Rapid Application Development
- Spiral Model

In practice an organization may use some combination of the processes listed above.

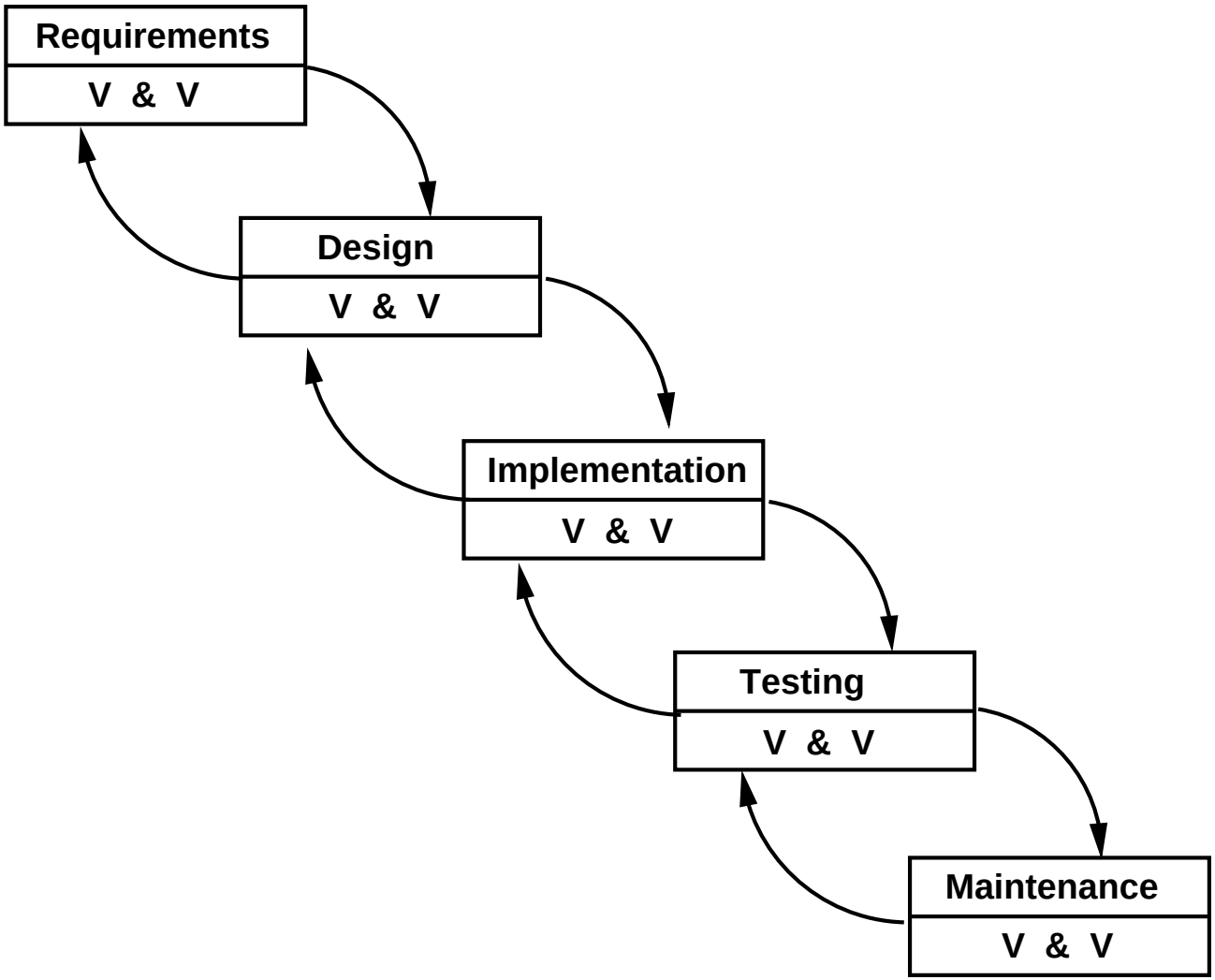
Verification and Validation

- Throughout the development of a software system
  - We want to be sure we are building a system that satisfies its requirements.
  - We want to be sure that we are building a system that meets the users requirements.
- **Verification** determines if the system being built satisfies its requirements,

  are we building the system right?
- **Validation** determines if the system meets the user's requirements.

  are we building the right system?
- Validation and Verification (**V & V**) are used to make sure that the transition from one step in the software process to the next one will be successful
- Proactive V & V is one good technique to help improve the quality of the software.

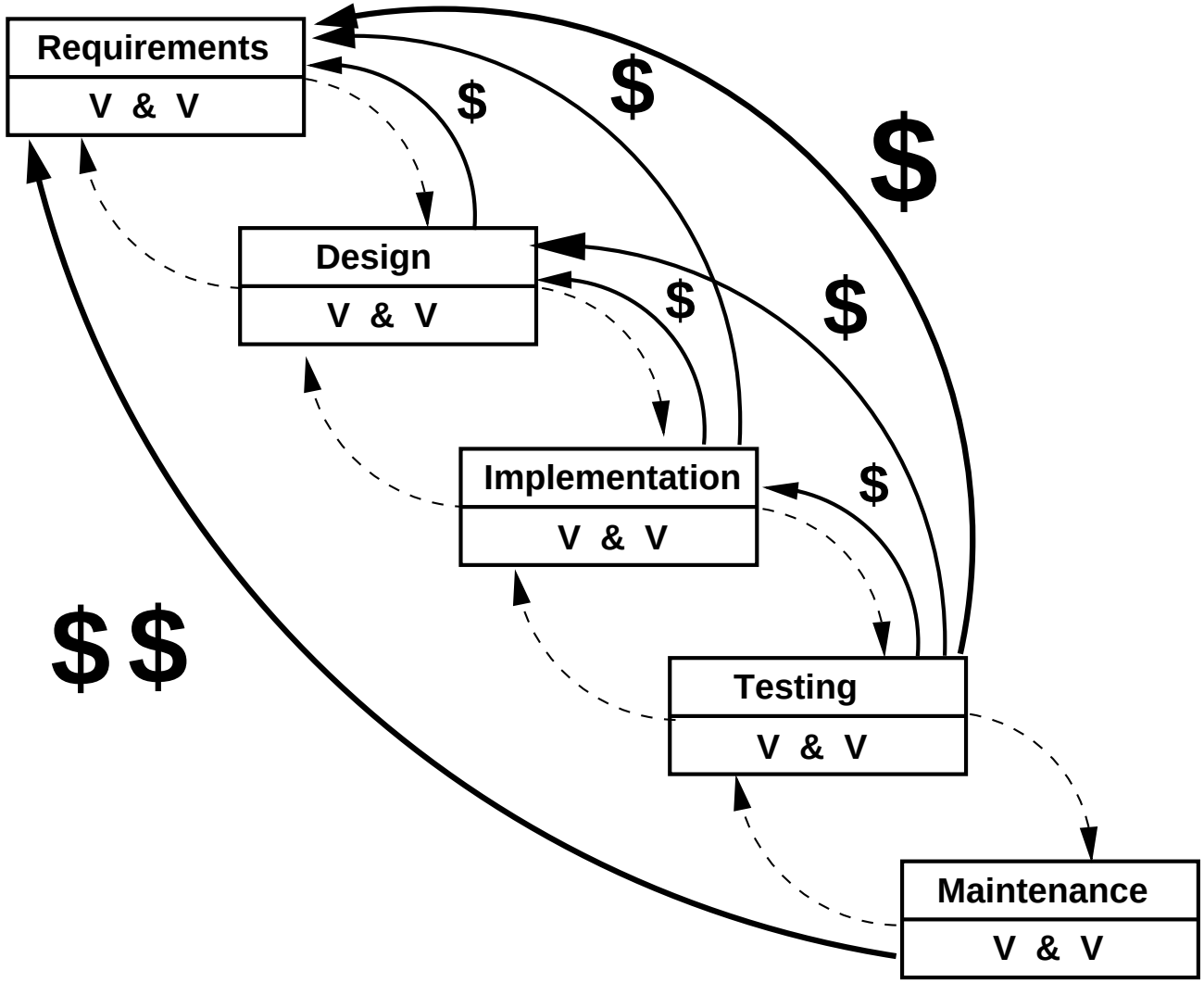
Classical Waterfall Model



Why the Classic Waterfall Model is Unrealistic

- Ignores feedback between phases
- Ignores iteration between phases
- Doesn’t allow for overlapped/phased development
- Unrealistic to expect perfect set of requirements at the start
- User must wait a long time before a working version is available
- Doesn’t accommodate uncertainty well
- Pure Waterfall approach is uneconomic except for organizations where cost is not an issue (e.g governments and public utilities)
- But it does identify the major elements of the software creation process

Real Waterfall Model



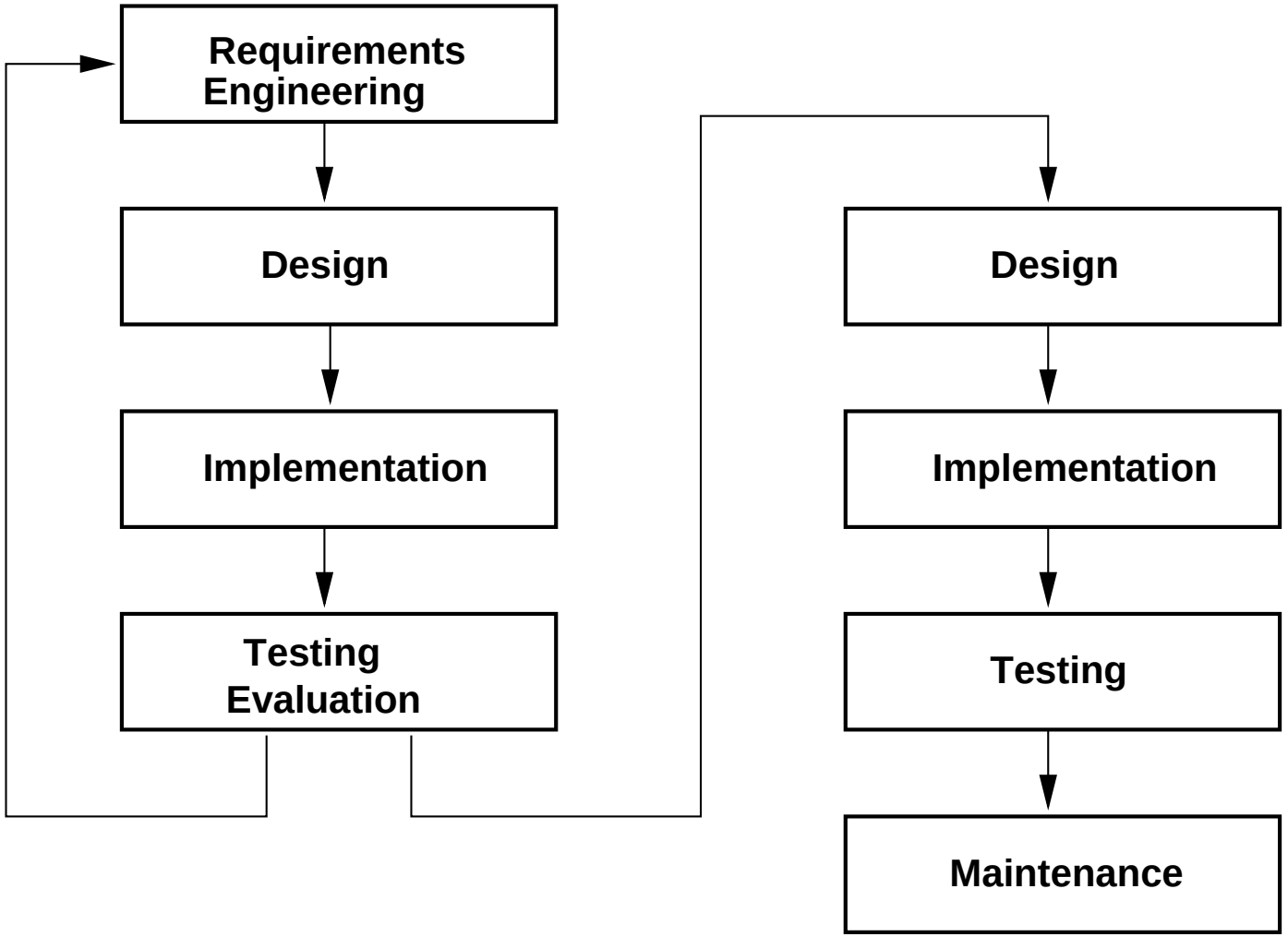
Prototyping

- A process used when general requirements for the software are known but precise details are unknown or unknowable
- Used when there is great uncertainty in how to build the system
- Allows experimentation with alternative user interfaces
- Allows user to get a feel for the system before a large amount of money is committed.
- Often used for systems with a large or complicated *User Interface*
- Allows user and software producer to reduce the risk inherent in a speculative project
- Prototyping is a tool for *Requirements Engineering*

Prototyping Process

- The key element of the prototyping process is that an initial version of the system is built quickly and inexpensively.
- The prototype system is evaluated for suitability and utility.
- The evaluation of the prototype is used to modify and refine the requirements for the system.
- The prototype process may be iterated several times until the user and software producer are satisfied with the quality and completeness of the system requirements.

Prototyping Process



Prototyping Techniques

- Construction of the prototype **must** be economical in terms of time and effort or the process will be wasted.
- Implementation techniques
  - Use very high level application development languages.
  - Initially at least omit hard/expensive features.
  - Initially use limited capacity data storage.
  - Use software tools to simulate (fake) parts of the system.
  - Sacrifice speed, robustness and maintainability to achieve fast implementation.
- Instrument the prototype if necessary to collect evaluation statistics, e.g. system *hot spots*

Prototyping Styles

- Throwaway Prototyping
  - After the system requirements have been finalized, **discard** the prototype.
  - Design and implement the system from the requirements using best software engineering techniques.
  - Customer must be made to understand that the prototype is not intended to serve as a production system.
  - Cost of discarding a working prototype and the effort that went into its development is often hard to bear.
- Evolutionary Prototyping
  - Use the last prototype as the starting point for the design of the production system.
  - Evolutionary prototyping must be planned for from the beginning.
    Requires much more effort on software architecture, maintainability and quality issues during prototype development.
  - Need to ensure during prototype design and implementation that the cost of converting the prototype to a production system will be less than the cost of developing the production system from the requirements.

When to Use Prototyping

- Systems where the user requirements are unclear, ambiguous or incomplete. Prototyping helps clarify the requirements.
- Systems with a major user interface and/or a lot of user interaction.
- Prototyping is a form a *risk sharing* between the user and the developer.
- Prototyping requires a strong commitment from the user(s) to evaluate and improve the prototype.

How to Manage Prototyping

- Users must be made aware of potential problems.
  - Major revisions to the prototype require substantial effort.
  - Prototype is **not** a production system.
  - Prototype process may lead to lower quality, less maintainable, less well documented software.
- Designers must learn to cope with rapid, contradictory changes in requirements. Be prepared to rewrite and revise the system.
- Prototyping must be managed
  - Impose limits on number of iterations.
  - Require version control and configuration management to keep track of the prototype as it evolves.
  - Establish firm procedures for documentation and testing of prototypes.

Incremental Development

- Incremental Development is a process similar to prototyping.
- Functionality of the system is delivered to the customer in small increments
  - Each increment is developed using the Waterfall model.
  - Gets parts of the system to the user early.
  - User is involved in planning for each increment.
  - Start with essential features to get initial system working
- Good approach for avoiding excess functionality (over-ambitious requirements).
- Tracking of project progress is easier.
- Tends to reduce chance of major error/mis-conception.
- Need good planning of increment sequence.

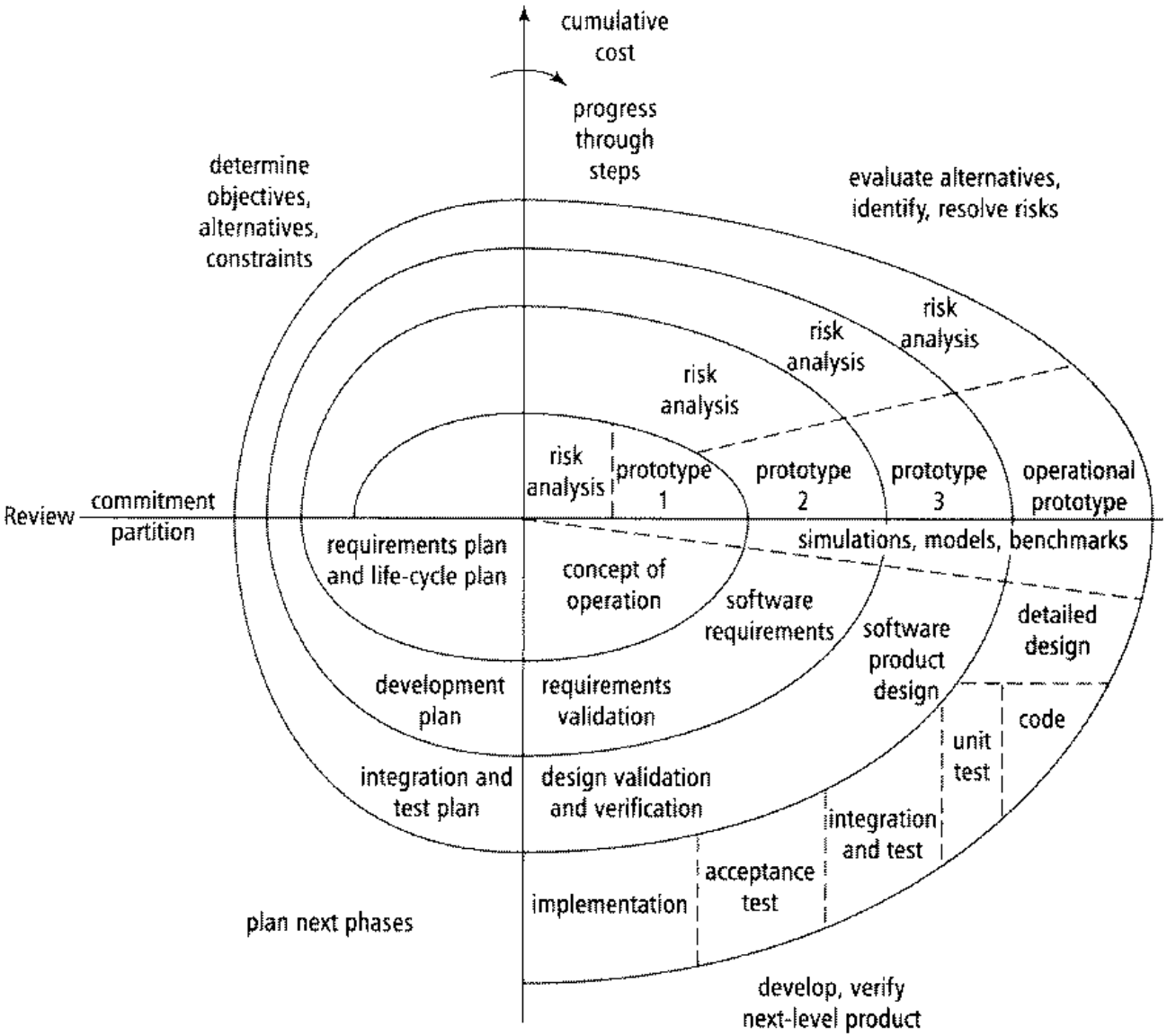
Uncertainty and Risk Removal

- One way to look at the software development process is to think of it as removing *uncertainty* and *risk*
- At the beginning of a project, there is high uncertainty about what the project is doing and how it is going to do it.
- As a project proceeds through requirements, specification, design and implementation, uncertainty is gradually removed as the solution becomes more and more specific.
- At the beginning of a project there is high risk that the project will not succeed in producing a software system to solve the given problem.
- As the project proceeds, risk is gradually removed as parts of the solution become evident and implemented.

The Spiral Model

- Combines classic life cycle and prototyping with risk analysis
- Develop system in an iterative fashion
  - identify sub-problem with highest risk
  - find a solution for that problem
- Each cycle removes some uncertainty about the final design of the system
- Each cycle brings the prototype closer to a production system.
- Risk analysis tries to identify problems and disasters early in the process

The Spiral Model



The Spiral Model

- Good model for real world software development.
  Cyclical approach is a good approximation to actual practice.
- Evolutionary approach makes sense given the inherent uncertainty in most software projects
- Forces frequent assessment of technical risks during development
- Needs care to guarantee convergence and cost containment
- Doesn’t guarantee that a major risk won’t go undiscovered

Project Management Cycle

- Plan the Project
  - What are the deliverables?
  - What tasks are needed to produce the deliverables?
  - What resources are needed and for how long?
  - What will the project cost?
- Schedule the Project
  - Who will work on it?
  - When are they available?
  - What other resources are needed?
  - How can the resources be acquired and used?
- Control the Project
  - What happened since the last time the project plan was updated?
  - What is the effect on the schedule?
  - What problems or opportunities need to be addressed?
  - Are we on time and within budget?

Project Plan & Project Control

- Project Plan includes overall control methods for
  - Project schedule
  - Project cost and budget
  - Resource management and team organization
  - Change and scope management
  - Status reporting methods
  - Project process model
- Project Plan includes steps in the delivery and support of the final product. Broken down into work packages (tasks) with specific deliverables.
  - Analysis and design
  - Construction
  - Data loading
  - Testing
  - Training
  - Delivery/installation
  - Maintenance

- Project Plan includes methods used to control the produce and process quality
  - Quality Assurance
  - Configuration Management
  - Security
  - Auditability
  - Risk Management
  - Documentation Management
  - Data Management
  - Proposed tools and techniques for any of the above

Purpose of a Project Plan

- Developing a formal project plan forces management to consider all of the activities and phases of a project.
- Project Plan is used for communication
  - Information about the scope and structure of a project
  - What must be done, how, why and when.
    Information for participants and stakeholders.
- The Project Plan is used
  - To schedule and organize the activities in a project.
  - To request appropriate time and budget.
  - To track project progress.

- The purpose of a project is to achieve clear goals that can be described by documented deliverables.
- Examples of intermediate deliverables
  - Design documents
  - Application source code
  - Approvals of intermediate work
  - Quality Assurance proofs
- Examples of final deliverables
  - Product and/or service
  - Customer/client satisfaction
  - Profit for the developers
  - i.e. the *realization of the project benefits*

Hierarchical Project Decomposition

- A project consists of hundreds or thousands of individual major *tasks*.
- Each task is broken down into *phases*.
- Each phase is broken down into *steps* (sub-phases)
- The steps are broken down into *activities* (also known as tasks) that describe the specific actions to be taken.
- For each activity the plan needs to specify the information in the table on the next slide.

why	purpose	text describing the importance of this task to the project.
what	deliverable	typically a small part of a larger deliverable
		e.g. source code for a single function or module.
who	personnel	type of person (e.g. tester, documentation writer)
	resources	later name actual individuals (e.g. Mary Smith)
		time required by each person to complete the activity
when	prerequisites	activity(ies) that must be completed before this
		activity can start and/or start dates.
		Start/End time may be calculated by project management software
how	material	Any hardware or software needed for this activity
	resources	Any items for which costs need to be tracked with this activity
where	location	Only if location management is necessary
		e.g. requires use of specific book-able facilities

Project Management Tools

- Most mangers use a project management tools such as MSProject, CA Super project, Timeline, Project Workbench, etc. to automate Project Plan creation and tracking.
- Input to a typical Project Planning Tool
  - Environment – work day calendar, typical hours per day
  - Staff resources – names, title, skills, availability, cost
  - Materials – name, description, price and payment schedule
  - Project – name, description, deliverables, budget (if not calculated), overhead costs, tool setup parameters (i.e. granularity of reports)
  - Phases – first level breakdown, includes same information as project
  - Task/Activity Descriptions

Task Activity Descriptions for Project Planning Tool

- Name and Task Identification number/code.
- Ownership (project, phase, step)
- Description (deliverables, outline of work)
- Bill-to information
- Task category (e.g. design, testing)
- Original time estimate
- Original budget (unless done at a higher level)
- Skill sets required (e.g. designer, tester, Windows NT Admin, etc.
- Earliest start date, fixed or flexible
- Latest end date, fixed or flexible.

- Any related tasks (name or identification number/code)

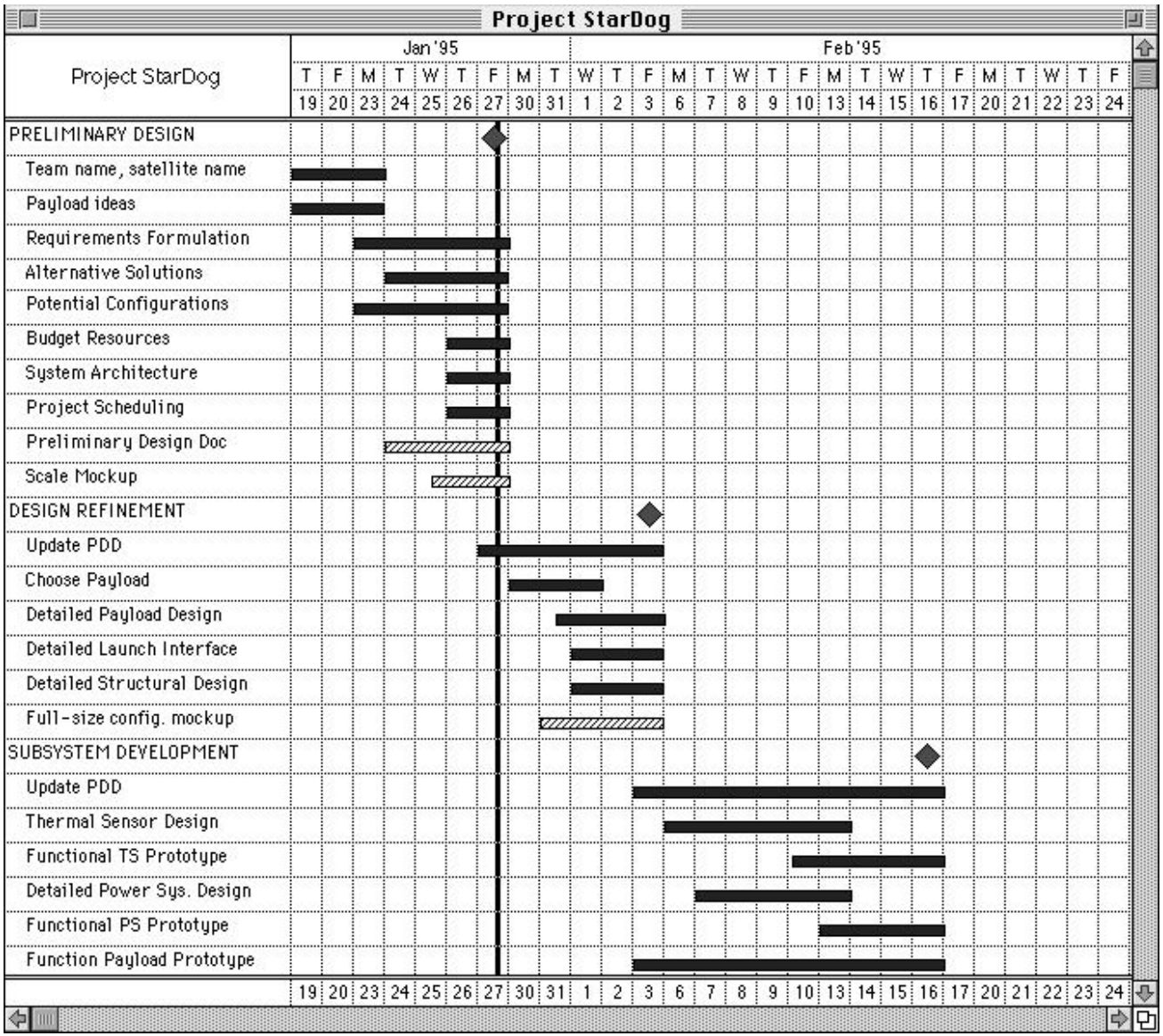
  Identify prerequisites and successors.

- Material resources used (identification, quantity, cost)

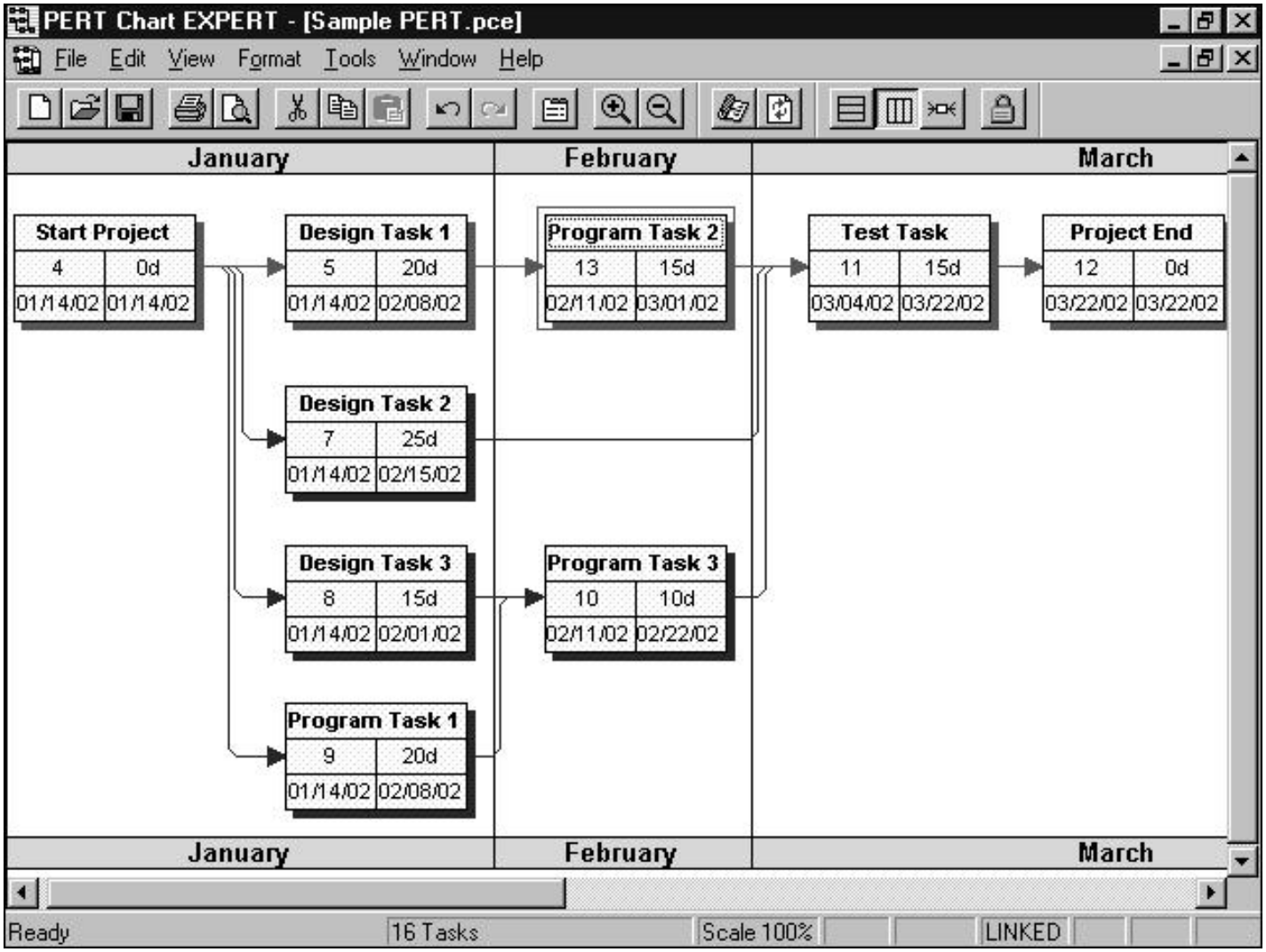
- Staff resources used (time allocated, availability)

- Additional costs (other than resource costs)

Gannt Chart Project Planning Tool



Pert Chart Project Planning Tool



Project Plan Update During Project

- Revised start dates
- Actual end dates
- Actual cost
- For each staff member
  - Time spent on the task/activity.
  - Estimated time to completion
- Material resources used.
- Changes in Phases, Tasks, Activities

  Add/delete tasks, reassign resources.

Project Planning Tool Output

Comparisons	Between original, revised and actual, schedule and budgets
Critical Path	Sequence of tasks that define the earliest completion date i.e. tasks with no float.
Earliest Completion	Total time for tasks on the critical path.
Float/Slack	Latest start date – earliest start date for each task Latest end date – earliest end date for each task Tasks that could be started earlier or end later without affecting project completion date.
Staff load	Staff who have too much or too little work assigned.

Phase Costs	Total of task costs + phase overhead costs
Project costs	Total of phase costs + project overhead costs
Resource costs	For staff – time * rate For equipment – usage * unit cost
Task costs	Cost of assigned resources + task overhead costs
Task overruns	Days or dollars that task exceeded schedule or budget.
Task start end dates	If not provided, then calculated by the tool
Task status	Outstanding (not started), in progress (expected end), completed.
Totals by category	Summaries of all of the above.
Personel reports	Assigned tasks, status, dates, etc.

Pert Chart with Critial Path