Other than OOAD?

CSCI 4448/5448: Object-Oriented Analysis & Design

Lecture 41

Acknowledgement & Materials Copyright

- I'd like to start by acknowledging Dr. Ken Anderson
- Ken is a Professor and the Chair of the Department of Computer Science
- Ken taught OOAD on several occasions, and has graciously allowed me to use his copyrighted material for this instance of the class
- Although I will modify the materials to update and personalize this class, the original materials this class is based on are all copyrighted © Kenneth M. Anderson; the materials are used with his consent; and this use in no way challenges his copyright

Before we start…

- Remaining OOAD classes
  - Mon 11/30 – Today's Lecture
  - Wed 12/2, Fri 12/4 – Graduate Pecha Kucha presentations
  - Mon 12/7 – Final brief OOAD Lecture – Final Exam Review, Class Wrap-up
  - Tue 12/8 – Reading Day, no classes
- Other assignments
  - Tue 12/1 – Pecha Kuchas must be submitted
    - For presentations, you can share slides on Zoom OR I can load and share them OR record your presentation
  - Project 6/Semester Project with recorded demonstration – due Fri 12/4 noon (adjusted late penalties, hard deadline of 12/7 midnight)
  - Quiz 11 this week – due Mon 12/7
  - Extra Credit/Participation Article Review [up to 2] – Open until Mon 12/7 at noon
  - Graduate Project final presentation – due Mon 12/7 noon
  - Final Exam (Online/Optional) – available starting Wed 12/9 morning through Thursday midnight

Before we start: Discussion Topics Revisited

1. Class Questions
   - Do let me know in Quiz 11 about any changes for OOAD you'd recommend, I do pay attention and make changes
2. Languages
   - Python is my first choice, have used Node.JS, R, C, and Matlab/Octave lately
3. Source Control
   - Git (& Git Bash) is my go-to, haven't worked in a group shared repo for some time
4. Design-First
   - Unless it's a simple piece of code, I'll use UML to sketch out the system, especially use cases
5. Favorite References
   - I'm a book guy (as you know), see the list at the end of this lecture
6. Team Projects
   - Last time I worked in industry, I drove Scrum and Kanban based s/w teams; I think a good agile process makes for a happier & productive experience
7. Best Software/System
   - Longest-lived: FireWorks by EST (started in 1998, still in use)
   - Most complex: SimPy based network communication simulation
8. What Went Wrong
   - Usually people issues; me or others
9. Testing
   - Honestly, not a good tester; haven't used TDD personally; more test after/with; put a test department in place at last position
10. Best Mentor, Leader, Manager, Peer
    - Been lucky to have a few folks who helped focus my career a little sharper or gave me an honest word or two when I needed it

Goals of the Lecture

- Review software language landscape
- Survey development processes other than OOAD
- The state of OO and software
- Some other sources to review…

Languages…

- The TIOBE Index – which language has been used for the most lines of code… Of ~700 Programming Languages… https://www.tiobe.com/tiobe-index/

1. Java
2. C
3. Python
4. C++
5. C#
6. Visual Basic
7. JavaScript
8. PHP
9. SQL
10. R
11. Swift
12. Go
13. Ruby
14. Assembly
15. PL/SQL
16. Perl
17. Objective-C
18. MATLAB
19. Classic Visual Basic
20. Scratch

Object Oriented Languages…

- Object-oriented – at least in part

1. Java
2. C
3. Python
4. C++
5. C#
6. Visual Basic
7. Javascript
8. PHP
9. SQL
10. R
11. Swift
12. Go
13. Ruby
14. Assembly
15. PL/SQL
16. Perl
17. Objective-C
18. MATLAB
19. Classic Visual Basic
20. Scratch

?. Kotlin

One of the reasons for the OOAD class…

Most development is in OO languages, whether used in true Object fashion or not.

What about the others?

- C (and Pascal, Rust, etc.) – Imperative, Procedural
- SQL and PL/SQL – Domain Specific for Databases, Procedural
- Assembly – Microprocessor dependent, functional/procedural?
- R (and Mathematica, MATLAB) – Domain Specific for Math/Statistics, primarily Procedural
- Scratch – Educational visual programming

- Not Listed:
- F#, Erlang, Haskell - Functional
- Lisp – Functional, used for AI
- APL – Symbolic language for mathematics
- Prolog – Logic/goal programming
- Etc…

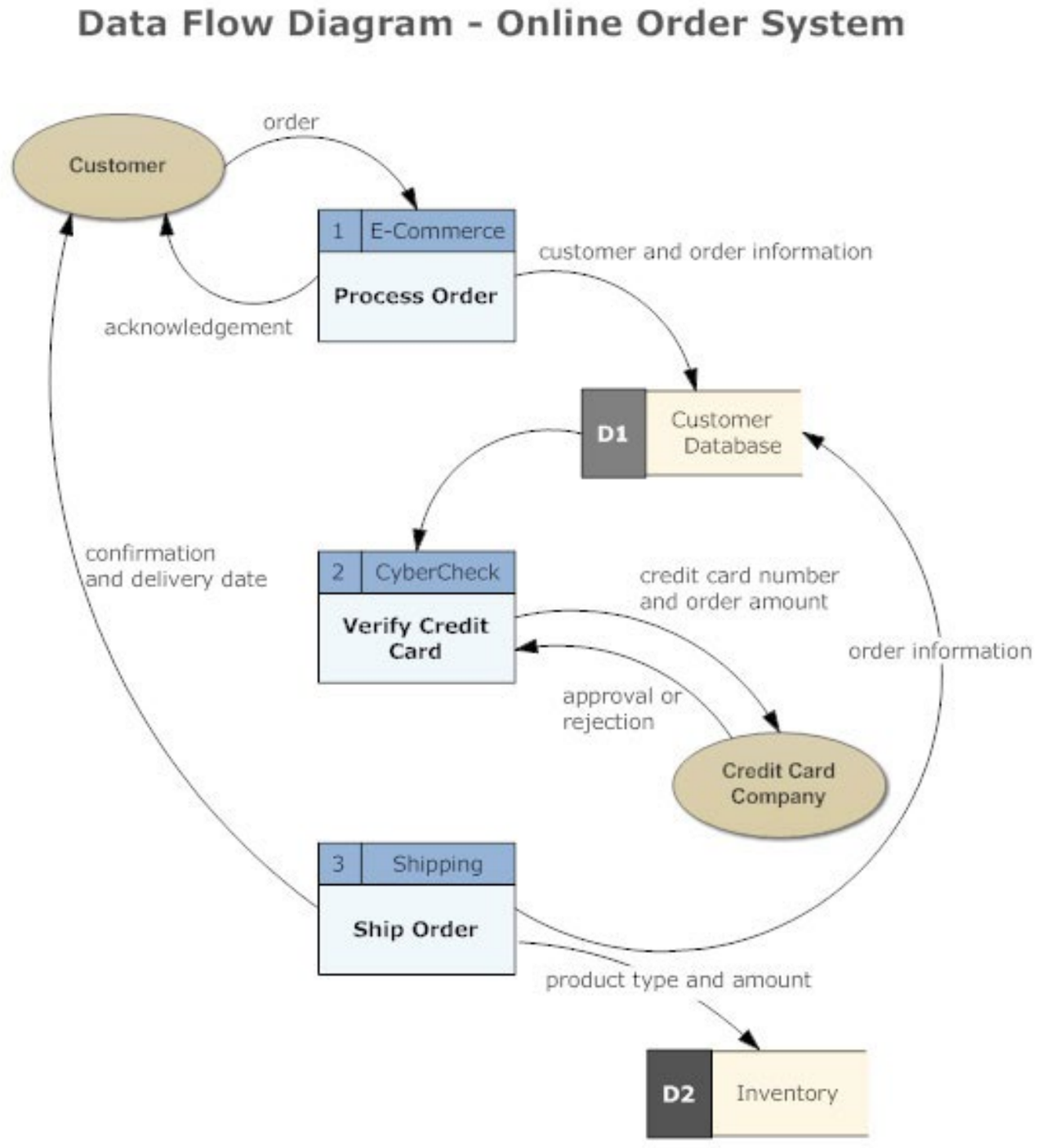
If not OO, what?

- If not all languages are OO…
- What approaches are there for Analysis and Design besides OOAD?

- Data Flow Oriented Design
- Structured Analysis and Design Technique (SADT)
  - IDEF0 through IDEF5
- Cleanroom Software Development
- Dynamic Systems Development Method (DSDM)
- Test Driven Development (we've reviewed… doesn't replace OO)
- Domain Driven Design (DDD)
- Behavior Driven Development (BDD)

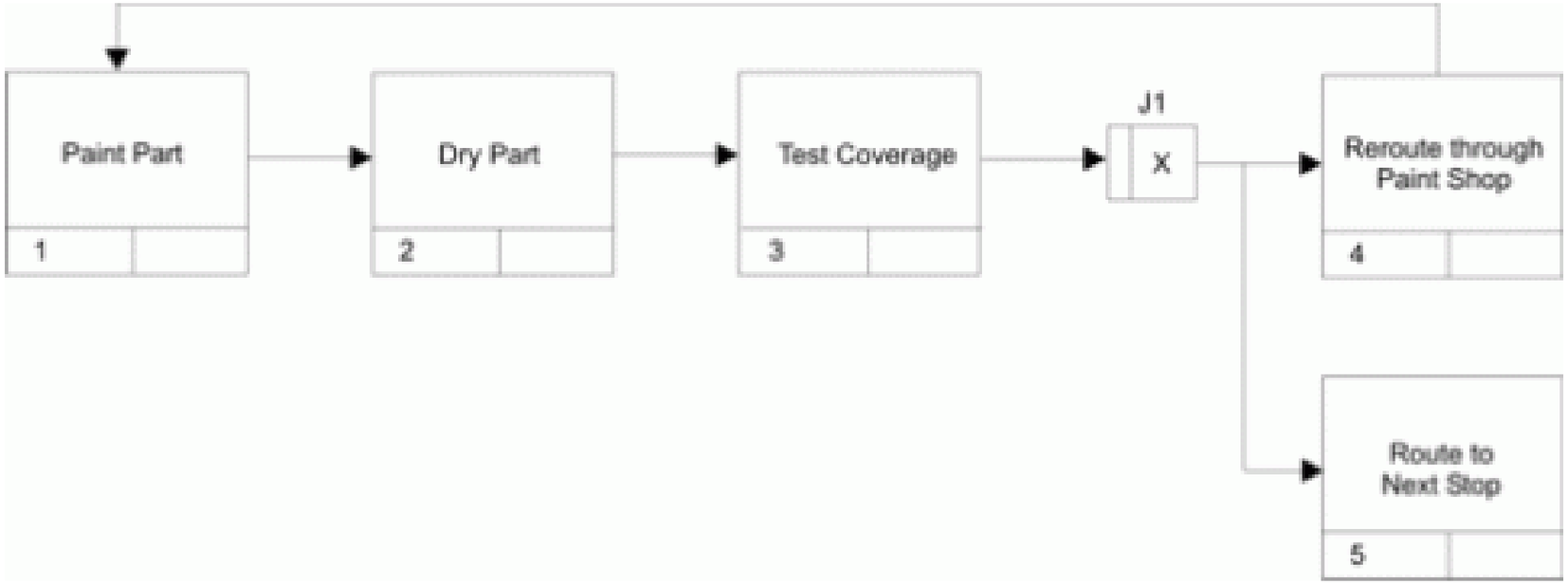
Data Flow Oriented Design

- Data Flow Diagrams (we mentioned these before as a design technique…)
  - From Structured Design in 1970s
- Data Flow Diagrams aren't flow charts
- Process of Data Flow design
  - Trace the "main path" from the primary input to the primary output
  - Identify the most abstract input (MAI) and most abstract output (MAO) points
  - Top module decomposes into "get MAI", "MAI-->MAO", and "put MAO" (with actual names)
  - Recursively decompose "get MAI" modules into "get next MAI" and "next MAI-->MAI" modules
  - Recursively decompose "put MAO" modules into "MAO-->next MAO" "put next MAO" and modules
  - Apply this entire process recursively to the "MAI-->MAO" modules
- Plus: A repeatable/cookbook like procedure
- Minus: May not scale up or work with non-hierarchical architectures
- https://www.cs.odu.edu/~price/cs451/Lectures/05design/dataflow/
- https://www.smartdraw.com/data-flow-diagram/



SADT – Structured Analysis and Design Technique

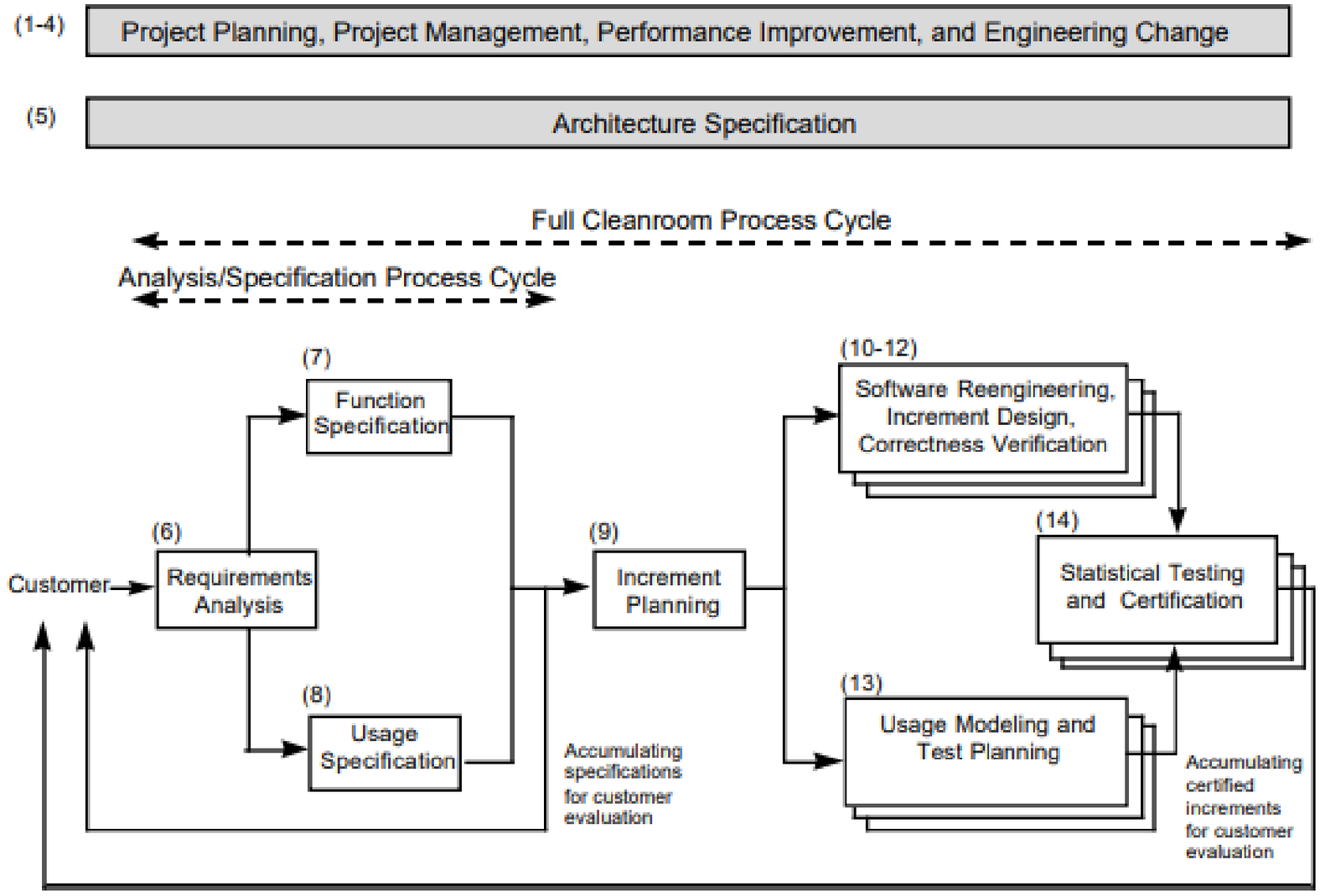
- Starts with Ross in late 60s, formalized as IDEF0 (Integrated DEFinition for Process Description) in 1981
- Later: IDEF3 for business process models, IDEF5 for ontology description capture (showing properties and relations between concepts)



- https://www.idef.com/idef3-process-description-capture-method/

Cleanroom Software Engineering

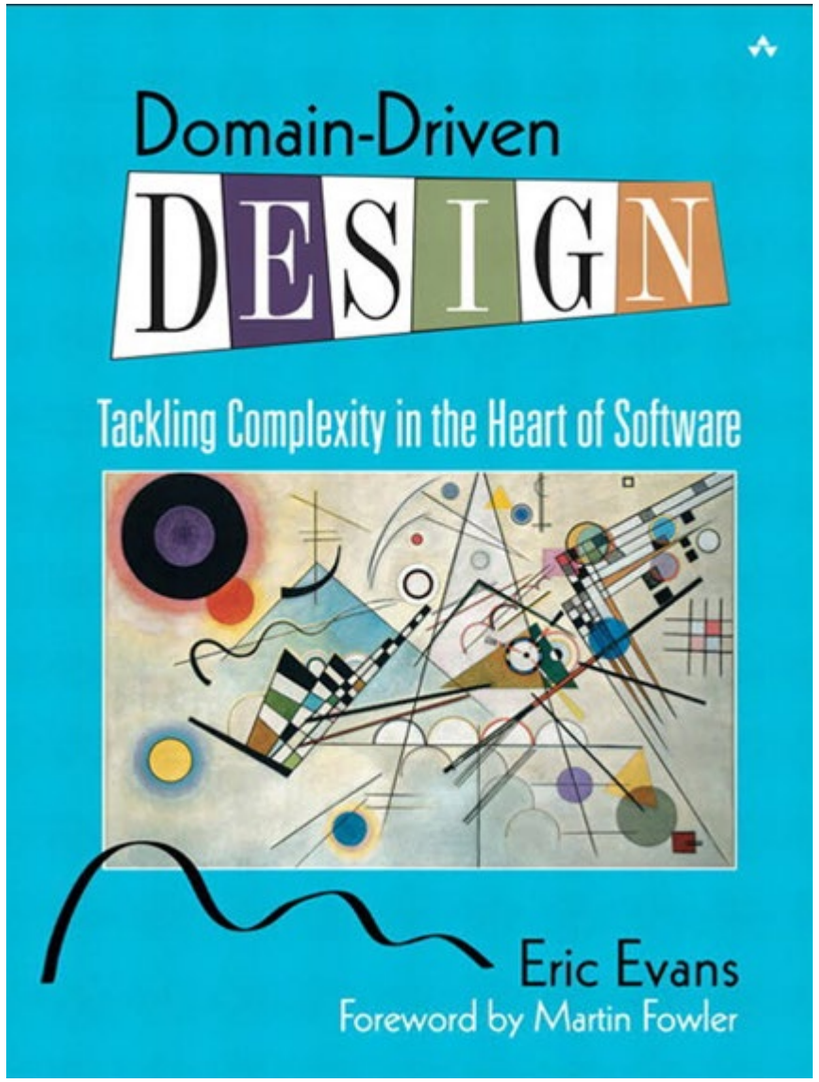
- Development of high-reliability software systems under statistical quality control
- Key objective is zero failures in use
- Includes formal methods and usage models
- Includes full life cycle elements for design, test, and management
- https://resources.sei.cmu.edu/asset_files/TechnicalReport/1996_005_001_16502.pdf



Dynamic Systems Development Method (DSDM)

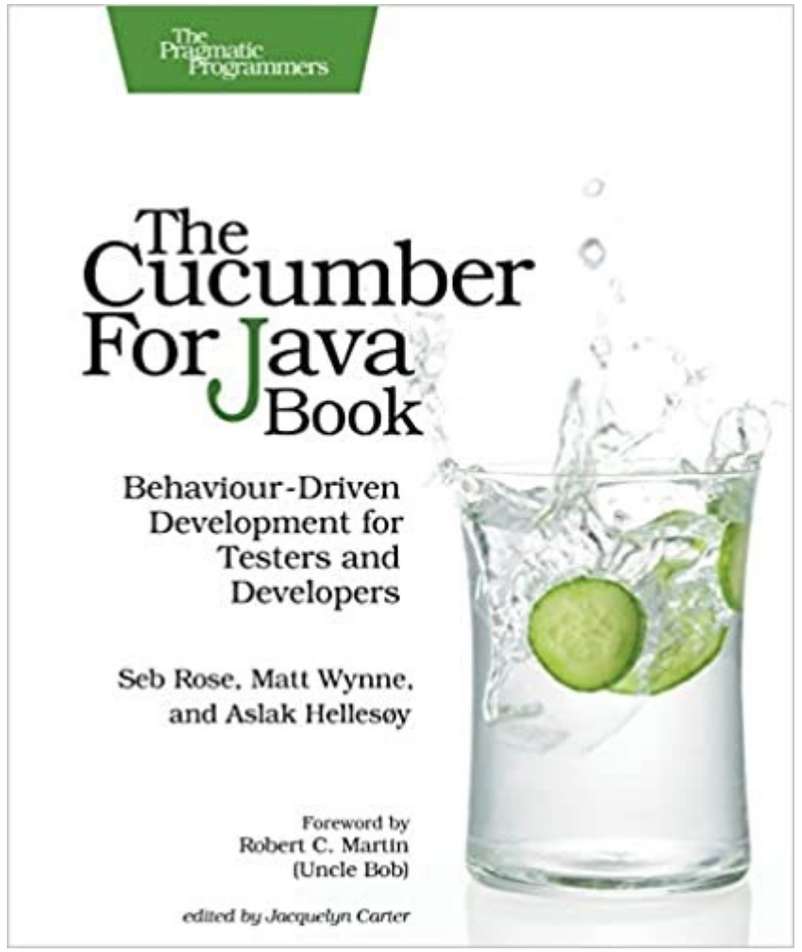
- More a project management methodology…
- An agile method for projects to strengthen the earlier RAD (Rapid Application Development)/JAD (Joint Application Development) methods
- Eight Principles
  - Focus on the business need
  - Deliver on time
  - Collaborate
  - Never compromise quality
  - Build incrementally from firm foundations
  - Develop iteratively
  - Communicate continuously and clearly
  - Demonstrate control
- Can mix with Scrum or other project management approaches
- https://www.agilebusiness.org/page/whatisdsdm
- https://www.agilebusiness.org/page/TheDSDMAgileProjectFramework

Domain-driven Design (DDD)



- Term coined in Evans 2004 Book – widely praised
- (Business) Domain-driven (Software) Design
  - Develop software for complex needs by connecting implementation to a model of core business concepts
  - Primary focus on core domain and domain logic and a model to drive complex designs
  - Technical and domain experts work to iterate to the conceptual core problems
    - https://dddcommunity.org/learning-ddd/what_is_ddd/
- Strategic Design
  - Business Domains – areas of activity or services
  - Ubiquitous Language – speak in terms of the business domain
  - Context Maps and Boundaries – designs for model subdivisions and interfaces
- Tactical Design
  - Business Logic Patterns – Transaction Scripts, Active Records, Domain Models
  - Architecture Patterns – Layered, Ports & Adapters, Command Query Responsibility Segregation (CQRS)
- Event Storming
  - Structured group modeling technique for the business process – find domain events, users, business processes, commands, aggregates, external systems, and views

Behavior Driven Development



- First came from Dan North around 2009
- Refinement of Test Driven Development and Acceptance Test Driven Development…
  - Apply the “Five Why’s” to user stories to make their purpose match business outcomes
  - Minimize waste by only implementing elements needed for business outcomes
  - Drive these approaches down to lowest abstraction levels in software, distribute behavior to make evolution easier
- BDD aka Specification by Example
  - As A-I Want-So That = narrative notation for user stories
  - Given-When-Then = notation to develop specification of acceptance criteria
  - Three Amigos = Business, Developer, Tester work in Specification Workshops
- Cucumber – a tool for developing requirements as shown above
- https://www.agilealliance.org/glossary/bdd

Is OOAD alive or dead?

- Many of the processes we just looked at are using OO elements in their modeling or process discovery
- Majority of most used languages are OO – it's not leaving us any time soon
- Procedural/functional languages still have a place
  - But will struggle with increases in size and change
- Like most tools, OO is dangerous in the wrong hands
  - To do it right takes some work and study, not usually appreciated by casual coders and scripters
- Learn from other's OO experience – patterns, code smells, refactoring, test approaches
- Use strong OO design for test, maintainability, and change
  - Leave a clean campsite for the next camper!

Where do we go from here? Software Trends

- One person's view (but I don't disagree much)…
- All about the cloud – leveraging services, tools, and infrastructure (with the IoT) – AWS, Google, Azure
- Containers – Docker and Kubernetes are key technologies
- Microservice Architectures – vs. large monolithic systems (almost OO at larger scale)
- OO rules: Python in general, Java (with Spring) for enterprise scale
  - Still C for embedded systems
  - Watch the newer languages – Rust, Swift, Kotlin, TypeScript…
- For the Web and Cross Platform Apps – JavaScript and React (Angular, Vue also)
- Native App Development – Kotlin/Android, Swift/Apple/iOS
- APIs = REST
- Databases – top choices are still SQL based
  - Some roles for NoSQL: Mongo, Redis, Cassandra
- Agile development processes and DevOps continuous test/integration
- Be user aware – UX and UI design are key skills
- https://towardsdatascience.com/20-predictions-about-software-development-trends-in-2020-afb8b110d9a0

Further Review - Videos

- Agile Product Ownership in a Nutshell
  - Favorite video for explaining Agile best practices and key elements in 15 minutes
  - https://www.youtube.com/watch?v=502ILHjX9EE
- Martin Fowler
  - A popular speaker, he has a list of videos at his site https://martinfowler.com/videos.html
  - Check out “Not Just Code Monkeys”: https://www.youtube.com/watch?v=Z8aECe4lp44
- Steve McConnell
  - Lots of videos out there from his training company Construx
  - “Seven Unbreakable Rules of Software Leadership”
  - https://www.youtube.com/watch?v=iW3ED4UxVgA
- Rich Sheridan
  - Lots of TED type talks on software related topics
  - https://www.youtube.com/watch?v=fj7QK-hOBDI

From a CS book guy…

- Code Complete 2 – Steve McConnell
  - Also Clean Code – Robert “Uncle Bob” Martin (the other “Clean” books are good too, esp. Clean Architecture)
- Patterns of Enterprise Application Architecture, Refactoring, UML Distilled – Martin Fowler
- Working Effectively with Legacy Code – Michael Feathers
- A good OO Pattern book – Head First is fine, might want one for your language
- A good TDD book – I like TDD for Embedded C by Grenning
  - You might want something specific to another language or a general book like The Art of Unit Testing – Manning or TDD by Beck
- A good UX book or two
  - UX Team of One – Buley; Don’t Make Me Think - Krug
- The free Pro Git – Chacon & Straub; Git for Teams – Westby
- The Pragmatic Programmer: Journeyman to Master – Hunt & Thomas
  - “Why spend your life developing software unless you care about doing it well?” —Andrew Hunt and Dave Thomas
- Joy Inc. and Chief Joy Officer – Rich Sheridan

Next Steps

- Last Quiz (11) open this week through Mon 12/7 – just a class survey – free quiz grade if you complete it
- Wed/Fri: Pecha Kuchas!
- Project 6 (Semester Project submission/demo) – Friday 12/4
- Bonus points are posted; Extra Credit/Participation Article Review (s) [up to 2] – Mon 12/7
- Graduate Project Final Presentation
- Final Exam (Online/Optional) – starting Wed 5/9 on Canvas
- See the class staff for anything you need
- Hang in there, it's almost done…