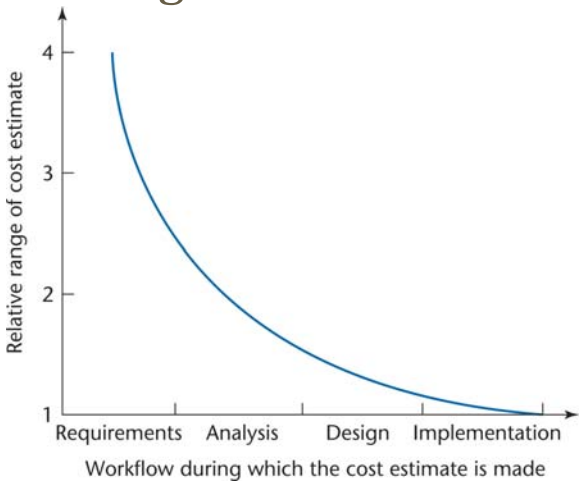
Software Engineering

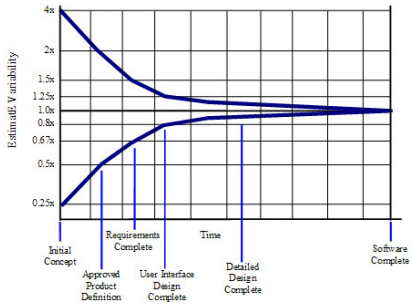
CSC440/640
Prof. Schweitzer
Week 5

Planning and the Software Process



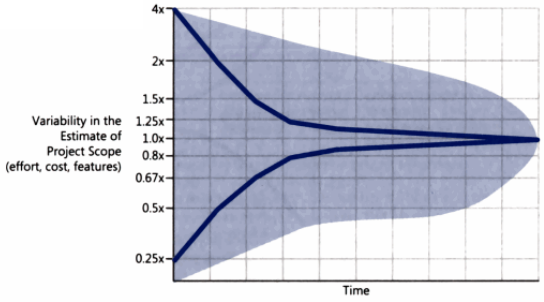
- The accuracy of estimation increases as the process proceeds

Cone of Uncertainty



Your estimate cannot have more accuracy than is possible at your project's current position in the cone.

If your project does not reduce variability over time, then you have a "cloud of uncertainty" instead



Are You Good at Estimating?

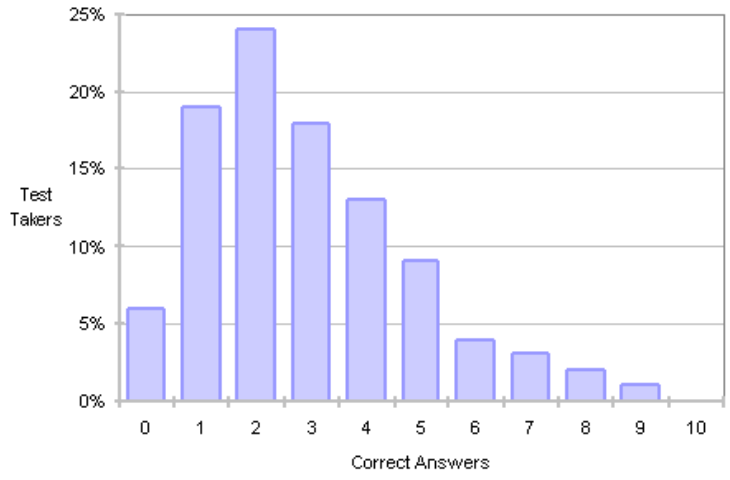
- Estimate a Minimum and Maximum for Each of These:
  - Surface temperature of the sun
  - Latitude of Shanghai
  - Area of the Asian continent
  - The year of Alexander the Great's birth
  - Total value of U.S. currency in circulation in 2004
  - Worldwide box office receipts for the movie Titanic
  - Total length of the coastline of the Pacific Ocean
  - Number of book titles published in the U.S. since 1776
  - Heaviest blue whale ever recorded
  - Area of the State of Wisconsin
- **Estimate at the 90 percent confidence level**
  - 90% sure the correct answer is in your range
- No Internet Research (Phone, Laptop, etc.)
- Take 10 Minutes. You won't be graded on this.

Are You Good at Estimating? (2)

Question	Answer
Surface temperature of the sun	10,000F / 6,000C
Latitude of Shanghai	31° North
Area of the Asian continent	17,139,000 mi^2 44,390,000 km^2
The year of Alexander the Great's birth	356 BC
Total value of U.S. currency in circulation in 2004	$719.9 billion
Worldwide box office receipts for the movie Titanic	$1.835 billion
Total length of the coastline of the Pacific Ocean	84,300 mi / 135,663 km
Number of book titles published in the U.S. since 1776	22 million
Heaviest blue whale ever recorded	380,000 lbs / 170,000 kg
Area of the State of Wisconsin	65,498 mi^2 169,639 km^2

How Good Are Most People at Estimating?

- Steve McConnell's Results



What Affects Your Estimates?

- Number of Requirements
- Inappropriate Precision
- Omitted Activities
- Complexity of Individual Requirements
- Team Skill
- Confirmation Bias
- Number of Members on Your Team
- Technology
  - Bleeding Edge
  - Does the Technology Match the Problem Domain
- Integration Points (Number and Complexity)
- Non-Functional Requirements
  - Performance
  - Uptime

Accuracy vs. Precision

- *Accuracy* refers to how close to the real value a number is.
- *Precision* refers to how exact a number is.
- What is the value for Pi (π)?
  - Accurate: 3, 3.14, 3.2
  - Precise: 3.37882
  - Accurate and Precise: 3.14159265359
- Airline Schedules are Precise to the *Minute* but aren't very accurate

Metrics for the Size of a Product

- Lines of code (LOC, KDSI, KLOC)
- FFP
- Function Points
- COCOMO

Lines of Code (LOC)

- Alternate metric
  - Thousand delivered source instructions (KDSI)
- Source code is only a small part of the total software effort
- Different languages lead to different lengths of code
- LOC is not defined for nonprocedural languages (like LISP)
- It is not clear how to count lines of code
  - Executable lines of code?
  - Data definitions?
  - Comments?
  - JCL statements?
  - Changed/deleted lines?
- Not everything written is delivered to the client
- A report, screen, or GUI generator can generate thousands of lines of code in minutes

Lines of Code

- LOC is accurately known only when the product finished
- Estimation based on LOC is therefore doubly dangerous
  - To start the estimation process, LOC in the finished product must be estimated
  - The LOC estimate is then used to estimate the cost of the product — an uncertain input to an uncertain cost estimator
  - Lines of Code can change based on the language, and even the *version* of a language being used.

FFP Metric

- For cost estimation of medium-scale data processing products
- The three basic structural elements of data processing products
  - Files
  - Flows
  - Processes
- Given the number of files (*Fi*), flows (*Fl*), and processes (*Pr*)
  - The size (*S*), cost (*C*) are given by

    $S = Fi + Fl + Pr$

    $C = b \times S$
- The constant *b* (efficiency or productivity) varies from organization to organization

Function Points

- Based on the number of inputs (Inp), outputs (Out), inquiries (Inq), master files (Maf), interfaces (Inf)
- For any product, the size in "function points" is given by
  - $FP = 4 \times Inp + 5 \times Out + 4 \times Inq + 10 \times Maf + 7 \times Inf$
- This is an oversimplification of a 3-step process

Function Points

- Step 1
  - Classify each component of the product (Inp, Out, Inq, Maf, Inf) as simple, average, or complex
  - Assign the appropriate number of function points
  - The sum gives UFP (unadjusted function points)

	Level of Complexity		
Component	**Simple**	**Average**	**Complex**
Input item	3	4	6
Output item	4	5	7
Inquiry	3	4	6
Master file	7	10	15
Interface	5	7	10

Function Point

- Step 2
  - Compute the technical complexity factor (TCF)
  - Assign a value from 0 ("not present") to 5 ("strong influence throughout") to each of 14 factors such as transaction rates, portability

1. Data communication
2. Distributed data processing
3. Performance criteria
4. Heavily utilized hardware
5. High transaction rates
6. Online data entry
7. End-user efficiency
8. Online updating
9. Complex computations
10. Reusability
11. Ease of installation
12. Ease of operation
13. Portability
14. Maintainability

Function Points

- Add the 14 numbers
  - This gives the total degree of influence (DI)
    - TCF = 0.65 + 0.01 × DI
- The technical complexity factor (TCF) lies between 0.65 and 1.35
- Step 3
  - The number of function points (*FP*) is then given by

    $FP = UFP \times TCF$

Analysis of Function Points

- Function points are usually better than KDSI — but there are some problems
- "Errors in excess of 800% counting KDSI, but only 200% in counting function points" [Jones, 1987]

Expert Judgment by Analogy

- Experts compare the target product to completed products
  - Guesses can lead to hopelessly incorrect cost estimates
  - Experts may recollect completed products inaccurately
  - Human experts have biases
  - However, the results of estimation by a broad group of experts may be accurate
- The Delphi technique is sometimes needed to achieve consensus

Bottom-up Approach

- Break the product into smaller components
  - The smaller components may be no easier to estimate
  - However, there are process-level costs
- When using the object-oriented paradigm
  - The independence of the classes assists here
  - However, the interactions among the classes complicate the estimation process

Granular Estimates

- Subdivide large tasks into small tasks
- If you don't understand a task, don't deemphasize it
  - i.e. – One line item for "Data Conversion" estimated for two weeks
  - Sign that a task is not well understood, and estimate likely to be very inaccurate
- Tasks should require no more than 2 days effort
- Estimate a Best Case and Worst Case
  - Thinking about the worst case often makes the developer think about tasks they might have otherwise forgotten
- Remember to Compare Your Estimates with Actuals
  - Learn to Become a Better Estimator!

Count, Compute, Judge

- Don't *Estimate* When You Can *Count*
- Find something to count that's highly correlated to the size of the software you're estimating
  - Feature Points, Requirements, Screens, Use Cases, Reports, Classes, etc.
- Use Historical Data to Compute Time Based on Counts
  - Average Effort Hours per Requirement, Report, Screen, etc.
- Use Judgment Only as Last Resort
  - Expert judgment least accurate method of estimation
  - Avoid using expert judgment to "tweak" an estimate

Algorithmic Cost Estimation Models

- A metric is used as an input to a model to compute cost and duration
  - An algorithmic model is unbiased, and therefore superior to expert opinion
  - However, estimates are only as good as the underlying assumptions
- Examples
  - SLIM Model
  - Price S Model
  - COnstructive COst MOdel (COCOMO)

Calibration and Historical Data

- Three Types of Data
  - Industry – Data from other organizations that do similar work
  - Historical – Data from *your* organization
  - Project – Data generated from previous iterations of this project
- Accounts for Organizational Influences
  - Can you create stable requirements?
  - Is the team able to concentrate solely on the project?
  - Does the organization use good process?
- Avoids Subjectivity and Unfounded Optimism
- Reduces Estimation Politics

T-Shirt Sizing

Feature Size	Average LOC Per Feature	Number of Features	Estimated LOC
Extra Small	127	22	2,794
Small	253	15	3,795
Medium	500	10	5,000
Large	1,014	30	30,420
Extra Large	1,998	27	53,946
Total	**-**	**104**	**95,955**

- Average LOC from Historical Data
- Law of Large Numbers Applies Here
  - Whole Estimate More Accurate than Individual Pieces

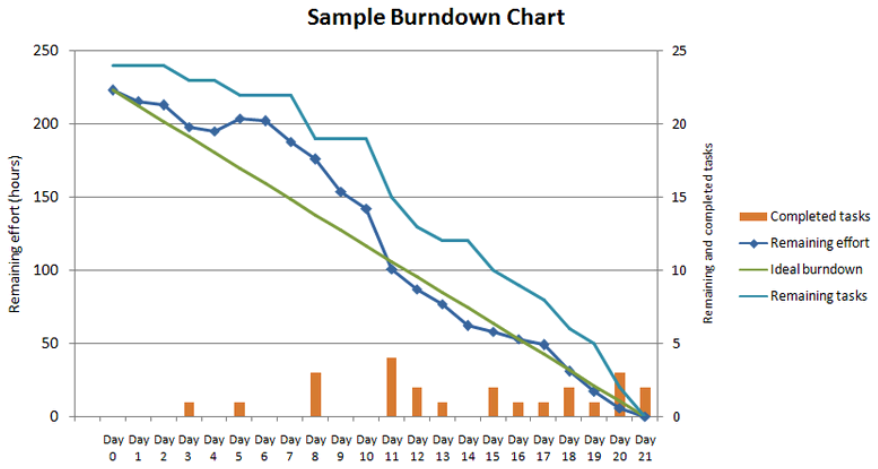
Other Types of Proxies

- Standard Components
  - Dynamic Web Pages
  - Static Web Pages
  - Database Tables
  - Reports
  - Business Rules
- Story Points
  - Unit-less Measure Approximating Size of a Feature
  - Must be Calibrated to Historical Data
  - Can Be Used in Conjunction with Velocity to Show Progress

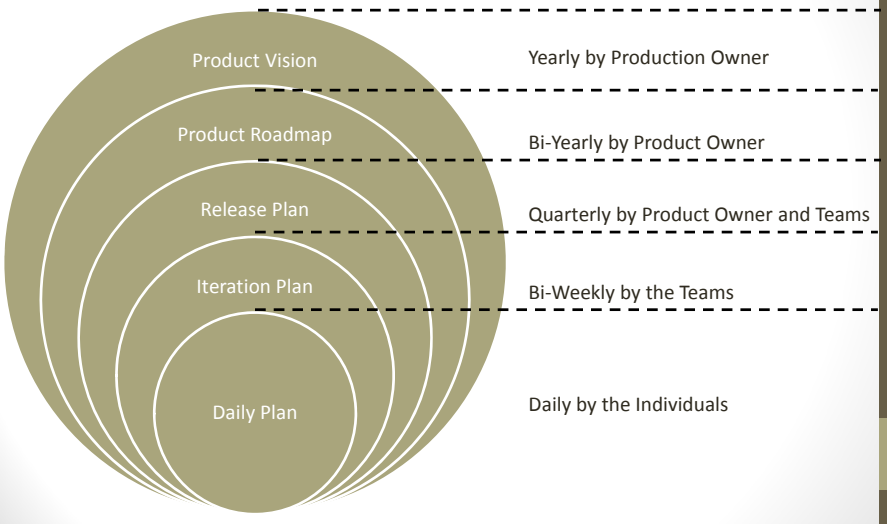
Software Project Management Plan Framework

- There are many ways to construct an SPMP
- One of the best is IEEE Standard 1058.1
  - The standard is widely accepted
  - It is designed for use with all types of software products
  - It supports process improvement
    - Many sections reflect CMM key process areas
  - It is ideal for the Unified Process
    - There are sections for requirements control and risk management
- Agile Project Management
  - Depends on Measuring *Velocity*
    - Number of Story Points Per Iteration a Team Can Complete

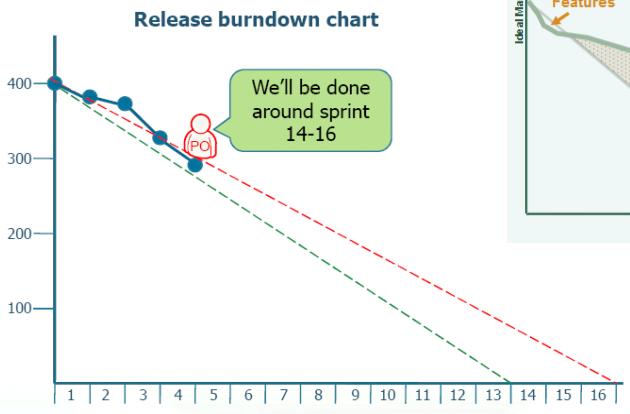
Burn Down Chart

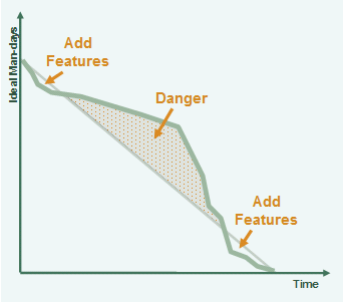


Agile Planning



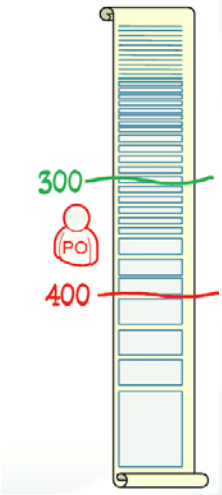
Scope Driven Forecasting





Time Driven Forecasting

- Velocity: 30 (Worst Case) – 40 (Best Case)
- Remaining Sprints: 10



The Specification Document Must Be…

- Informal enough for the client
  - The client is generally not a computer specialist
- Formal enough for the developers
  - It is the sole source of information for drawing up the design
- These two requirements are mutually contradictory

The Specification Document

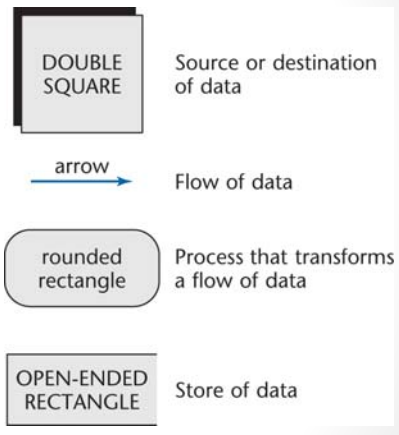
- The specification document is a contract between the client and the developers
- Typical constraints
  - Deadline
  - Parallel running
  - Portability
  - Reliability
  - Rapid response time
- For real-time software
  - Hard real-time constraints must be satisfied
- Acceptance criteria
  - It is vital to spell out a series of tests
- If the product passes the tests, it is deemed have satisfied its specifications
- Some acceptance criteria are restatements of constraints

Structured Systems Analysis

- Three popular graphical specification methods of 1970s
  - DeMarco
  - Gane and Sarsen
  - Yourdon
- All are equivalent
- All are equally good
- Many U.S. corporations use them for commercial products
- Gane and Sarsen's method is taught here because it is so widely used

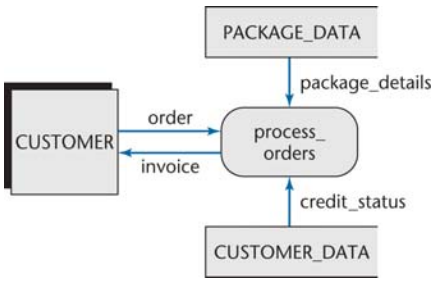
Sally's Software Shop Mini Case Study

- The data flow diagram (DFD) shows the logical data flow
  - "What happens, not how it happens"
- Symbols:



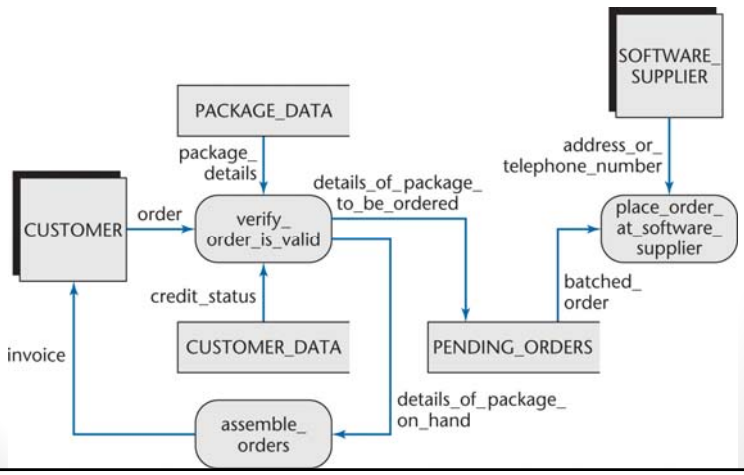
Step 1 - Draw the DFD

- First refinement
  - Infinite number of possible interpretations



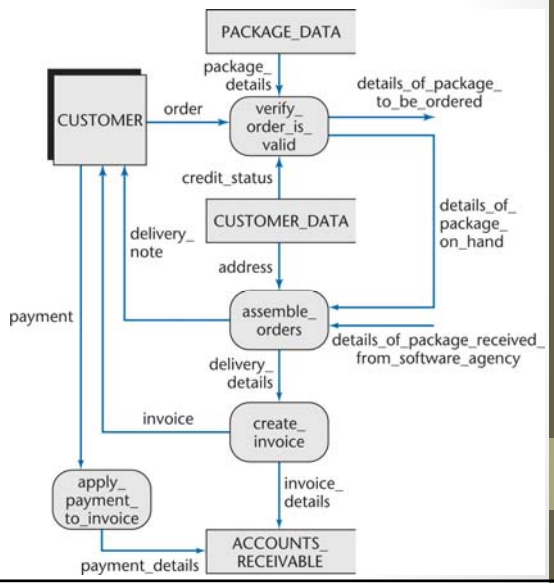
Step 1 – Second Refinement

- PENDING ORDERS is scanned daily



Step 1 – Third Refinement

- Portion of third refinement
- The final DFD is larger
  - But it is easily understood by the client
- When dealing with larger DFDs
  - Set up a hierarchy of DFDs
  - A box becomes a DFD at a lower level



Step 2 - Decide What Parts to Computerize and How

- It depends on how much client is prepared to spend
- Large volumes, tight controls
  - Batch
- Small volumes, in-house microcomputer
  - Online
- Cost/benefit analysis

Step 3 - Determine the Details of the Data Flows

- Determine the data items for each data flow
- Refine each flow stepwise
  - Example;

```
order:
        order_identification
        customer_details
        package_details
```

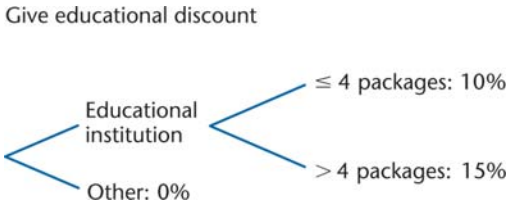
- We need a data dictionary for larger products

Sample Data Dictionary Entries

Name of Data Element	Description	Narrative
order	Record comprising fields order_identification customer_details customer_name customer_address . . . package_details package_name package_price . . .	The fields contain all details of an order
order_identification	12-digit integer	Unique number generated by procedure generate_order_number. The first 10 digits contain the order number itself, the last 2 digits are check digits.
verify_order_is_valid	Procedure: Input parameter: order Output parameter: number_of_errors	This procedure takes order as input and checks the validity of every field; for each error found, an appropriate message is displayed on the screen (the total number of errors found is returned in parameter number_of_errors).

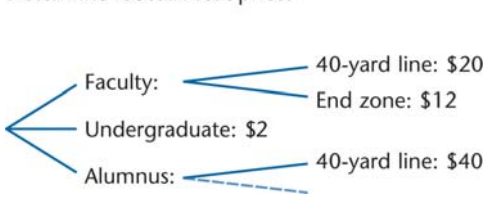
Step 4 - Define the Logic of the Processes

- We have process `give educational discount`
  - Sally must explain the discount she gives to educational institutions
    - 10% on up to 4 packages
    - 15% on 5 or more
- Translate this into a decision tree



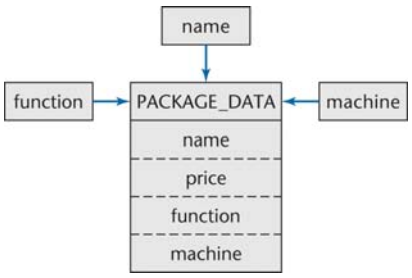
Step 4 - Define the Logic of the Processes

- The advantage of a decision tree
  - Missing items are quickly apparent



Step 5 - Define the Data Stores

- Define the exact contents and representation (format)
  - COBOL: specify to `pic` level
  - Ada: specify `digits or delta`
- Specify where immediate access is required
  - Data immediate-access diagram (DIAD)



Step 6 - Define the Physical Resources

- For each file, specify
  - File name
  - Organization (sequential, indexed, etc.)
  - Storage medium
  - Blocking factor
  - Records (to field level)
  - Table information, if a DBMS is to be used

Step 7 - Determine Input/Output Specifications

- Specify
  - Input forms
  - Input screens
  - Printed output

Step 8 - Determine the Sizing

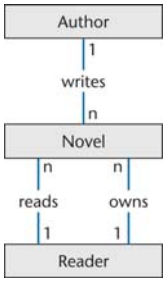
- Obtain the numerical data needed in Step 9 to determine the hardware requirements
  - Volume of input (daily or hourly)
  - Size, frequency, deadline of each printed report
  - Size, number of records passing between CPU and mass storage
  - Size of each file

Step 9 - Determine the Hardware Requirements

- Mass storage requirements
- Mass storage for back-up
- Input needs
- Output devices
- Is the existing hardware adequate?
  - If not, recommend whether to buy or lease additional hardware

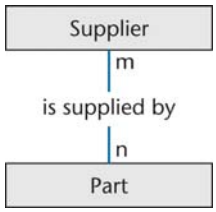
Entity-Relationship Modeling

- Semi-formal technique
  - Widely used for specifying databases
  - Example:



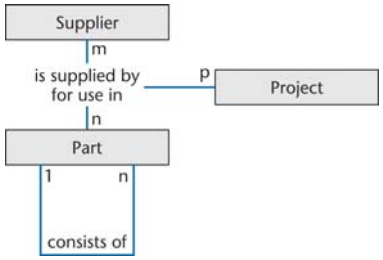
Entity-Relationship Diagrams

- Many-to-many relationship



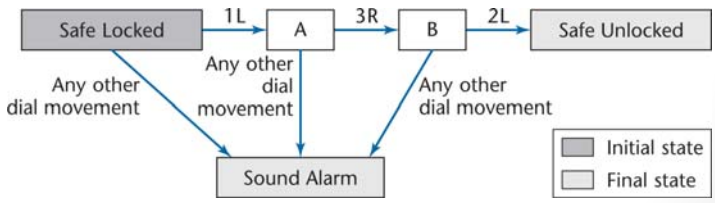
Entity-Relationship Diagrams

- More complex entity-relationship diagrams



Finite State Machines

- Case study
  - A safe has a combination lock that can be in one of three positions, labeled 1, 2, and 3. The dial can be turned left or right (L or R). Thus there are six possible dial movements, namely 1L, 1R, 2L, 2R, 3L, and 3R. The combination to the safe is 1L, 3R, 2L; any other dial movement will cause the alarm to go off



Finite State Machines

- The set of states J is {Safe Locked, A, B, Safe Unlocked, Sound Alarm}
- The set of inputs K is {1L, 1R, 2L, 2R, 3L, 3R}
- The transition function T is below
- The initial state J is Safe Locked
- The set of final states J is {Safe Unlocked, Sound Alarm}

Dial Movement \ Current State	Table of Next States Safe Locked	A	B
1L	A	Sound alarm	Sound alarm
1R	Sound alarm	Sound alarm	Sound alarm
2L	Sound alarm	Sound alarm	Safe unlocked
2R	Sound alarm	Sound alarm	Sound alarm
3L	Sound alarm	Sound alarm	Sound alarm
3R	Sound alarm	B	Sound alarm

Finite State Machines

- The power of an FSM to specify complex systems
- There is no need for complex preconditions and postconditions
- Specifications take the simple form
  - current state and event and predicate → next state

Finite State Machines

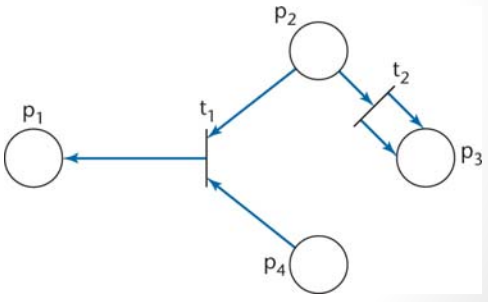
- Using an FSM, a specification is
  - Easy to write down
  - Easy to validate
  - Easy to convert into a design
  - Easy to convert into code automatically
  - More precise than graphical methods
  - Almost as easy to understand
  - Easy to maintain
- However
  - Timing considerations are not handled

Petri Nets

- A major difficulty with specifying real-time systems is timing
  - Synchronization problems
  - Race conditions
  - Deadlock
- Often a consequence of poor specifications
- Petri Nets
  - A powerful technique for specifying systems that have potential problems with interrelations
- A Petri net consists of four parts:
  - A set of places P
  - A set of transitions T
  - An input function I
  - An output function O

Petri Nets

- Set of places P is {p1, p2, p3, p4}
- Set of transitions T is {t1, t2}
- Input functions:
  - I(t1) = {p2, p4}
  - I(t2) = {p2}
- Output functions:
  - O(t1) = {p1}
  - O(t2) = {p3, p3}



Petri Nets

- More formally, a Petri net is a 4-tuple C = (P, T, I, O)
  - P = {p1, p2, …, pn} is a finite set of places, n ≥ 0
  - T = {t1, t2, …, tm} is a finite set of transitions, m ≥ 0, with P and T disjoint
  - I : T → P^{∞} is the *input* function, a mapping from transitions to bags of places
  - O : T → P^{∞} is the *output* function, a mapping from transitions to bags of places
  - (A bag is a generalization of a set that allows for multiple instances of elements, as in the example on the previous slide)
  - A marking of a Petri net is an assignment of tokens to that Petri net

Z - Notation

- Z (pronounced “zed”) is a formal specification language
- Z is the most widely used formal specification language
- It has been used to specify
  - CICS (part)
  - An oscilloscope
  - A CASE tool
  - Many large-scale projects (especially in Europe)
- Difficulties in using Z
  - The large and complex set of symbols
  - Training in mathematics is needed

Z - Notation

- Reasons for the great success of Z
  - It is easy to find faults in Z specifications
  - The specifier must be extremely precise
  - We can prove correctness (we do not have to)
  - Only high-school math needed to read Z
  - Z decreases development time
  - A “translation” of a Z specification into English (or another natural language) is clearer than an informal specification

Behavior Driven Development

- Combines Domain Driven Design and Test Driven Development
- Makes heavy use of different tools to allow developers and business analysts to collaborate
- User Stories have a formal language
  - Tooling being developed to make this into a DSL for Testing or Even Development
  - JBehave/NBehave
  - Fitness
  - SpecFlow

Example BDD Story

Scenario: A trader is alerted of status

Given a stock and a threshold of 15.0
When stock is traded at 5.0
Then the alert status should be OFF
When stock is traded at 16.0
Then the alert status should be ON

Comparison of Classical Analysis Techniques

- Formal methods are
  - Powerful, but
  - Difficult to learn and use
- Informal methods have
  - Little power, but are
  - Easy to learn and use
- There is therefore a trade-off
  - Ease of use versus power

Next Week

- Reading – Chapters 13 & 14
- Feedback Will Be Provided on Use Cases