WBS & Requirements

CSCI 5040: Professional Master’s Project (1 of 2)

Lecture 7

Thanks to Alan Paradise

- I should mention…
- When I learned I was setting up this capstone, I worked with Alan Paradise over the summer
- Alan kindly gave me access to his Moodle for his UG capstone
- Much of the flow of the class and some of the materials I share here come from Alan's class materials and are used with his permission
- Thanks, Alan!

Learning Objectives

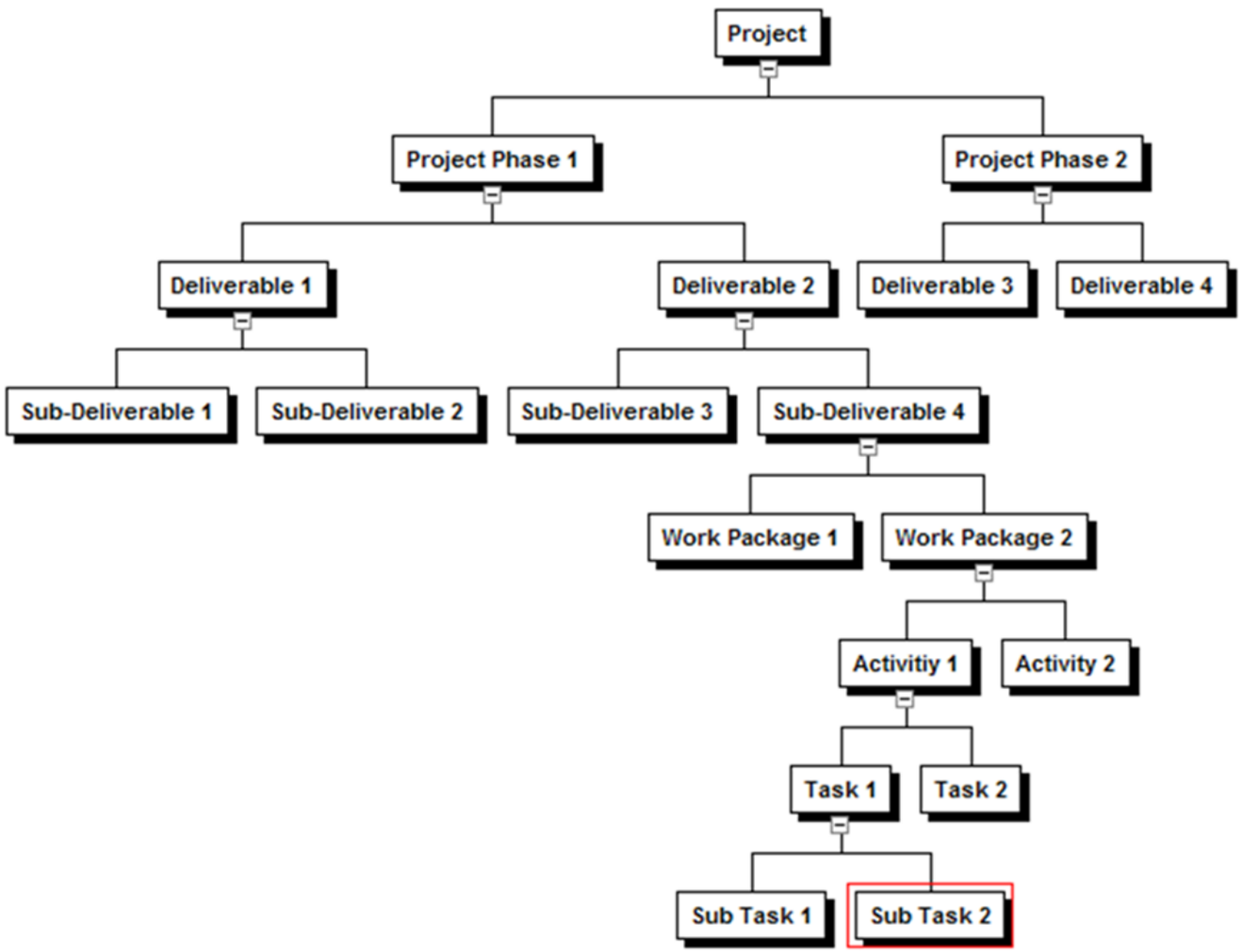
- Review Work Breakdown Structure best practices
- Review requirements best practices
- Using UML in requirements
- Task tracking/project rhythms
- Review design team activities/class schedule & deliverables

Work Breakdown Structures

- A WBS (Work Breakdown Structure) is
  - A deliverable-oriented hierarchical decomposition of work to accomplish objectives or create deliverables (internal or external)
  - It defines the total scope of a project or project phase
  - Each level deeper in a WBS represents increasing detailed definition of project work
- The WBS is decomposed into work packages
- Process originally comes from the DOD & NASA in the 1960s
- Can be done as a graphical diagram or as an outlined text document

- From: Practice Standard for Work Breakdown Structures (2nd Ed.), 2006, Project Management Institute

WBS - Diagram



- A complex graphical WBS example, including multiple project levels:
  - Projects
  - Phases
  - Deliverables
  - Sub-deliverables
  - Work packages
  - Activities
  - Tasks
  - Sub-tasks
- You can decide what levels are appropriate for your project

WBS – Outline/Text

- 1 Project
- 1.1 Phase 1
- 1.1.1 Deliverable 1
- 1.1.1.1 Sub Deliverable 1
- 1.1.1.2 Sub Deliverable 2
- 1.1.2 Deliverable 2
- 1.1.2.1 Sub Deliverable 3
- 1.1.2.2 Sub Deliverable 4
- 1.1.2.2.1 Work Package 1
- 1.1.2.2.2 Work Package 2
- 1.1.2.2.2.1 Activity 1
- 1.1.2.2.2.1.1 Task 1
- 1.1.2.2.2.1.1.1 Sub Task 1
- 1.1.2.2.2.1.1.2 Sub Task 2
- 1.1.2.2.2.1.2 Task 2
- 1.1.2.2.2.2 Activity 2
- 1.2 Phase 2
- 1.2.1 Deliverable 3
- 1.2.2 Deliverable 4

The same WBS example as previously shown, this time presented as a textual outline view.

The outline numbering represents that elements position in the project structure.

Why a WBS Matters

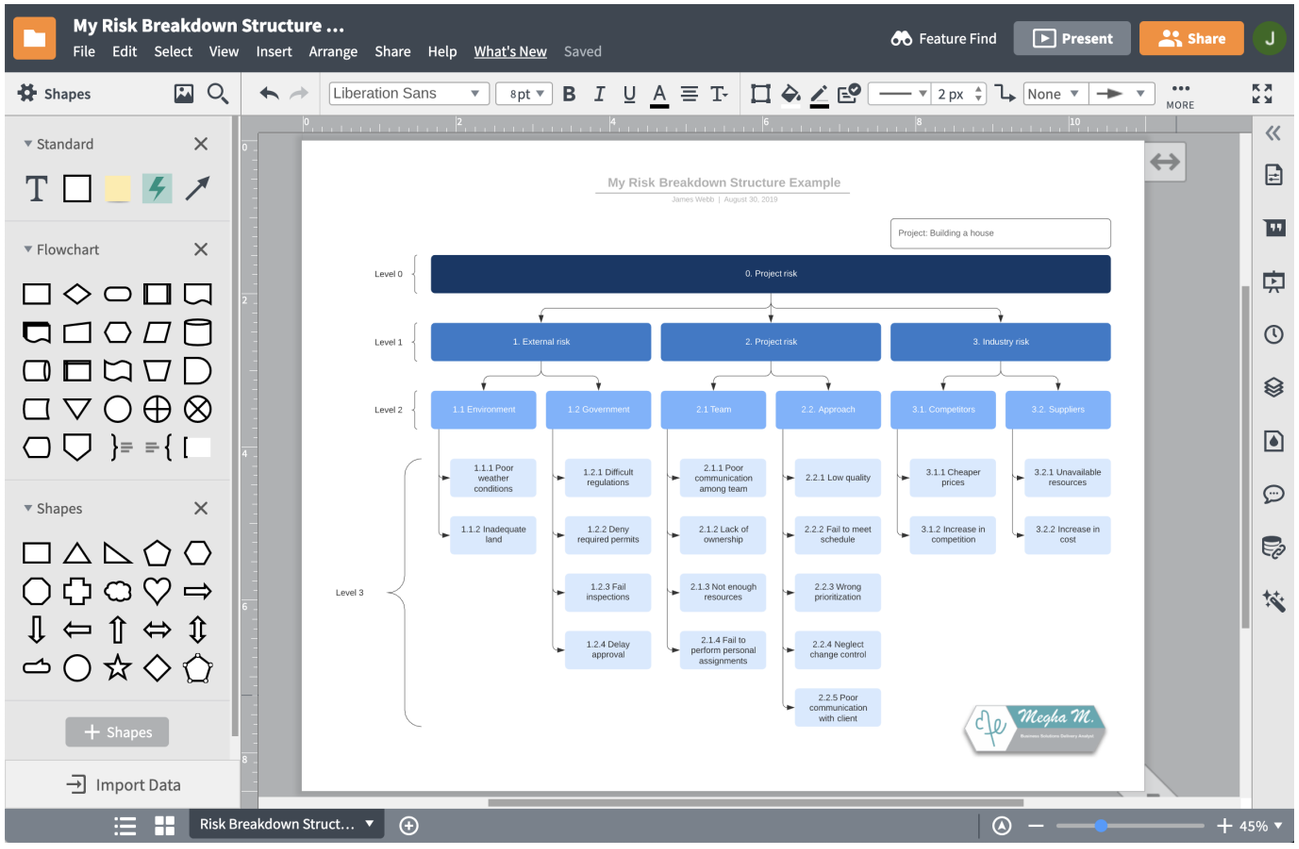
- A successful project must focus on deliverables
  - what are all of the project deliverables – full scope – in and out of project
  - what has to be done or produced to create them
  - who is responsible
  - when is it due
  - what does it cost
  - what is the acceptance criteria
- Thorough identification of project scope and deliverables provides for a better chance of a successful delivery
- The WBS process also provides for team building and buy-in through confirming common understanding of project scope and deliverables
- A WBS reduces scope creep by gathering and reviewing all requirements from all stakeholders, and provides a baseline for future change control
- The WBS supports communication, estimation, confidence, and control of project deliverables

WBS Best Practices

- The focus of creating a project WBS should be on scope and deliverables, not on time, resources, or other concerns
  - Focus on the what initially, not who or when
- The 100% rule: a project WBS should represent 100% of the work and deliverables within scope of the project
- Anything not in the WBS is considered out of scope
- A project WBS should include all cross-functional areas that impact the deliverables – including project management tasks – that occur in the project (presentations, reporting, etc.)
- Ideally, the lowest level of an WBS should consist of elements that can be managed, estimated, and measured
- Clearly identify deliverables (nouns not verbs)
- Each element represents a single deliverable (internal and external)
- Items that are considered high cost or high risk should be further decomposed
- The 80-hour rule: each low-level task should be 8-80 hours in duration
- Where possible, limit each new level to 5-7 elements (for readability) – this may require addition of summary tasks or other breakdowns
- Like all project documents, the WBS should be version controlled and reviewed regularly by stakeholders
- Create the WBS jointly with the project team – for ownership and team building
- The WBS should be reviewed before a schedule is created

WBS Tools

- Whiteboard & camera – easy, free!
- Specialized App
  - WBS Schedule Pro (trial/$)
- Web tools
  - Lucidchart (free/limited)
  - Diagrams.net (free/open source)
  - Plan Hammer (trial/$)
- Word/Excel
  - Word Outline View (part of Word)
  - My Word Templates
- Mind-mapping tools
  - Mindmeister
  - XMind



WBS to Stories and Estimates

- It's easy to take the sub-tasks in a WBS (the bottom layer of work packages) and create stories or deliverables for an agile Scrum or Kanban tracking tool
- Tasks can be "t-shirt sized" for initial estimates of project resource needs [5]
  - T-shirt sizing: set effort durations to Small/Medium/Large/XL type categories – allows a roll up for resource and timing prior to detailed planning
- Tasks can easily be assigned to resources on the project
- Or missing resources can be identified
- A WBS will be more accurate if you drive to tasks that are small and owned by one person
- It's a very strong tool for estimating – because of the bottom up approach
- If you've used T-Shirt sizes for tasks, rolling up an estimate is simple
- You could also do a similar task cost roll-up for a preliminary budget
- It's a great tool for alignment on the actual work that is being asking for compared to what you're planning to do

WBS In Use

- You could use a WBS to track progress without using other tools
- But usually you'll go to stories or cards to track in a Scrum or Kanban
- Or move the tasks into something like a Microsoft Project Gantt chart or similar project scheduling tool
- For more information, see the PMI's WBS practice standard (on O'Reilly site)
- The CDC web site also provides a
  - WBS use guideline
    - http://www2.cdc.gov/cdcup/library/practices_guides/CDC_UP_WBS_Practices_Guide.pdf
  - Project management templates, examples, and checklists
    - http://www2.cdc.gov/cdcup/library/matrix/default.htm

Requirements Engineering

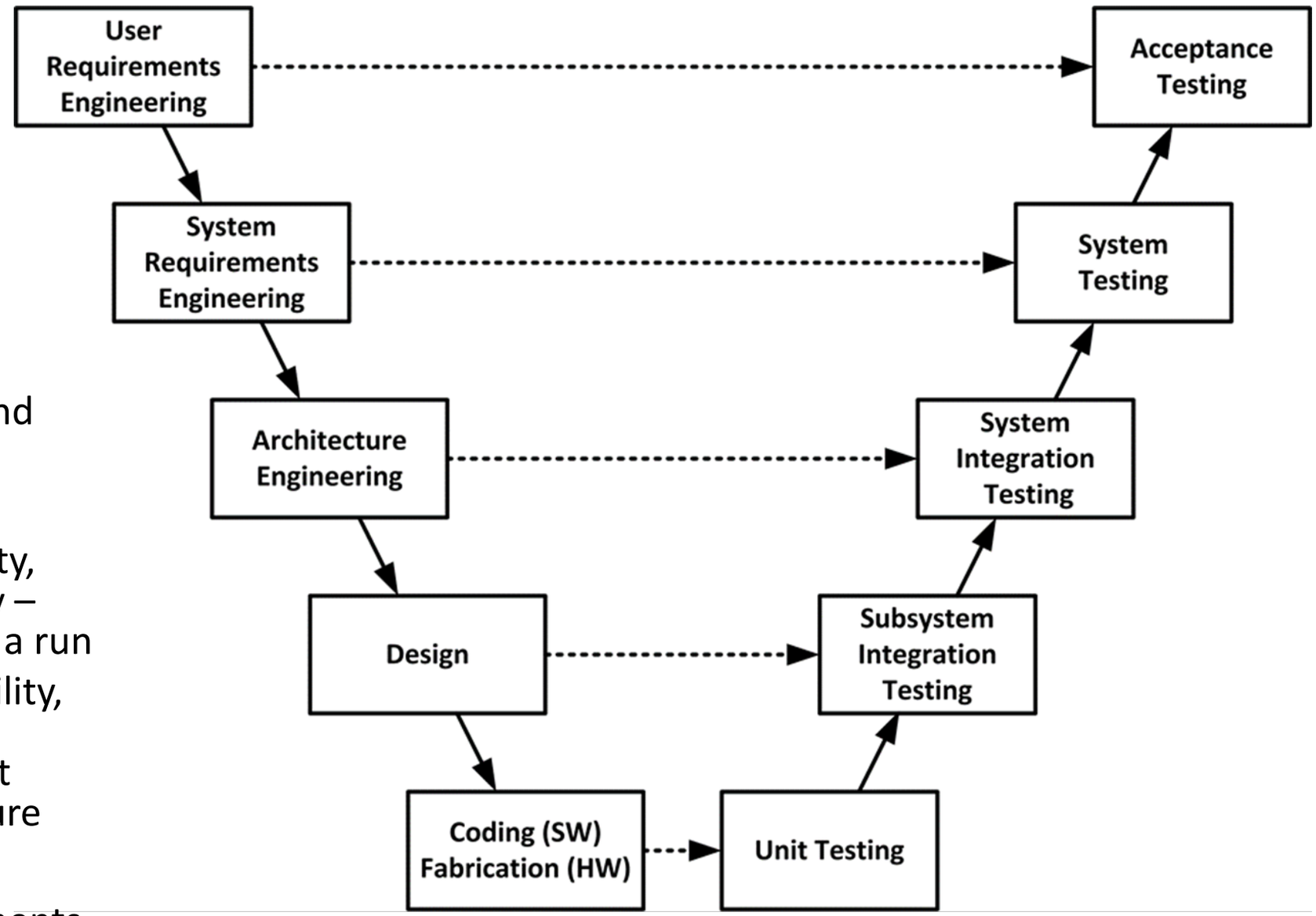
- Why Requirements?
  - User wants, needs, feasibility, specification, validation, management
  - Agreeing to a solution approach
- Typical stages
  - Inception – basic problem understanding, identification of stakeholders, the charter
  - Elicitation – asking for details of the problem
    - Limits/issues: scope, understanding, volatility
  - Elaboration – expand/refine information from earlier stages
  - Negotiation – dealing with conflicting requirements or other mismatches
  - Specification – capture the requirements for project use
    - Text, Graphical/UML, Use Cases, Prototypes, User Stories, any combination
  - Validation – Look for any inconsistencies, missing information, or errors and make corrections

Requirements Elicitation

- Elicitation Workshop
  - Collaboration/Brainstorming with a Facilitator
  - Capture – whiteboard, post-its, shared Google Doc
  - Later – affinity grouping/what goes with what, share the results
- Stay at a high (or selected) level
  - Use a parking lot to capture issues/details to come back to
- Sparking discussion
  - What's needed?
  - Who are the customers/users?
  - What is the technical environment?
  - Typical usage scenarios?
  - Interfaces or connections?
  - Other limits or performance needs?
- Other gathering methods: User Interviews, Focus Groups, Observation, Questionnaires, Examine Current Software/Documentation/Interfaces

Requirement Level and Types

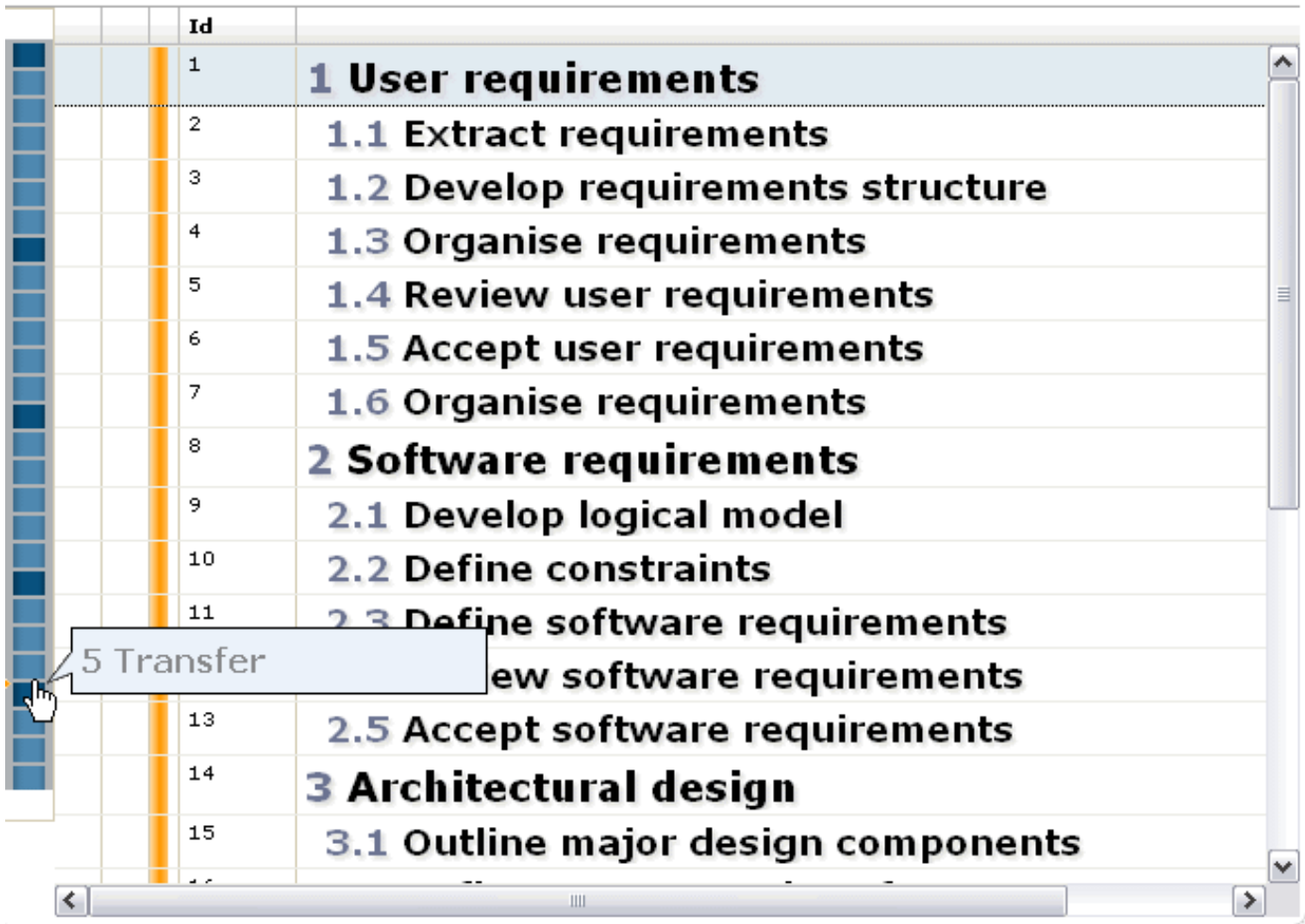
- Levels
  - The “V”…
- Types
  - Functional
    - Clear steps to implementation and assessment
  - Non-functional
    - Execution – security, reliability, usability – observable during a run
    - Evolution – testability, maintainability, scalability – look at static code/structure
  - Constraints
    - Specific measurements that impact the design



https://insights.sei.cmu.edu/sei_blog/2013/11/using-v-models-for-testing.html

Requirements Specification Formats

- Text
  - Specifications Documents
  - Requirements Tracking Systems (e.g. DOORS →)
- Graphical/UML
- Use Cases
- Prototypes
- User Stories
- Any combination of these



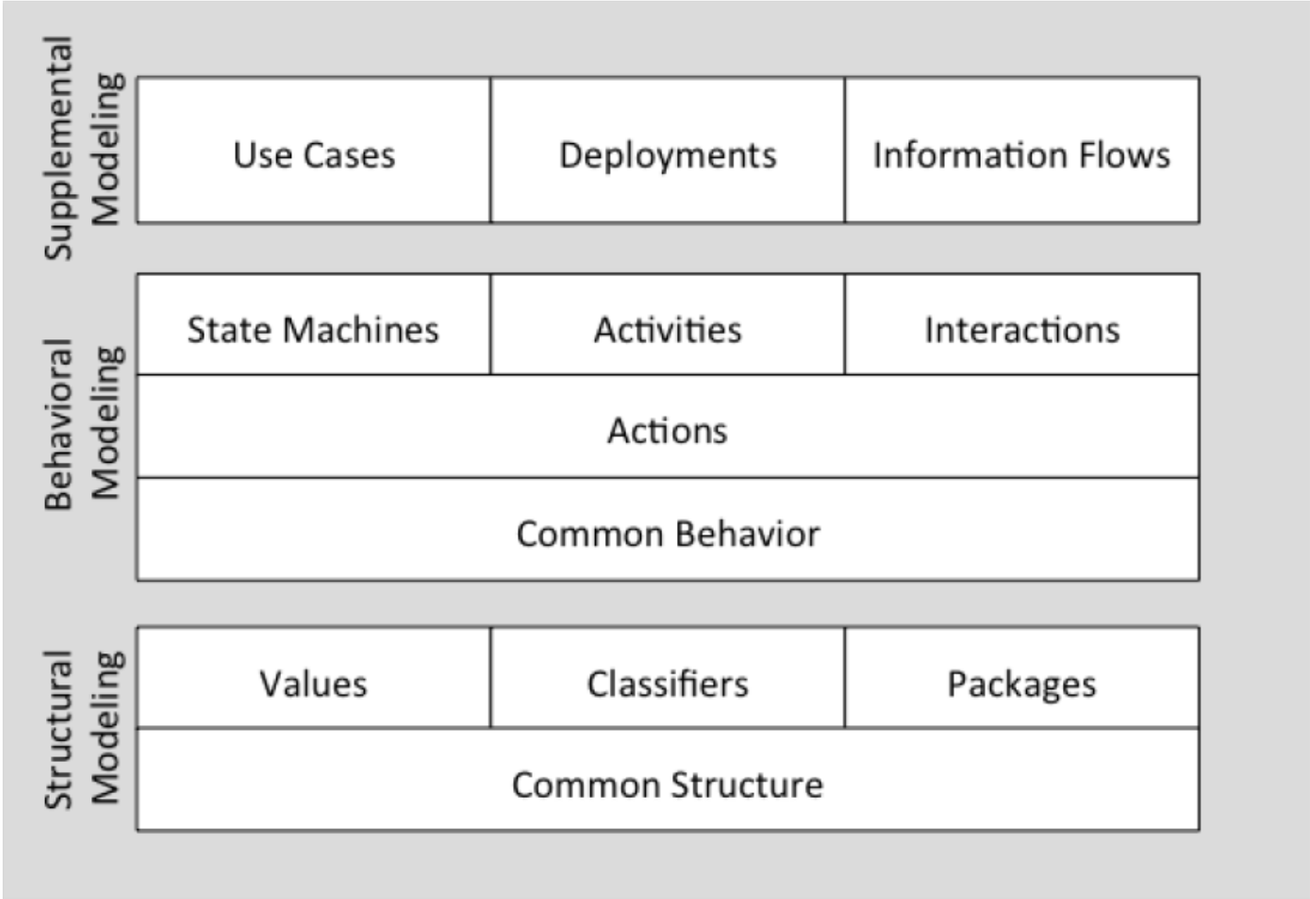
- Note, your sponsor may have a preference… Discuss it.

Text Requirements

- **SMART** – Specific, Measurable, Achievable, Relevant, Timeboxed
- Typical Software Requirements Specification template here →
- Provides a good overview of typical software requirement areas to consider
- Available in class files as srs_template-ieee.docx, many online

UML Diagrams

- Diagrams from the current UML release (https://www.omg.org/spec/UML/2.5.1/PDF)
- Structural
  - **Class**
  - Object
  - Package
  - Model
  - Composite Structure
  - Internal Structure
  - Collaboration Use
  - Component
  - Manifestation
  - Network Architecture
  - Profile
- Supplemental (both structural and behavioral elements)
  - **Use Case**
  - Information Flow
  - Deployment

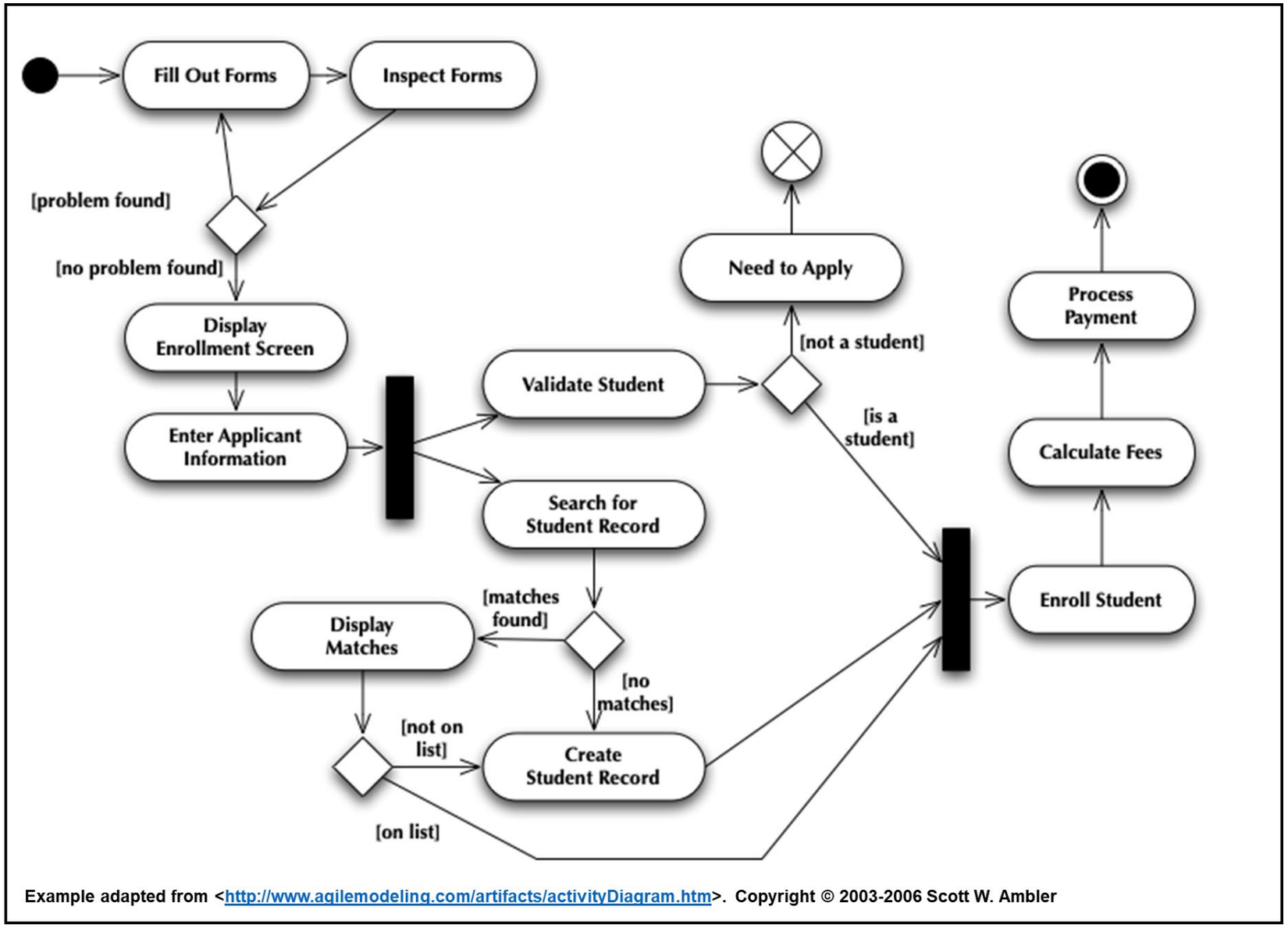


- Behavior
  - **Activity**
  - **Sequence**
  - **State (Machine)**
  - Behavioral State Machine
  - Protocol State Machine
  - Interaction
  - Communication (was Collaboration)
  - Timing
  - Interaction Overview
- Diagrams in **BOLD** are most often used

UML Activity Diagrams

Notation

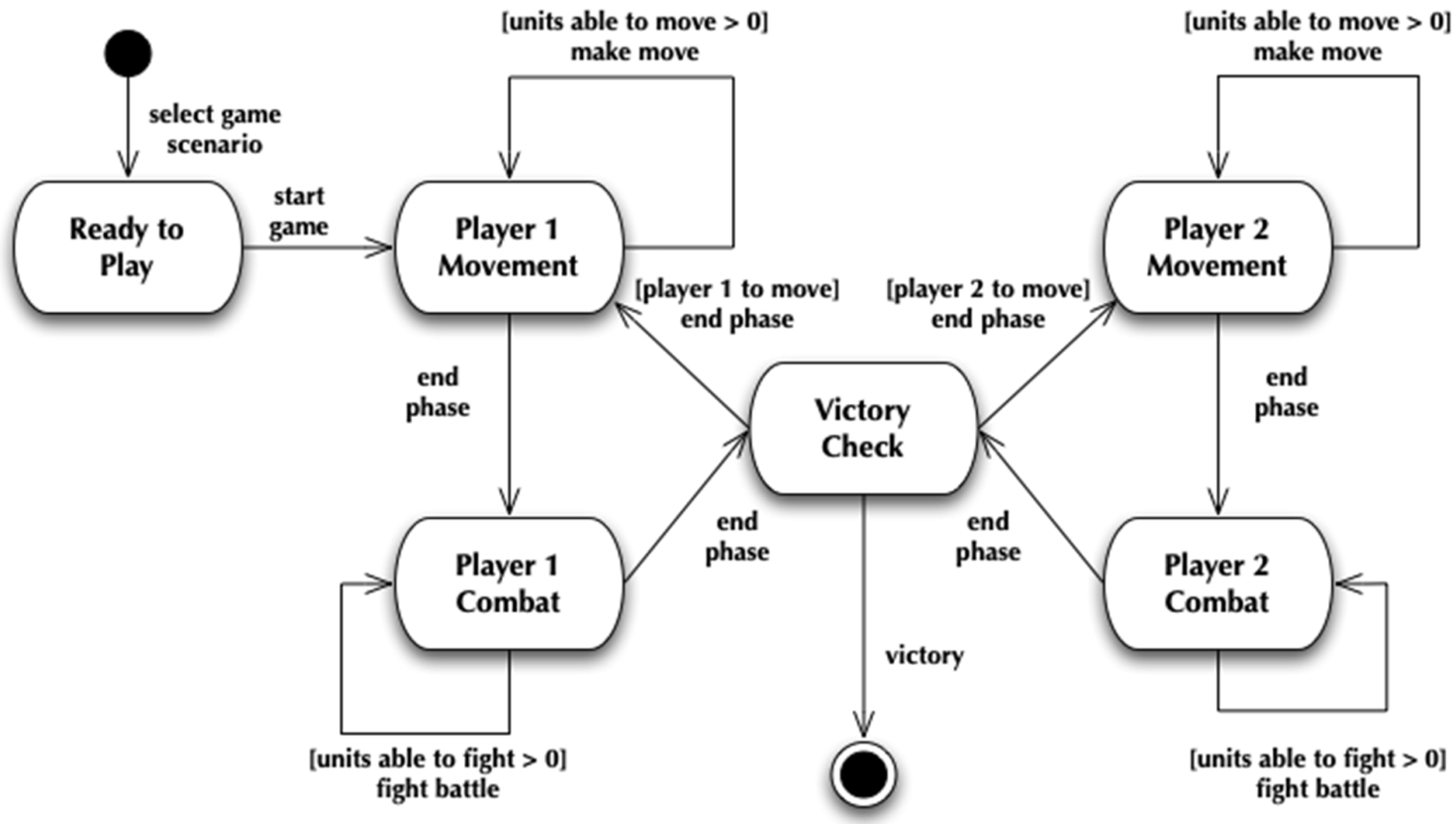
- Initial Node (circle)/Final Node (circle in circle)/Early Termination Node (circle with x through it)
- Activity: Rounded Rectangle indication an action of some sort either by a system or by a user
- Flow: directed lines between activities and/or other constructs. Flows can be annotated with guards "[student on list]" that restrict its use
- Fork/Join: Black bars that indicate activities that happen in parallel
- Decision/Merge: Diamonds used to indicate conditional logic.



Example adapted from <http://www.agilemodeling.com/artifacts/activityDiagram.htm>. Copyright © 2003-2006 Scott W. Ambler

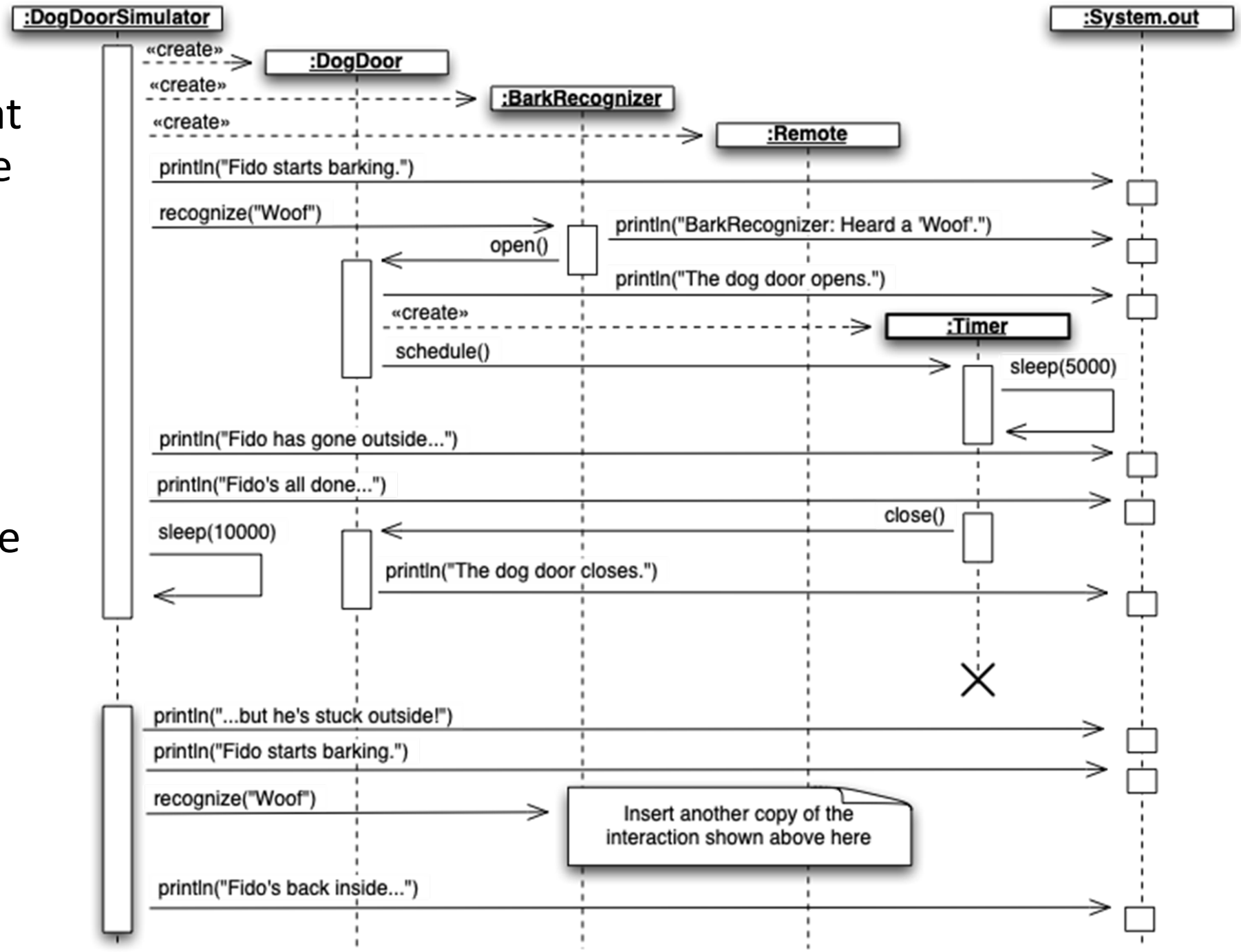
UML State Diagrams

- Each state appears as a rounded rectangle
- Arrows indicate state transitions
  - Each transition has a name that indicates what triggers the transition (often times, this name corresponds to a method name)
  - Each transition may optionally have a guard that indicates a condition that must be true before the transition can be followed
- A state diagram also has a start state and an end state



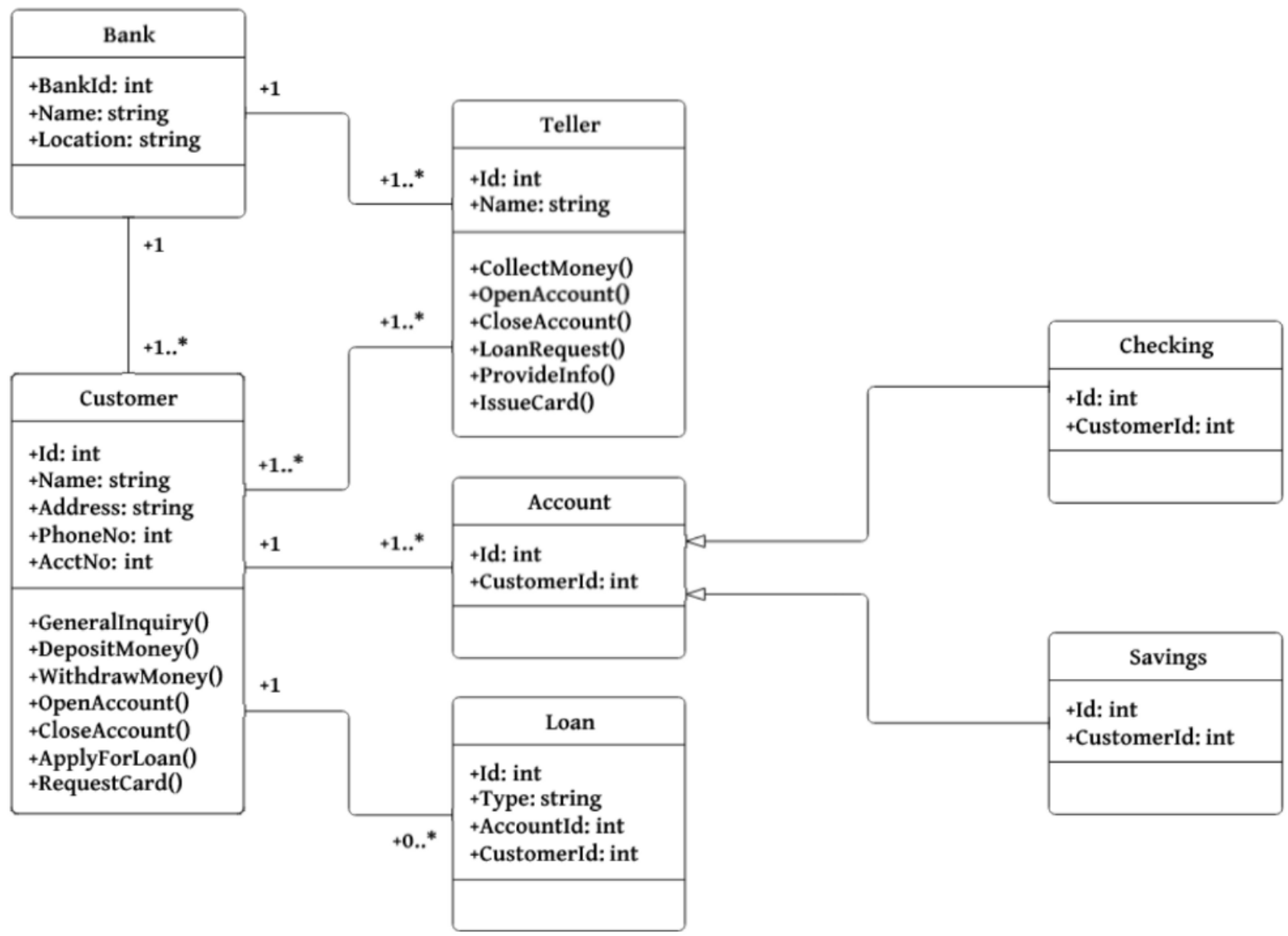
UML Sequence Diagrams

- Usually looking at object or module communications
- Most flow is to the right
- Arrows show messages
- Vertical boxes are object lifetimes
- X indicates an object is closed
- Can use text notes to make clearer



UML Class Diagrams

- Class Diagrams show
  - Inheritance, references, aggregation, composition
  - Class names, attributes, methods
    - Not usually constructors or destructors
  - Types
  - Multiplicity
  - Often highlight use of OO patterns
- Unlikely to create these during higher-level requirements work – usually for architecture and design



Text-based Use Cases

- Four benefits:
  - Short summary of system goals
  - Main success scenario (system responsibility)
  - Extension conditions (things to watch for or consider)
  - Extension handling (decisions on policy)
- From a presentation by Alistair Cockburn
  - Agile Use Cases
  - http://alistair.cockburn.us/get/2231

Robert Martin: "It shouldn't take longer than 15 minutes to teach someone how to write a use case!"

***Use case:* Text describing scenarios of user succeeding or failing to achieve goal.**

(goal of primary actory) (level of goal [summary, user, subfunction]) (primary actor)

"Place an order" **(User goal / Clerk)**

(action steps: full sentences showing who takes the action! 3 - 9 steps long.)

Main scenario:

1. **Clerk identifies customer, item and quantity.**
2. **System accepts and queues the order.**

(condition causing different actions)

Extensions:

(action step(s) handling those conditions)

1a. Low credit & Customer is 'Preferred':
System gives them credit anyway.

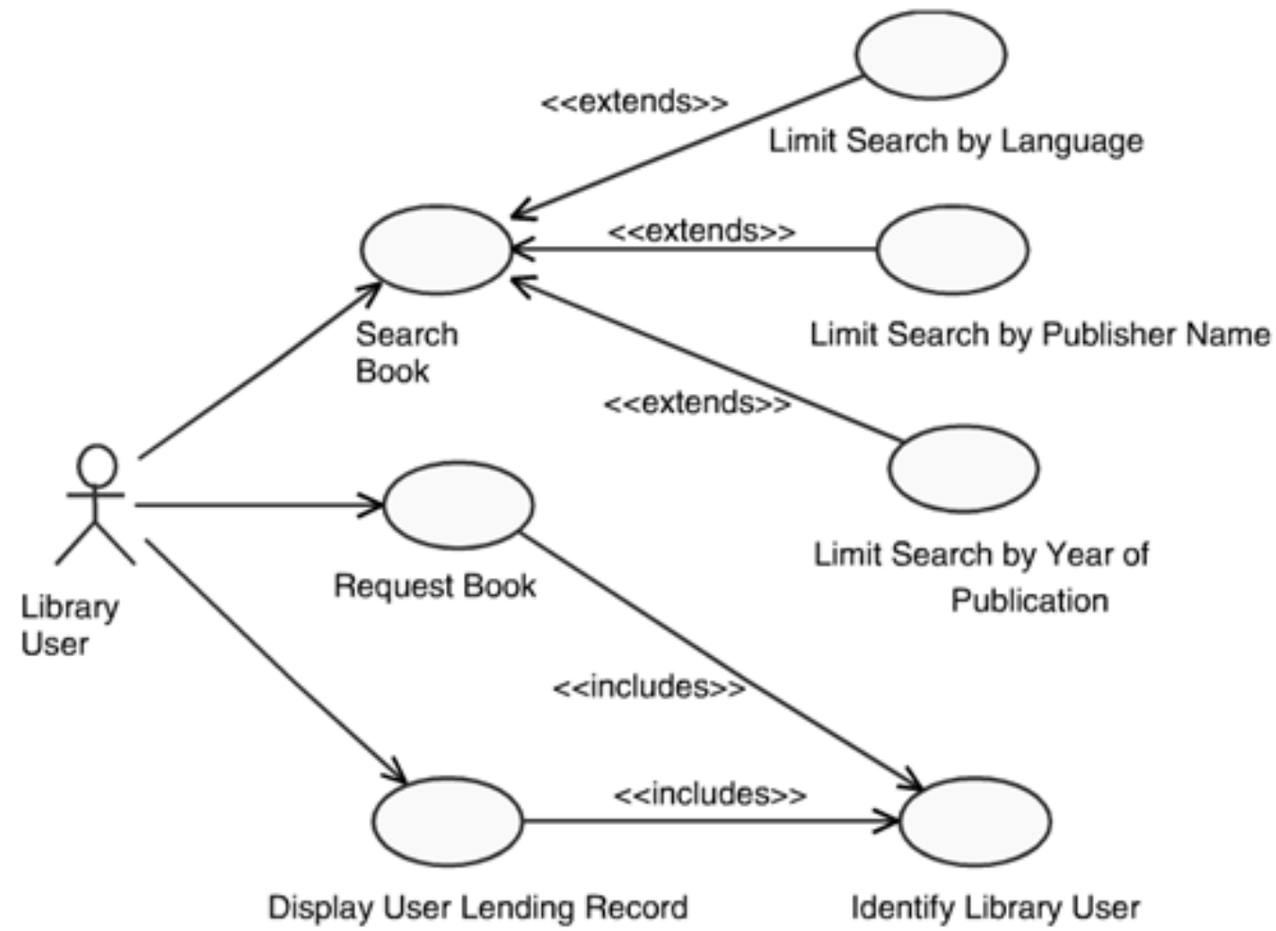
1b. Low credit & not 'Preferred' customer:
Clerk accepts only prepayment.

2a. Low on stock: Customer accepts rain-check:
Clerk reduces order to available stock level.

- Use Cases contain scenarios
  - A complete path through a use case from the first step to the last is called a **scenario** 22
  - Some use cases have multiple scenarios but a single user goal
    - All paths try to achieve victory

UML Use Cases

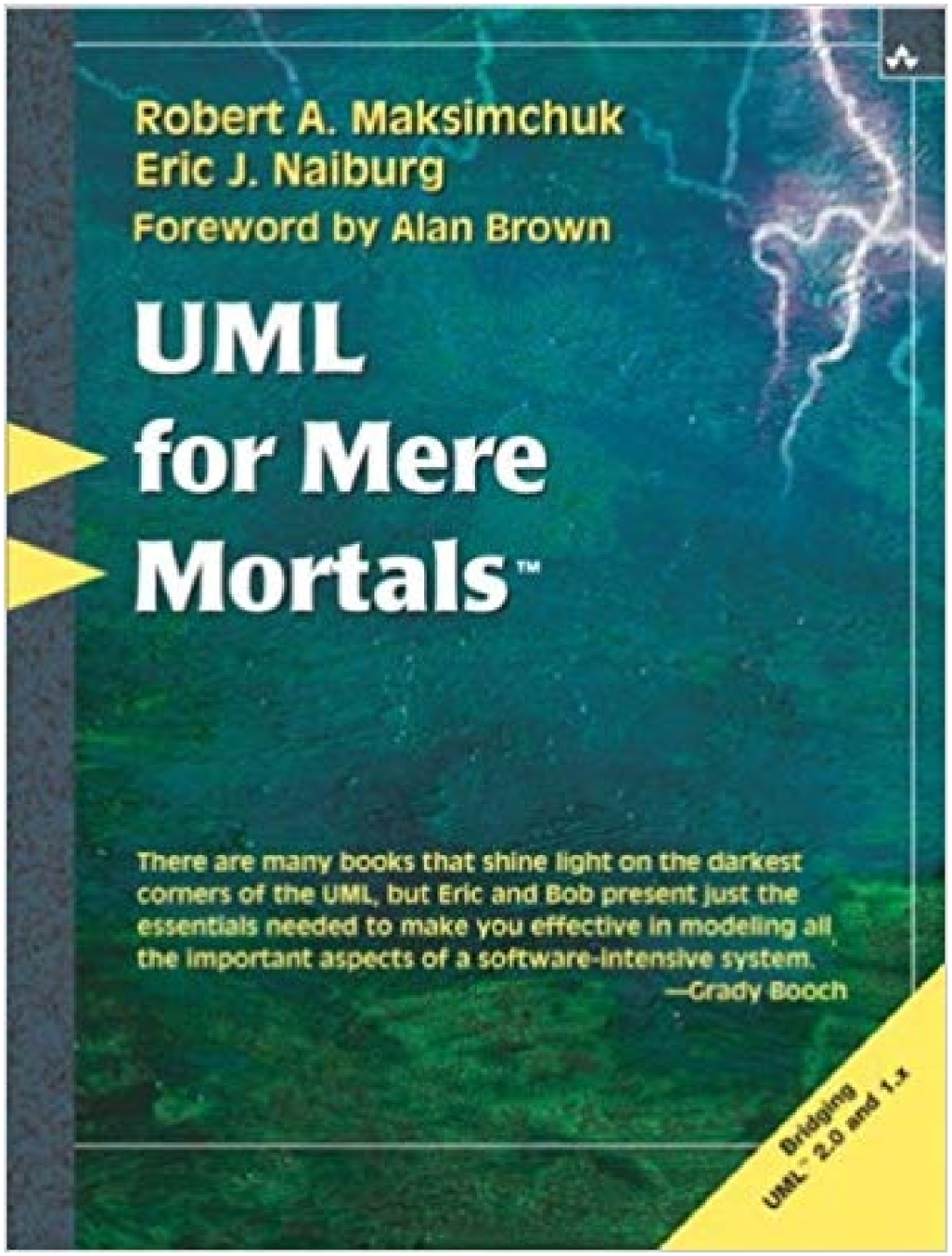
- Always design use cases from the actor's point of view
- Model the entire flow of a given operation
- For most systems, use cases should number in the tens, not hundreds
- <include> cases: not optional, base use case not complete without it, not conditional, and doesn't change the base use case behavior
- <extend> cases: Can be optional, not part of base use case, can be conditional or change behavior



WAVE Test for Use Cases (from Maksimchuk)
W: Use case describes WHAT to do, not how
A: ACTOR'S point of view
V: Has VALUE for actor
E: Use case models ENTIRE scenario

UML Tools

- References
  - Tutorials
    - https://www.tutorialspoint.com/uml/index.htm
  - Book
    - UML for Mere Mortals, Maksimchuk & Naiburg, 2005, Addison Wesley
- Tools
  - **Diagrams.net – has UML tools/templates (Free!)**
  - Lucidchart.com – UML Templates
    - (Free access available)
  - TopCoder UML Tool
    - sequence, class, use case, and activity diagrams
    - Free - Requires registration
    - https://www.topcoder.com/tc?module=Static&d1=dev&d2=umltool&d3=description
  - ArgoUML – open source
    - http://argouml.tigris.org/
  - Visio
  - Whiteboards and a phone/camera
  - Paper & pencil



Prototypes

- A prototype can be anything from a rough paper sketch to UI wireframes to a functional combination of hardware and software (low to high fidelity)
- In some cases you may want it to be testable
- In some cases you may want some level of interactivity
- Level of fidelity – how close to the real system
- Looks-like vs. works-like

- Best Practices
  - Remember prototypes are generally made to be disposable
  - Have a reason for making the prototype
  - Define scope and requirements for the prototype
  - Avoid interactivity unless absolutely needed
  - Test as the prototype is built
  - Keep expectations for the prototype realistic
- Proof-of-concept: Not a prototype, usually used to verify a tool or an element of an overall design

User Stories

- **INVEST** – Independent, Negotiable, Valuable (to user or customer), Estimateable, Small, Testable
- Sized less than one agile iteration/sprint, generally keep small to control estimate error
- Elements
  - What: Description of the feature
  - Who: Person who will use or benefit from the feature.
  - Why: “business value”
  - When: Depends on the priority
- Behavior Driven Design
  - As A – I Want – So That = narrative notation for user stories
  - Given – When – Then = notation to develop acceptance criteria
  - Cucumber – executable specifications (https://cucumber.io/tools/cucumber-open/)

The process I'd recommend

- Do the WBS and Requirements gathering in parallel
- The WBS is something your team should work on Top-Down, what are the big tasks, how do they break down, what things may also have to get done besides code
- The requirements should be done with your sponsor, interactively if possible
- Once you have data, formalize the WBS and requirements
- Requirements could be a combination of text, diagrams, and use cases
- These can lead to user stories later when the work for each sprint is planned
- The sponsor should review both the WBS and the requirements for signoff (between you and them)
- I'd like to see your first reviewed WBS and requirements from your teams by Wednesday 9/30

Task Tracking

- As your team starts to get things going, you'll want to track tasks for individuals
- For now, you could use a WBS to track work
- In a few weeks (unless your sponsor says otherwise), we'll want to shift to a Scrum rhythm assigning stories to individuals on the teams
- Potential tools
  - Trello – I have a Trello environment set up and can create a task board for your team (https://trello.com/en-US)
  - Asana – free for basic use (https://asana.com/)
  - Freedcamp – free for basic use (https://freedcamp.com)
  - Github Issues Tracking
- Whatever your sponsor would like you to use (and share)

Overall PMP Schedule

- Week 3: 9/7
  - Initial and final team assignments
  - Bruce will notify sponsors of team assignments
  - Bruce will send out NDAs for signatures – Due Wed 9/16 (one exception)
  - Teams should hold an initial meeting and discuss team roles
  - Charter and project briefs assigned/initial development
  - Thursday speaker (attendance tracked) – Amy & Rae
  - Request an introductory meeting for your team and the sponsor
- Week 4: 9/14
  - First meetings with sponsors – if you can share initial charters/project briefs all the better
    - Review any process, deliverables, or tool requirements they may have
  - First meetings with Bruce/Preethi for project status updates
    - Begin using status update forms, share with sponsors
  - Charter and project brief due, charter submitted for sign off by sponsors
    - Interim by 9/18, Final signed by 9/23

Overall PMP Schedule

- Week 5: 9/21
  - Start development of WBS & Requirements
- Week 6: 9/28
  - WBS & Requirements – pass 1 reviewed by sponsor
- Week 7: 10/5
  - WBS & Requirements (if needed, otherwise start Scrum sprints)
- Week 8: 10/12
  - WBS & Requirements – (if needed) pass 2 reviewed by sponsor
  - Midterm exam (take home)

Overall PMP Schedule

- Week 9: 10/19
  - Begin 1st 2 week Scrum sprint – Architectural/System Design
- Week 10: 10/26
  - Scrum
- Week 11: 11/2
  - Begin 2nd sprint – Design/Prototyping
- Week 12: 11/9
  - Scrum
- Week 13: 11/16
  - Begin 3rd sprint – Design/Prototyping
- Week 14: 11/24 (off 11/26-11/27)
  - Close 3rd sprint
- Week 15: 11/30
  - Final sponsor and in-class presentations
  - Assessments: Instructor, GSS, sponsors, peer
- Week 16: 12/7
  - Final exam (take home)

Communicating with Project Sponsors

- Please remember, if you communicate via e-mail, cc both Bruce and Preethi, to allow us to monitor written communications between teams and sponsors
- For our projects, the sponsor owns all Intellectual Property (IP) rights resulting from the master's project
- Do not discuss, reveal, or distribute project materials outside of your team, sponsor, and the class staff without express permission from your sponsor

Regular Project Status Meetings

- With Preethi
  - Inspiring - Site Rework
  - Helping - Web Integration
  - Double Helix - Microscopy Algorithms
  - Inovonics - ADL Algorithms
- With Bruce
  - Status - Conversational Interface
  - Status - Psychological State Algorithm
  - Trimble - IoT Network
  - Edwards - Network Simulation

- Should have a regular Zoom meeting for your entire team with Preethi or I scheduled weekly
- Attendance by whole team required, bring your status update for the week
- Discuss ANY issues you see in your team's progress!

Next Steps

- Speaker 9/24 – former Buff CS Grad student panel (tracking attendance) – during class hours
- Next Tuesday 9/29: tbd after team meetings – what you all need
- New Discussion Topics up on Piazza!
  - Please try to visit weekly for comments (and participation grades)
- Teams should be finishing Charter and Brief, starting WBS and Requirements
  - First pass due Wednesday 9/30, with sponsor review
- Regular meetings with sponsor should be set
- Aligned with sponsors on tools, project processes, deliverables
- Teams meet with me or Preethi 30 min each week (eventually, we'll get that to 15 minutes) (tracking attendance)
  - Started last week! Bring completed Project Status Forms, turn in weekly!
- Always cc Bruce & Preethi on sponsor e-mails
- Preethi and I are available for questions or other support