ECE4574 – Large-Scale SW Development for Engineering Systems
Lecture 1 – Introduction / SW Development

Creed Jones, PhD

Topics for Today

- Introduction to the Course
  - Syllabus
  - Course Schedule
- Software for Engineering Systems
- What is a Software Development Methodology
  - Waterfall Methodology
  - Agile Methodologies

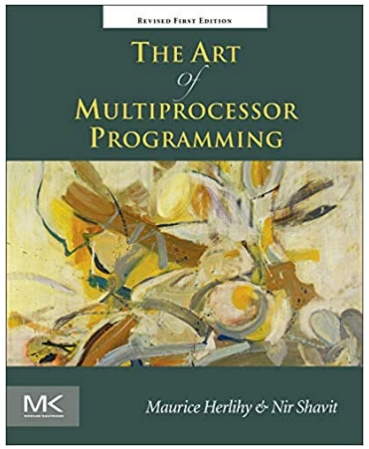
Required Resources

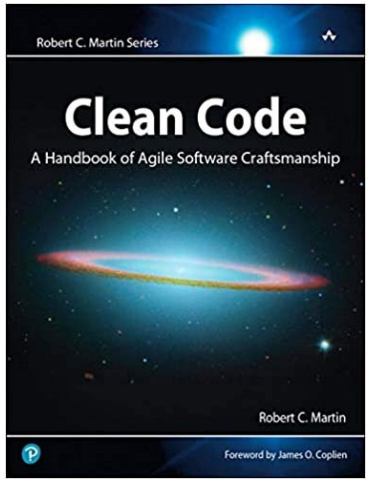
Software:

- Qt and Qt Creator (I use it on Windows 11)
- Microsoft Visual studio for C++ (I use VS Code)
- Other SW development environments as required by your project

Attendance at Class Sessions:

- M W 2:30–3:45 p.m. in 349 Whittemore Hall
  - Or virtually for the online section





Recommended Resources

Textbooks:

<u>No textbook is required</u>, but you might find these (and others) useful to look at…

- Herlihy and Shavit, *The Art of Multiprocessor Programming*, Morgan-Kaufman; 2012, ISBN-10: 0123973376, ISBN-13: 978-0123973375
- Robert Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*, 1st edition, Pearson, 2008, ISBN-10: 9780132350884, ISBN-13: 978-0132350884
- Ivan Mistrik, *Software Architecture for Big Data and the Cloud*, 1st edition, Morgan Kaufmann, 2017, ISBN-10: 0128054670, ISBN-13: 978-0128054673

So what's this course about? Let's look at the catalog description

- Large-scale software implementations of the hierarchy of engineering analysis, design, and decision evaluation.
- Computer-aided engineering programs with state-of-the-art computer tools and methods.
- Operator overloading, dynamic polymorphism, graphical user interfaces, generic programming, dynamic link libraries, and multiple threads.

ECE4574 Course Objectives; what skills should you acquire from this course?

Upon successful completion of this course, students will be able to:

1. Develop large-scale, event driven programs for engineering systems
2. Demonstrate ability to work with frameworks
3. Design and implement multi-threaded, runtime modular programs
4. Design and implement graphical user interfaces

Student Assessment

- There will be three homework assignments, a final exam, a team project assignment and eight quizzes throughout the semester. Each assignment is due at 11:59 PM on the due date. Homework assignments (only) will be accepted up to three days late, but will be penalized 10% per day.

Graded Item	# of Items	Points per Item	Total Points	Percentage	Indiv/Team?
Homework Assignments	3	25	75	18.75%	I
Final Exam	1	100	100	25.00%	I
Quizzes	8	10	80	20.00%	I
Project Topic	1	25	25	6.25%	T
Sprint Reports	4	10	40	10.00%	T
Final Project Presentation	1	30	30	7.50%	T
Final Project Report	1	50	50	12.50%	I
			400	100%	

Academic Integrity

- **<u>All homework assignment and exams must be your own work</u>**

- Cheating: copying another student's work on any assignment
- Plagiarism: quoting or using content from a published source without proper citation

- Cheating or plagiarism will result in a zero on the assignment in question
- The university policies for academic integrity and the honor code will be followed

	Module	Lec	ECE4574 FA22 Daily Schedule Topics	Due	Project
21-Aug	I - SW Development	1	Introduction - SW development methodology		
23-Aug		2	Scrum		
28-Aug		3	Object-oriented design		
30-Aug		4	OO system design and architecture	quiz 1	
4-Sep			*No Class - Labor Day*		
6-Sep	II - Distributed Computing	5	Networks		
11-Sep		6	Frameworks and middleware	quiz 2	Topic Due
13-Sep		7	Middleware, message brokers and MQTT		
18-Sep		8	Web services		Sprint #1
20-Sep		9	Cloud computing	HW1 - quiz 3	
25-Sep		10	Cloud implementation		
27-Sep	III - SW Design	11	UML		
2-Oct		12	Design patterns	quiz 4	
4-Oct			Project Day		
9-Oct		13	Design patterns for engineering calculations	Sprint 1	Sprint #2
11-Oct	IV -SW Architecture	14	Architecting a component; careers	HW2	
16-Oct		15	Architectural issues for engineering systems	quiz 5	
18-Oct		16	Thoughts on large system design		
23-Oct		17	Cloud-based architectures		
25-Oct			Project Day		
30-Oct	V - Events and Threads	18	Event driven programming and messages	Sprint 2	Sprint #3
1-Nov		19	Events, Threads and Java	quiz 6	
6-Nov		20	Multi-threaded programming		
8-Nov		21	Run-time Modular programs	HW3	
13-Nov	VI - Wrapup	22	Developing, testing and debugging code	quiz 7	
15-Nov			Project Day		
20-Nov			*No Class - Thanksgiving Break*		
22-Nov					
27-Nov		23	Advanced Language Features	Sprint 3	
29-Nov			Project presentations	quiz 8	PRESENTATIONS
4-Dec			Project presentations		
6-Dec			Project presentations	Final rpt	
9-Dec	**FINAL EXAM (Tuesday, December 9, 10:05 AM to 12:05 PM)**				

VIRGINIA TECH

BRADLEY DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Our semester-wide project is a centerpiece of the course

- Work in teams of three or four
  - You can choose your own team, by August 30
  - On August 31, everyone else is added to a team
- You will use an agile development process, with three sprints
  - Don’t worry, I will explain it all
- Important due dates:
  - Sept. 11: project proposal with initial requirements
  - Oct. 9: sprint 1 wrapup report
  - Oct. 30: sprint 2 wrapup report
  - Nov. 27: sprint 3 wrapup report
  - Nov. 29 – Dec. 6: brief project presentation
  - Dec. 7: final report

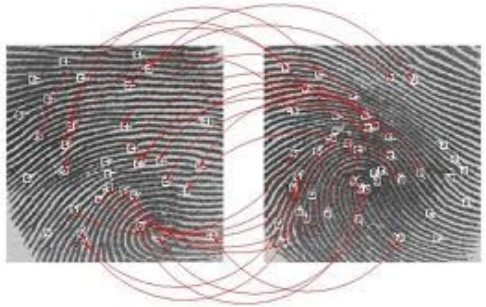
The project topic proposal and initial requirements are due on September 11!

- Topic proposal
  - A software engineering application
  - Not too hard, not too easy
  - For FA23, somehow related to **some field of engineering**
- Initial list of requirements
  - I will give you examples
- In grading your proposal, I will let you know of any needed changes
  - whether large or small

Dr. Creed Jones, Professor

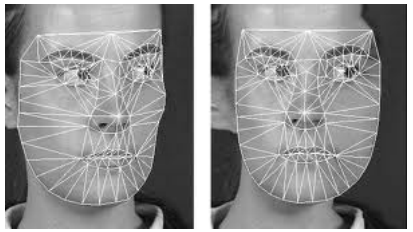
BS & MS in Electrical Engineering from Oakland University (Michigan)

PhD in Computer Engineering from Virginia Tech
Research - Automated Face Recognition



- Worked for:
  - Small technology companies in Image Processing
  - Sagem Morpho in Biometrics (Human Identification)
  - Humana in Pattern Recognition and Analytics

Humana.



Engaging the culture, changing the world.
Seattle Pacific UNIVERSITY

- Taught Computer Science at Seattle Pacific University and California Baptist University
- Hold ten patents; authored a number of international standards
- Founding partner in Globe Biomedical, a medical device startup in Riverside, CA

CBU

crjones4@vt.edu

GLOBE BIOMEDICAL

You can ask questions during lecture, office hours, by email or through Piazza

Office hours will be held both in-person and online

- M 1:00 PM – 2:00 PM, T Th 2:30 PM – 4:00 PM, W 10:00 AM – noon
- In 462 Whittemore, or via Zoom, ID = **705 806 0713**

You can also reach me by email – put "4574" in the subject line!

We will use Piazza as a forum for questions

- The system is highly catered to getting you help fast and efficiently from classmates and myself. I encourage you to post your questions on Piazza. You will see Piazza in the navigation bar in Canvas

How can you succeed in this course?

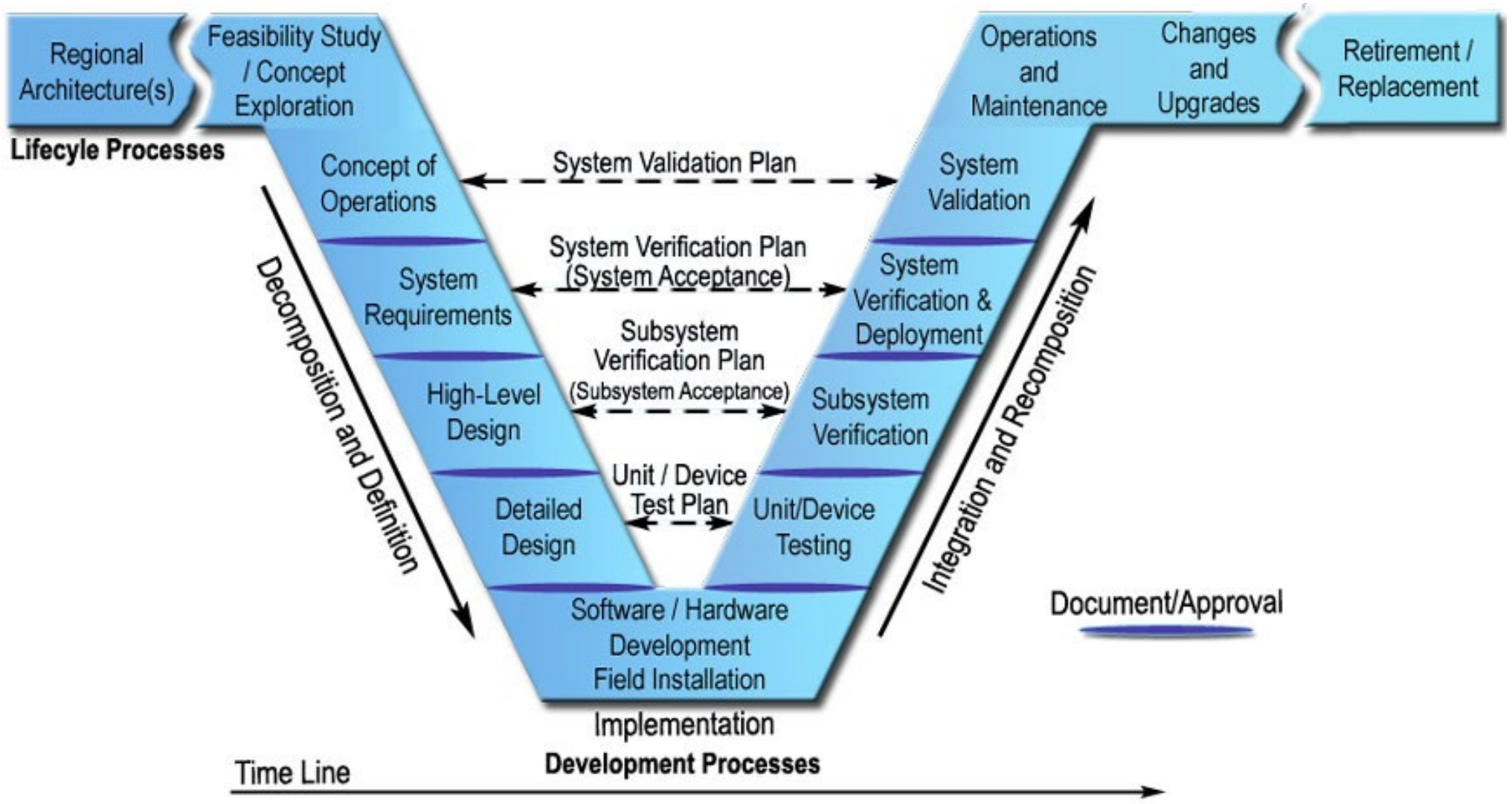
- Don’t miss any assigned work!
  - Homework assignments
  - Quizzes
  - Project milestones
  - Final Exam
- There is an allowance for late work on homework assignments – try not to use it!
  - You will fall behind and have to work harder to catch up
- Maintain a steady pace of work on the project
- Ask questions!

SOFTWARE FOR ENGINEERING APPLICATIONS

"Software for Engineering Systems" describes many different things

- Software Applications used in intensive engineering work
  - MATLAB, libraries like scikit and OpenCV
- Generally used applications with heavy math/technical/performance requirements
  - imaging, networking, multimedia
  - security and privacy
- Embedded systems
  - controls, aerospace, IoT, systems with severe performance/power constraints
- Control systems
  - manufacturing, autonomy, automation, safety

Note that the above covers the majority of (interesting) software written today



Caltrans and USDOT. 2005. Systems Engineering Guidebook for Intelligent Transportation Systems (ITS), version 1.1. Sacramento, CA, USA: California Department of Transportation (Caltrans) Division of Reserach & Innovation/U.S. Department of Transportation (USDOT), SEG for ITS 1.1.

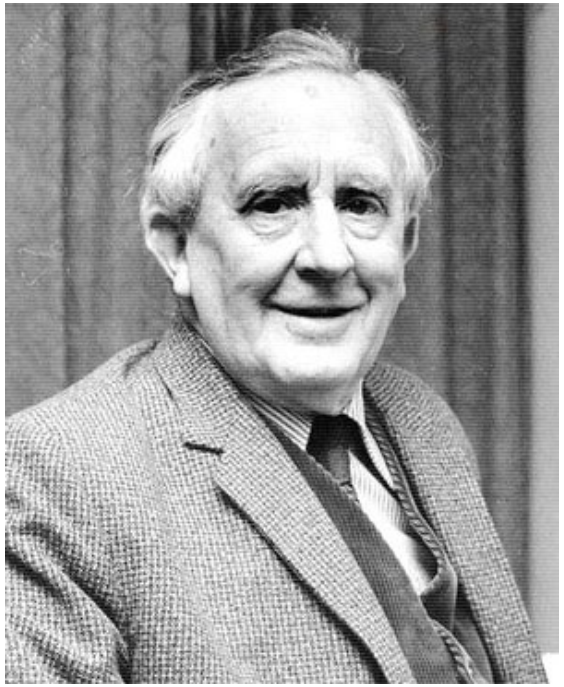
Many people use the metaphor of building a house to describe software systems development – but it's not really like that

- We are all very familiar with houses and what's in them
- Houses share many similar characteristics
- Uniqueness is bounded
  - more BR
  - kitchen on the other side
  - add a deck

Some people liken SW development to other activities – these can give us some insight into the process

- Writing a novel
- Growing a garden
- Composing a symphony

Like writing a novel, SW development begins with a blank piece of paper, and usually ends differently than the author expected



- JRR Tolkien started *The Lord of the Rings* in 1937 and didn't finish it until 1949
  - Much of it was written a section at a time and sent by mail to his son who was serving in the Royal Air Force
  - He had no clear ideas of the most of the plot and the conclusions when he started; the "tale grew in the telling"

- "Writing is an adventure." Winston Churchill
- "However great a man's natural talent may be, the art of writing cannot be learned all at once." Jean Jacques Rousseau
- "I was working on the proof of one of my poems all the morning, and took out a comma. In the afternoon I put it back again." Oscar Wilde
- "A writer is working when he's staring out of the window." Burton Rascoe
- "Nothing you write, if you hope to be any good, will ever come out as you first hoped." Lillian Helman
- "You must write every single day of your life…You must lurk in libraries and climb the stacks like ladders to sniff books like perfumes and wear books like hats upon your crazy heads….may you be in love every day for the next 20,000 days. And out of that love, remake a world." Ray Bradbury
- "Writing is an exploration. You start from nothing and learn as you go." E.L. Doctorow
- "The idea is to get the pencil moving quickly…Once you've got some words looking back at you, you can take two or three - throw them away and look for others." Bernard Malamud
- "You only learn to be a better writer by actually writing." Doris Lessing
- "Inspiration is wonderful when it happens, but the writer must develop an approach for the rest of the time…The wait is simply too long." Leonard S. Bernstein

Today's software systems are among the most complex things humans have ever done

- Windows 10: around 50 million lines of code
  - At least 4,000 full-time developers at Microsoft
- The complete suite of Google apps and tools: Around 2 billion lines of code
- Healthcare.gov
  - Expected to cost $93M; actually cost $1.5B and didn't work!
  - From Wikipedia:
    "By some estimates, only 1% of people managed to successfully enroll with the site in its first week of operation. On October 20, 2013, President Barack Obama remarked, 'There's no sugar coating: the website has been too slow, people have been getting stuck during the application process and I think it's fair to say that nobody's more frustrated by that than I am.' "

Tasks that are part of software development can be either Simple, Complicated or Complex endeavors

- Simple tasks are easy to describe and understand – just do it
- Complicated tasks have many steps and can be hard to understand
- Complex tasks are complicated but also changeable and risky

Tasks that are part of software development can be either Simple, Complicated or Complex endeavors

- Simple tasks are easy to describe and understand – just do it
- Simple doesn’t mean easy or effortless
- Shoveling snow
- Complicated tasks have many steps and can be hard to understand

- Complex tasks are complicated but also changeable and risky

Tasks that are part of software development can be either Simple, Complicated or Complex endeavors

- Simple tasks are easy to describe and understand – just do it
- Simple doesn't mean easy or effortless
- Shoveling snow
- Complicated tasks have many steps and can be hard to understand

- Assembling an Ikea desk
- Lots of pieces, takes a while
- Complex tasks are complicated but also changeable and risky

Tasks that are part of software development can be either Simple, Complicated or Complex endeavors

- Simple tasks are easy to describe and understand – just do it
- Simple doesn't mean easy or effortless
- Shoveling snow

- Complicated tasks have many steps and can be hard to understand
- Assembling an Ikea desk
- Lots of pieces, takes a while

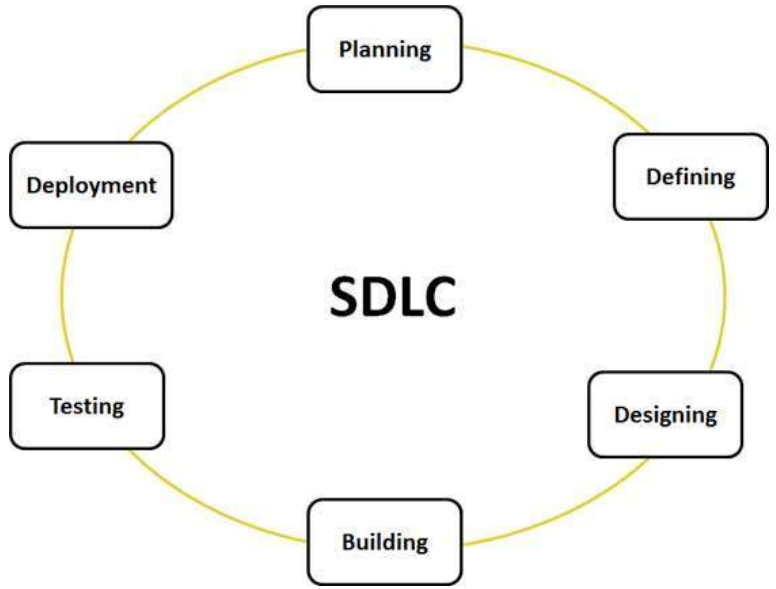
- Complex tasks are complicated but also changeable and risky
- Long, complicated, uncertain
- Heart surgery

Any decent-sized software project will have pieces that are each of these – simple, complicated and complex

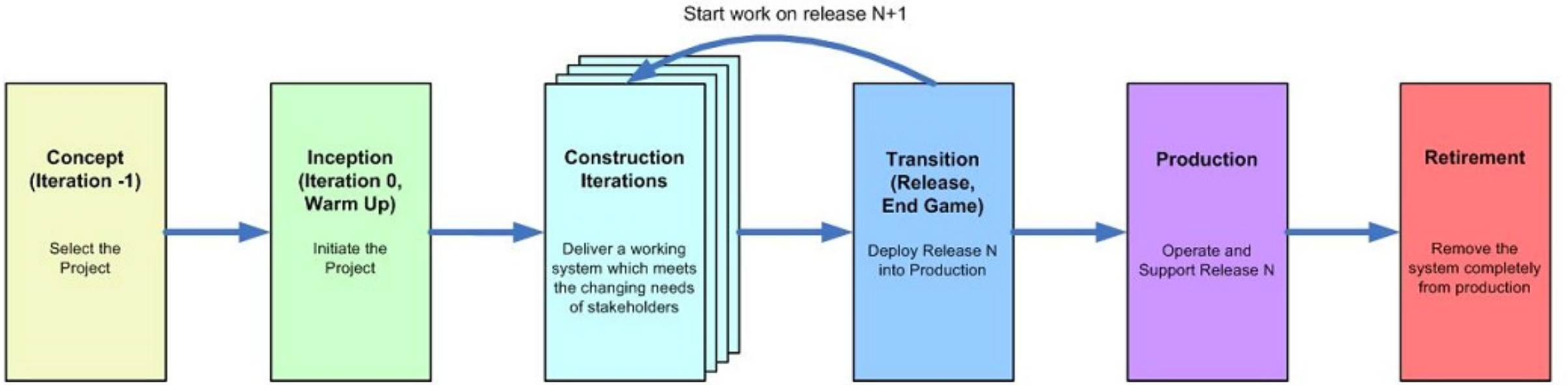
- *Implementing* a user interface is often **simple** – lots of coding that follows the same pattern
  - probably low risk, as well
- Dynamic web pages are often **complicated** – there can be many possible http requests to service with extensive code
  - but the risk is low, if we are careful
- Analysis of medical data to determine dosage is **complex:** complicated, but also risky
  - the "open data set" problem, and mistakes aren't allowed

The Systems Development Life Cycle (SDLC) is the process for all of the phases of delivering a software product

- An SDLC has as its inputs the requirements for the system desired
- Outputs are the completed SW, ongoing support, training and documentation, etc.
- Typically shown as a drawing
- ISO/IEC 12207

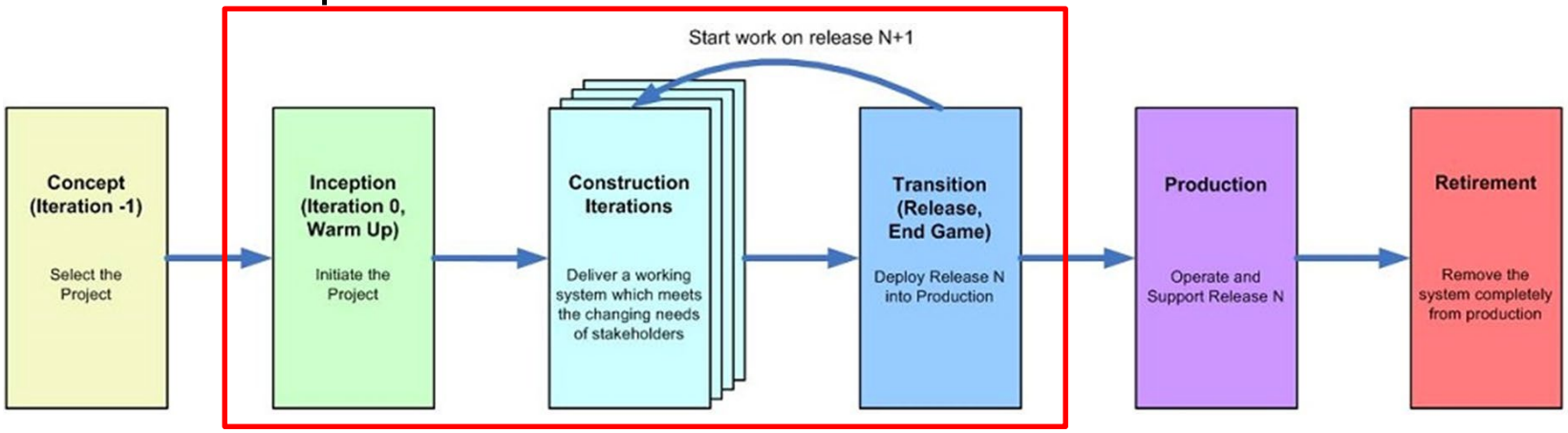


Though it's often drawn as a circle, the SDLC actually has start and endpoints in the real world



Software Development Methodology

- Our *Software Development Methodology* is the method that we use to perform the internal iterative portion of the SDLC



- One conceptualization of SDM
  - The phases or processes employed to produce software
  - The tools used to operate the process
  - The organization of people and division of labor to produce software
  - The coordination of tasks to produce software

Let's discuss Software Development Methodology – not how to write a program but how to work with others to create software

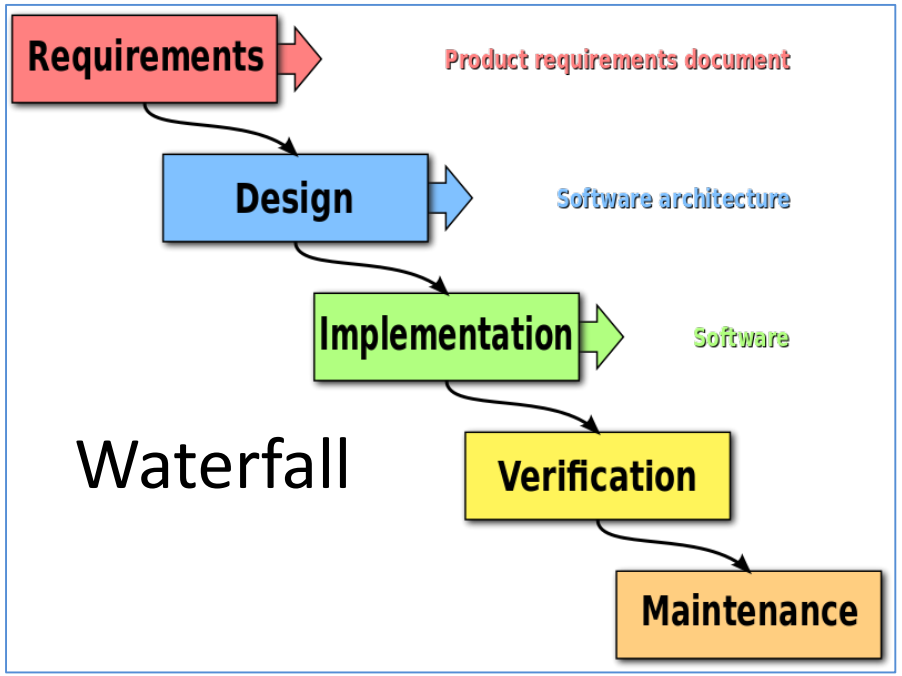
- Think about a team of programmers working on a single program
  - They can all be excellent and hard-working, but if they don't work on the right thing, or if they misunderstand what everyone's doing, the result will suffer

For a software project, the methodology is the process we use to:

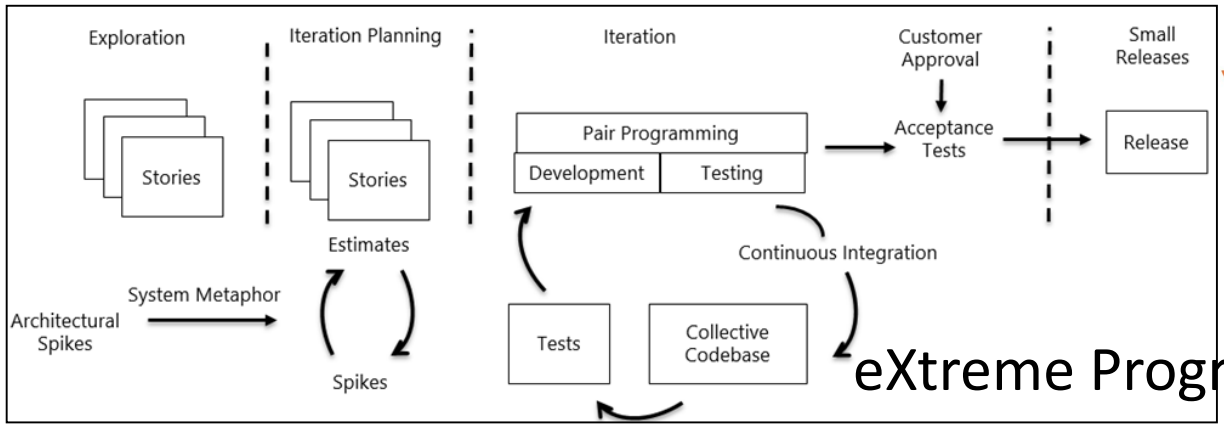
- Understand the requirements for the software
- Split up the work into phases
- Divide the tasks among individuals or teams
- Communicate with each other
- Check on our progress
- Handle changes

Question

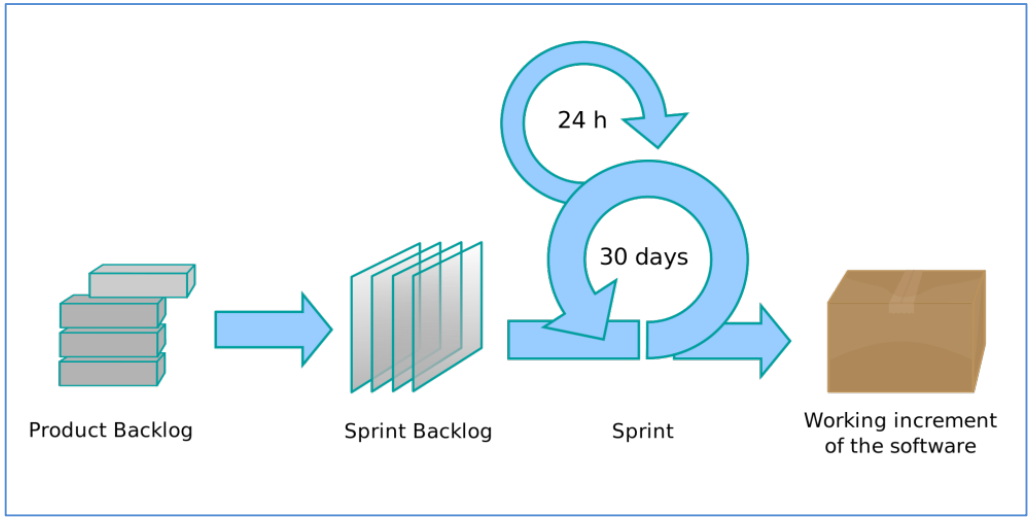
- If you have worked on a team project before (whether a software project or not), how did you do it?
- How did you assign tasks?
- How did you keep track of what was being accomplished?
- How did you deal with unanticipated changes or problems?
- How well did your methods work?



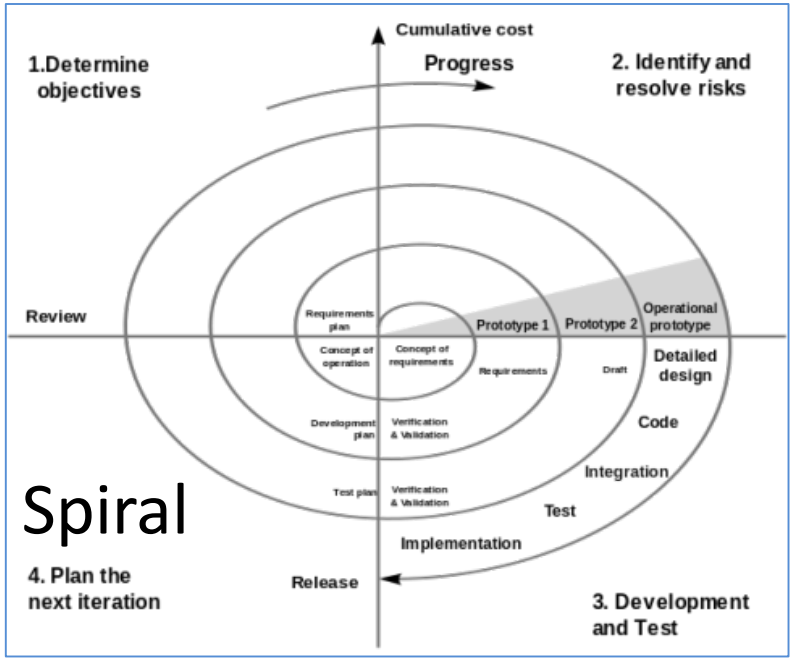
Waterfall



eXtreme Programming

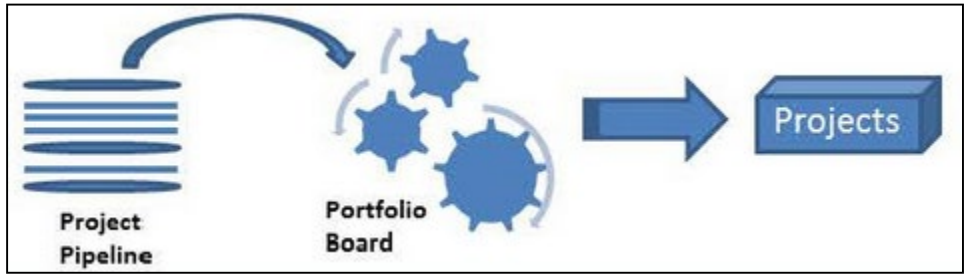


Scrum



Spiral

Kanban

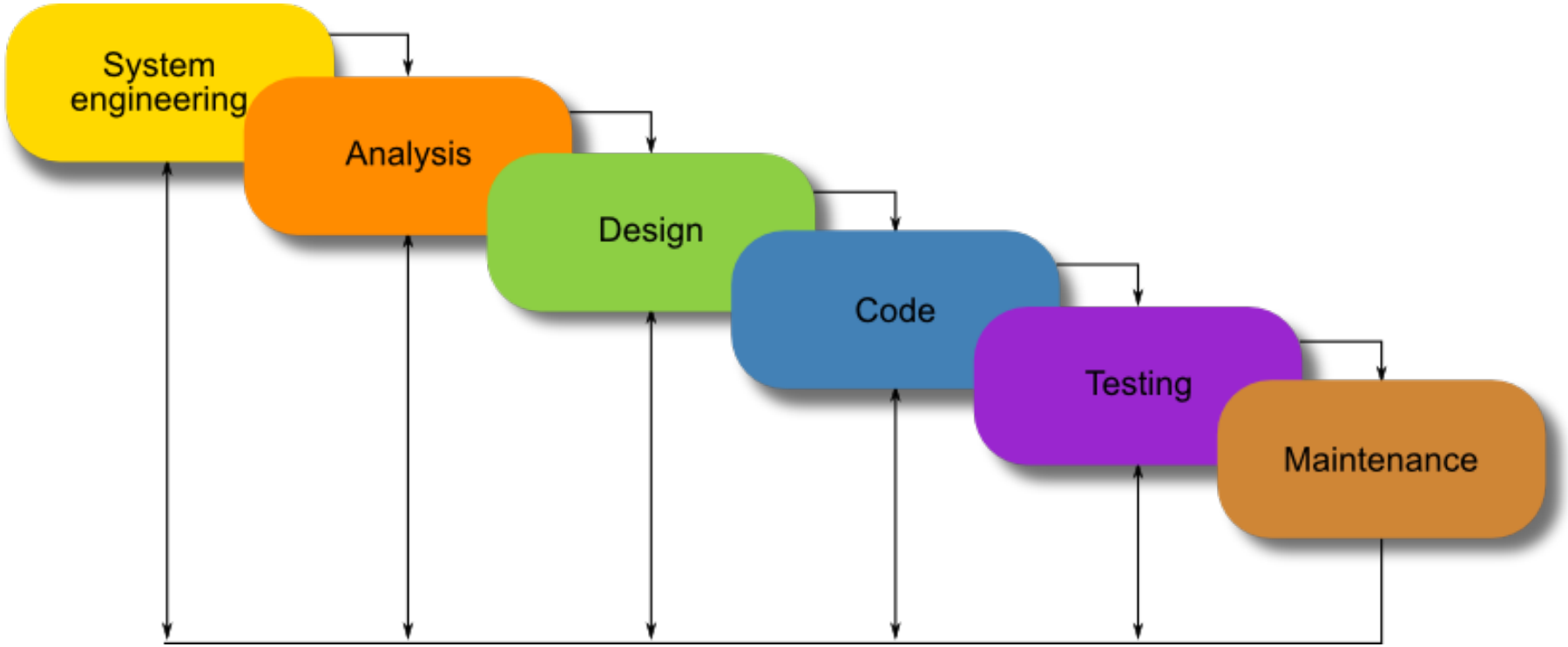


The Waterfall Development Methodology is non-iterative, structured, documentation-based and designed for determinism

1. System and software requirements: captured in a product requirements document
2. Analysis: resulting in models, schema, and business rules
3. Design: resulting in the software architecture
4. Coding: the development, proving, and integration of software
5. Testing: the systematic discovery and debugging of defects
6. Operations: the installation, migration, support, and maintenance of complete systems

Only move to the next phase when the current is complete

Waterfall, graphically



Advantages and Disadvantages of the Waterfall development methodology

- ☺ Designed for control of the phases
- ☺ Division of work is inherent
- ☺ Project divisions work well with a matrix structured organization
- ☺ Milestones are well-understood
- ☺ SW requirements are known in detail before coding begins

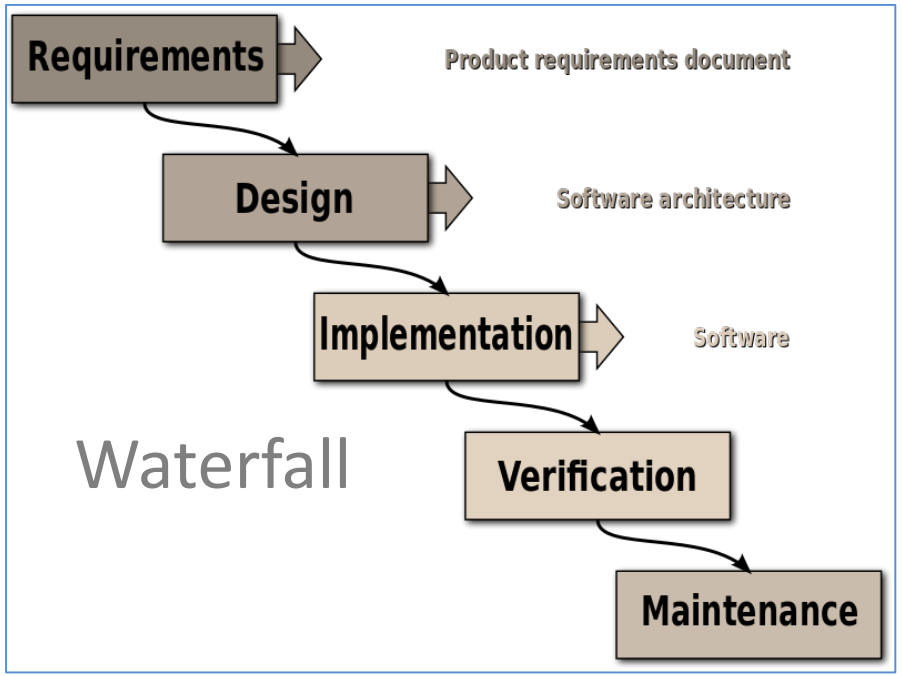
Advantages and Disadvantages of the Waterfall development methodology

- ☺ Designed for control of the phases
- ☺ Division of work is inherent
- ☺ Project divisions work well with a matrix structured organization
- ☺ Milestones are well-understood
- ☺ SW requirements are known in detail before coding begins

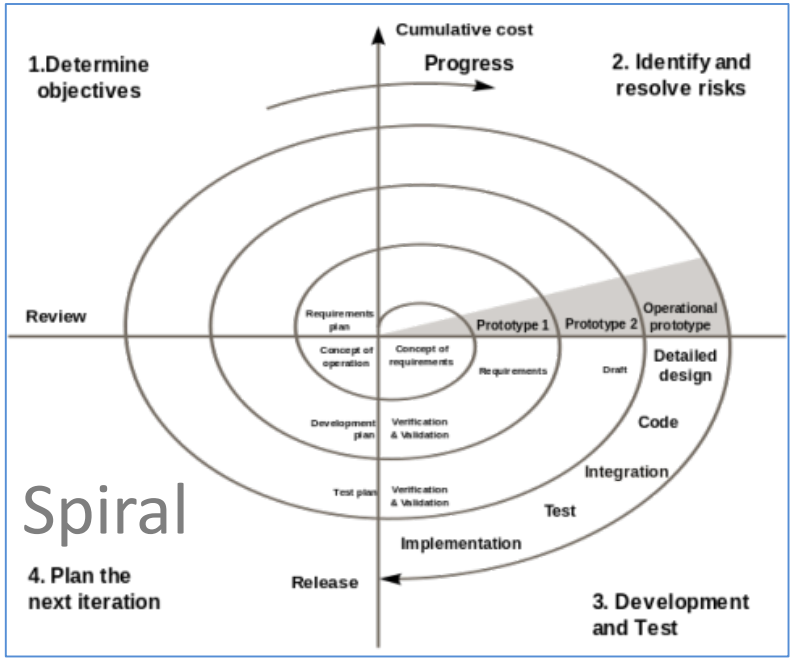
- ☹ Cannot accommodate changes in requirements!
- ☹ Overall project integration is at the end
- ☹ No working code is produced early in the project; nothing to show customers
- ☹ Areas of technical risk are not investigated until late in the work

So, when (if ever) is waterfall still used? When project, client or organizational conditions make it the best or only choice

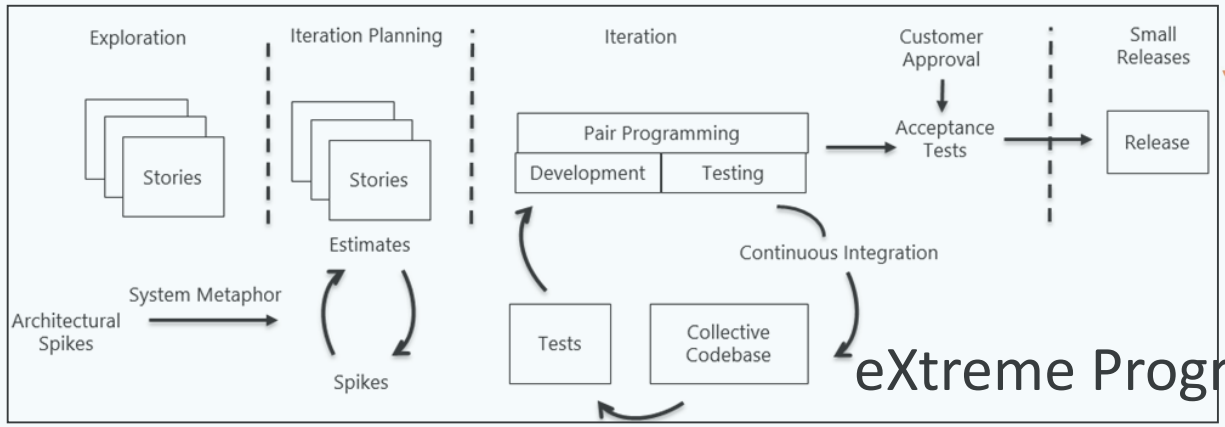
- If a project is:
  - well-understood, short, low-risk, with requirements that are clear and unchanging
- If the customer demands
  - Many federal, state and local government customers require (or at least prefer) waterfall
- If the organization typically works this way and it's successful



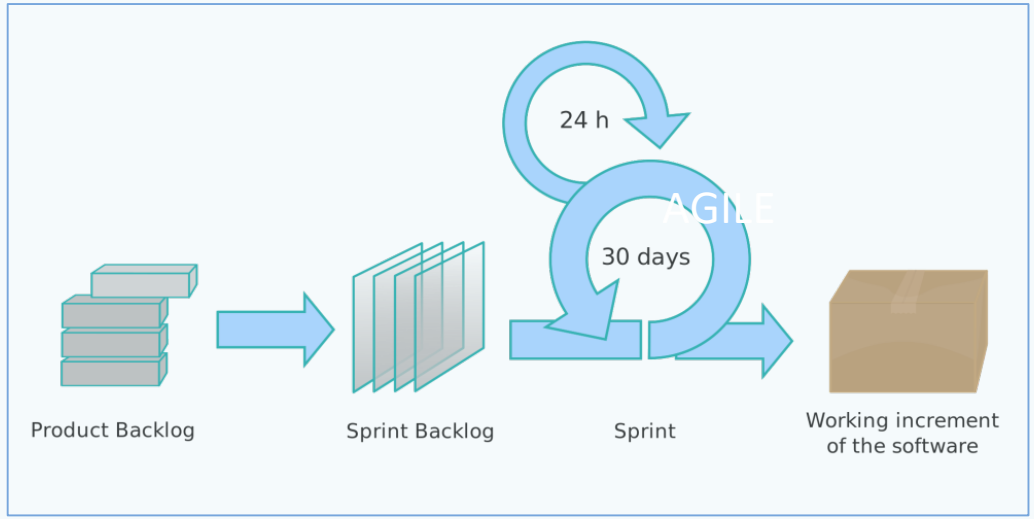
Waterfall



Spiral



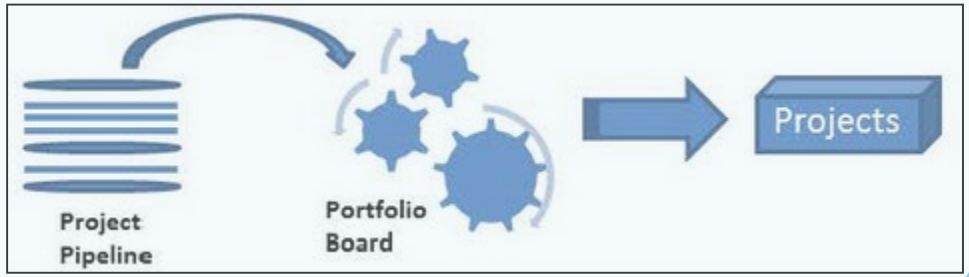
eXtreme Programming



Scrum

AGILE

Kanban



"Agile" isn't exactly a methodology – it's actually an approach to SW development; frequently-used agile methods include Scrum, Kanban and XP

- Scrum
  - *Product backlog* drives development
  - A chunk is pulled off and implemented in a *sprint*
  - Finish up deliverable version, and repeat until done
- Extreme Programming (XP)
  - Customer prioritizes requirements, which are handled in *iterations*
  - Changes during iterations are possible
  - Use of TDD, pair programming, automated testing, refactoring, etc.
- Kanban
  - Series of steps: Pending, Analysis, Development, Test, Deploy
  - Work items flow through the steps, adding to the product
  - Bottlenecks are identified and fixed

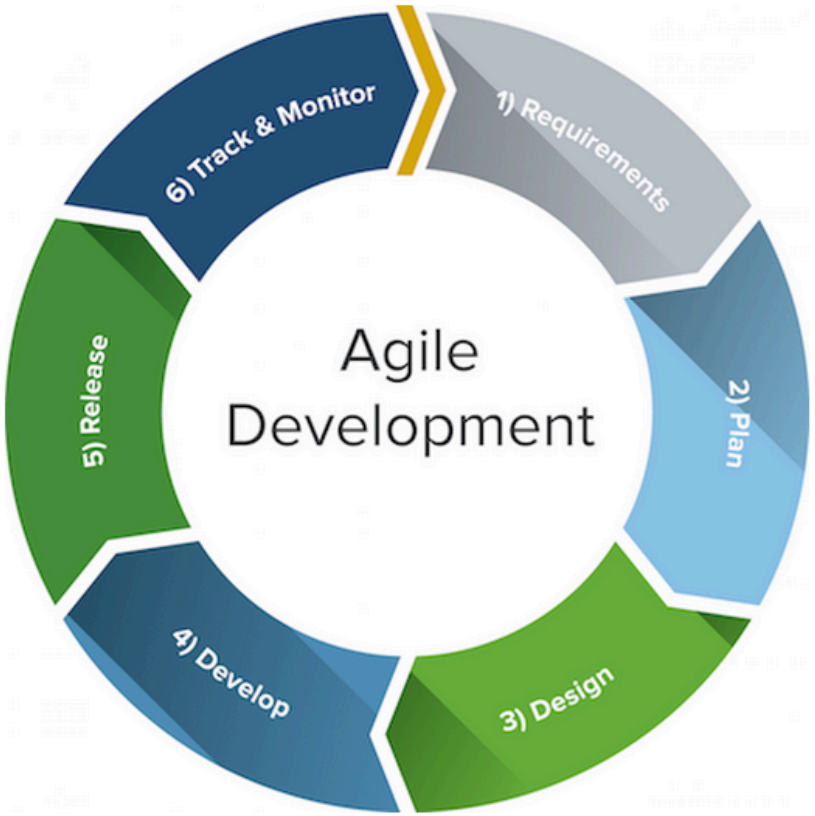
A *product backlog* is a list of requirements that must be met to produce the desired product (sometimes called the *project backlog*)

PRODUCT BACKLOG EXAMPLE						
ID	**As a…**	**I want to be able to…**	**So that…**	**Priority**	**Sprint**	**Status**
1	Administrator	see a list of all members and visitors	I can monitor site visits	Must	1	Done
2	Administrator	add new categories	I can allow members to create engaging content	Must	1	Done
3	Administrator	add new security groups	security levels are appropriate	Must	1	Done
4	Administrator	add new keywords	content is easy to group and search for	Must	1	Done
5	Administrator	delete comments	offensive content is removed	Must	1	Done
6	Administrator	block entries	competitors and offenders cannot submit content	Must	1	Done
7	Administrator	change site branding	the site is future-proofed in case brand changes	Could	1	Done
8	Member	change my password	I can keep secure	Must	1	Done
9	Member	update my contact details	I can be contacted by Administrators	Must	2	Work in Progress
10	Member	update my email preferences	I'm not bombarded with junk email	Should	2	Work in Progress
11	Member	share content to social networks	I can promote what I find interesting	Could	2	Work in Progress
12	Visitor	create an account	I can benefit from member discounts	Must		To be started
13	Visitor	login	I can post new entries	Must		To be started
14	Visitor	add comments	I can have a say	Must		To be started
15	Visitor	suggest improvements	I can contribute to the site usability	Should		To be started
16	Visitor	contact the Administrators	I can directly submit a query	Could		To be started
17	Visitor	follow a member's updates	I'm informed of updates from members I find interesting	Should		To be started
18	Visitor	view a member's profile	I can know more about a member	Must		To be started
19	Administrator	generate incoming traffic report	I can understand where traffic is coming from	Must		To be started

this info will be filled in as sprints progress

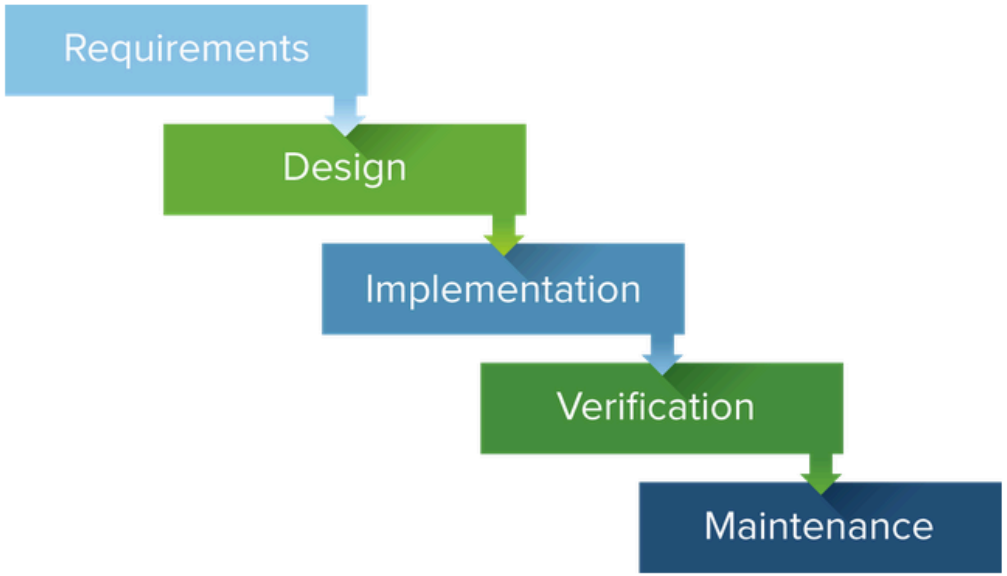
techno-pm.com

Agile



- Continuous cycles
- Small, high-functioning, collaborative teams
- Multiple methodologies
- Flexible/continuous evolution
- Customer involvement

Waterfall



- Sequential/linear stages
- Upfront planning and in-depth documentation
- Contract negotiation
- Best for simple, unchanging projects
- Close project manager involvement

Agile Origins

- Agile methods began to take shape in the 1990s
- Popularly defined and recognized in the early 2000s due to the Agile Manifesto
- The manifesto consists of four value statements and twelve principles
- It is not a specific methodology like Waterfall or Spiral, but a philosophy or perspective

http://agilemanifesto.org/


agilemanifesto.org
Apps LM A Bb BBC CBU CSDS eO f G G GG i K! Other Bookmarks
Manifesto for Agile Software Development
We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan
That is, while there is value in the items on
the right, we value the items on the left more.

The Values of Agile

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

The Agile Principles 1-6

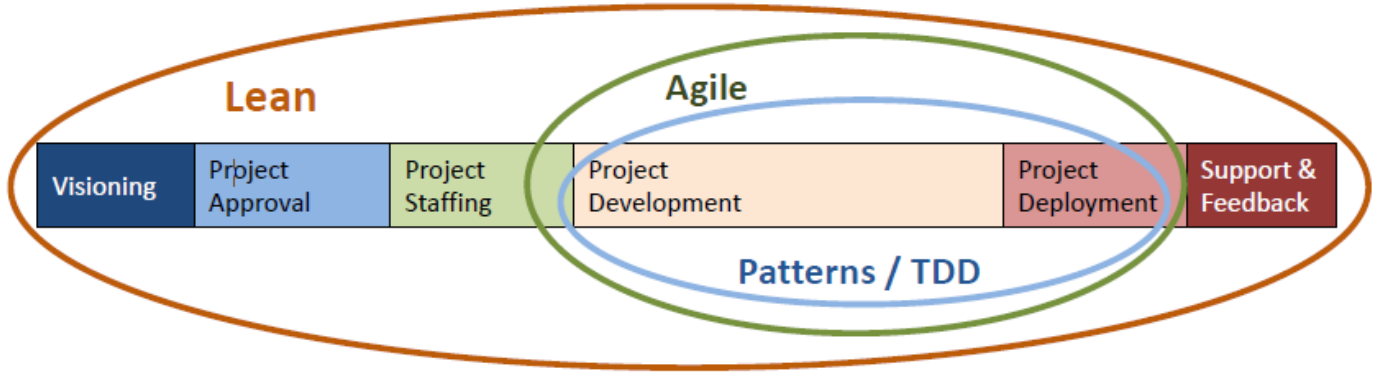
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

The Agile Principles 7-12

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Another word used in connection with agile methods is "lean" – Adopted from lean manufacturing processes

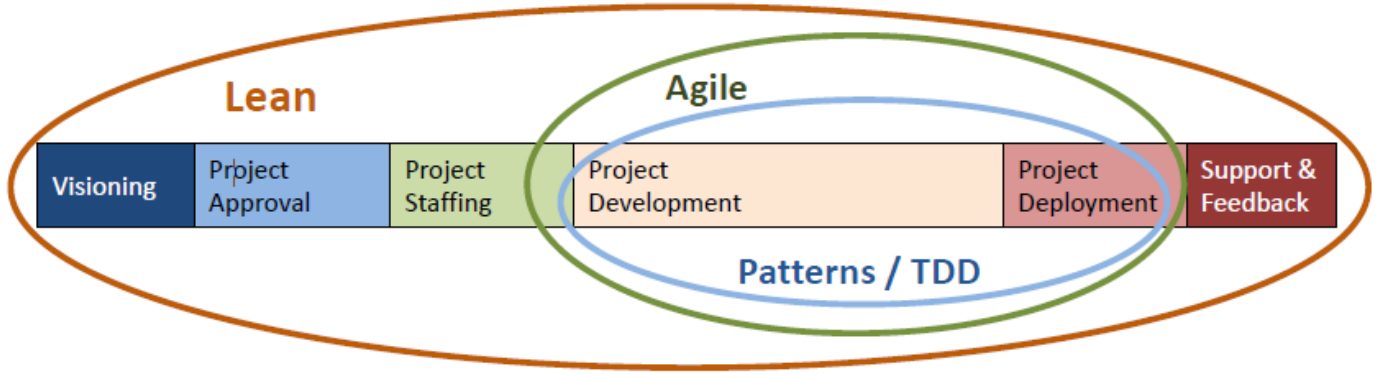
1. Eliminate waste
2. Increase learning
3. Decide as late as possible
4. Deliver as fast as possible
5. Empower the team
6. Build in quality
7. Optimize the system



https://www.infoq.com/news/2008/11/Lean-Agile-Alan-Shalloway/

Another word used in connection with agile methods is "lean" – Adopted from lean manufacturing processes

1. Eliminate waste
   - prevent duplication of effort (design, coding, testing)
2. Increase learning
   - share insights with the team
3. Decide as late as possible
   - push detail to the lowest level of code
4. Deliver as fast as possible
   - early delivery of working code
5. Empower the team
   - distribute responsibility
6. Build in quality
   - good design, coding, unit test
7. Optimize the system
   - focus on integrating everyone's pieces together

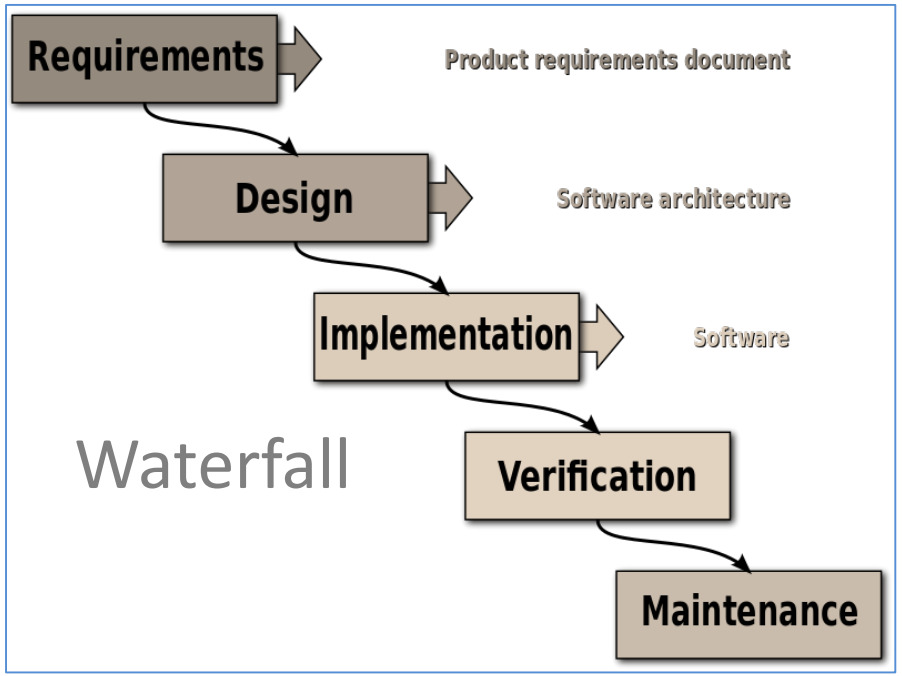


https://www.infoq.com/news/2008/11/Lean-Agile-Alan-Shalloway/

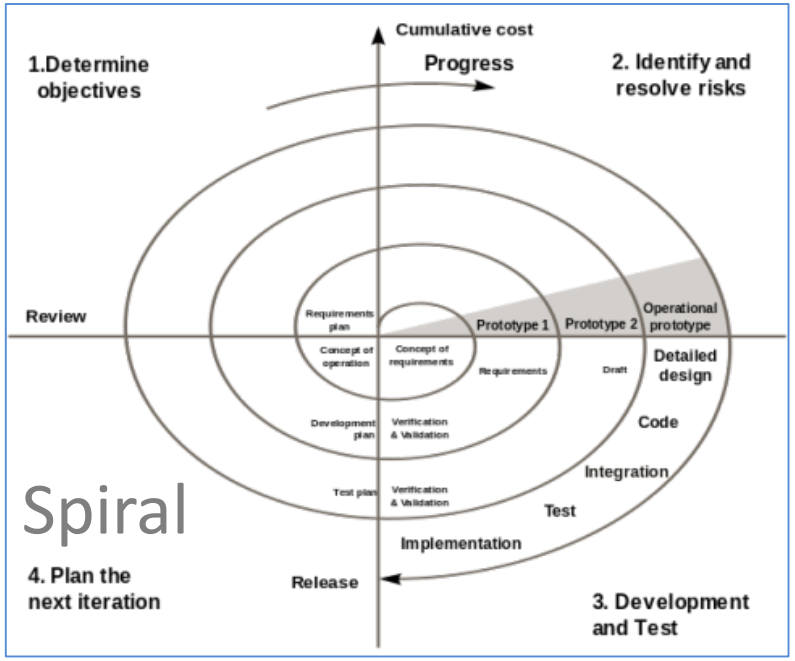
A really interesting example of an agile process is a Subway sandwich shop

1: Our highest priority is to satisfy the customer…
2: Welcome changing requirements, even late in development.
3: Deliver working software frequently…
4: Business people and developers must work together…
5: Build projects around motivated individuals...
6: …face-to-face conversation.
7: Working software is the primary measure of progress.
8: Agile processes promote sustainable development.
9: Continuous attention to technical excellence…
10: Simplicity … is essential.
11: …self-organizing teams.
12: …the team reflects on how to become more effective, then tunes...

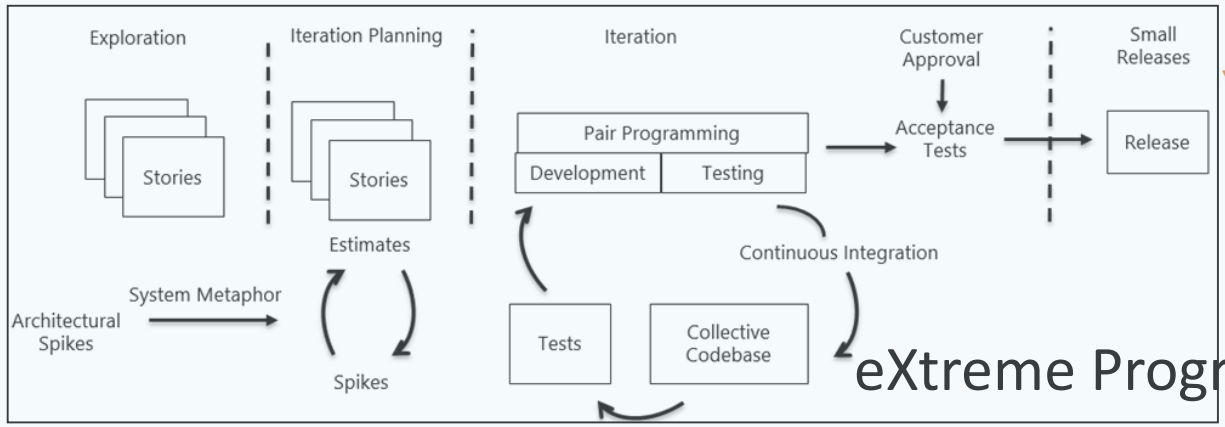
https://txm.com/benchmarking-lean-principles-subway/



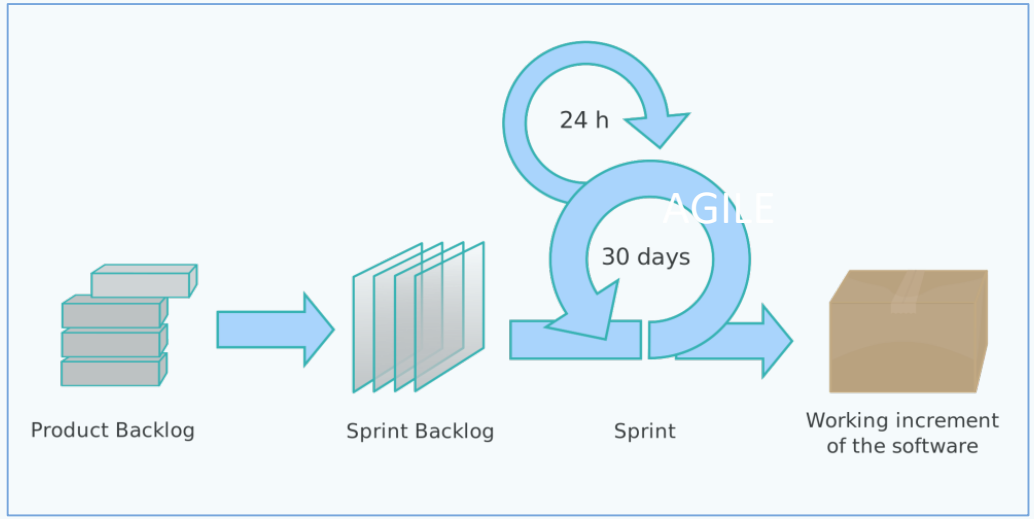
Waterfall



Spiral



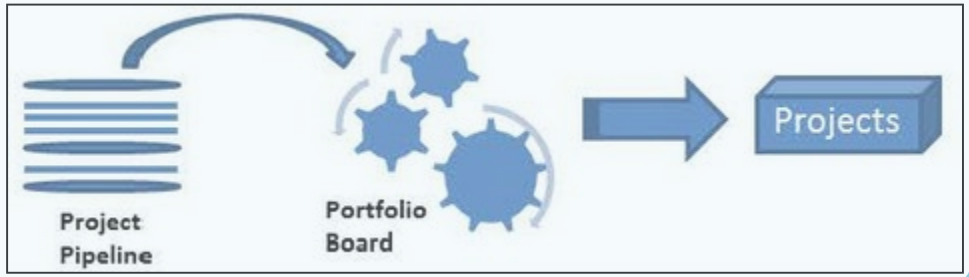
eXtreme Programming



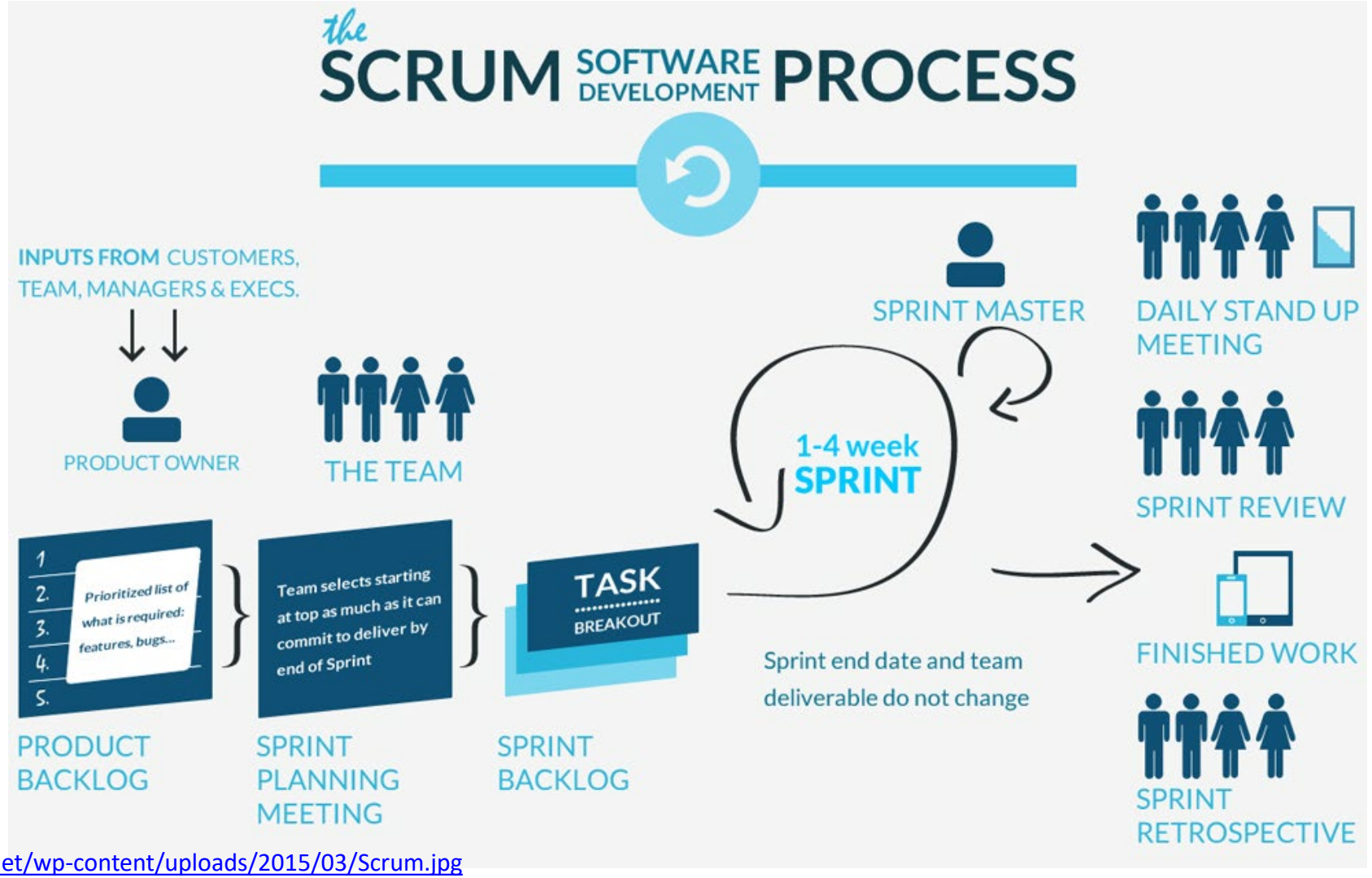
Scrum

AGILE

Kanban

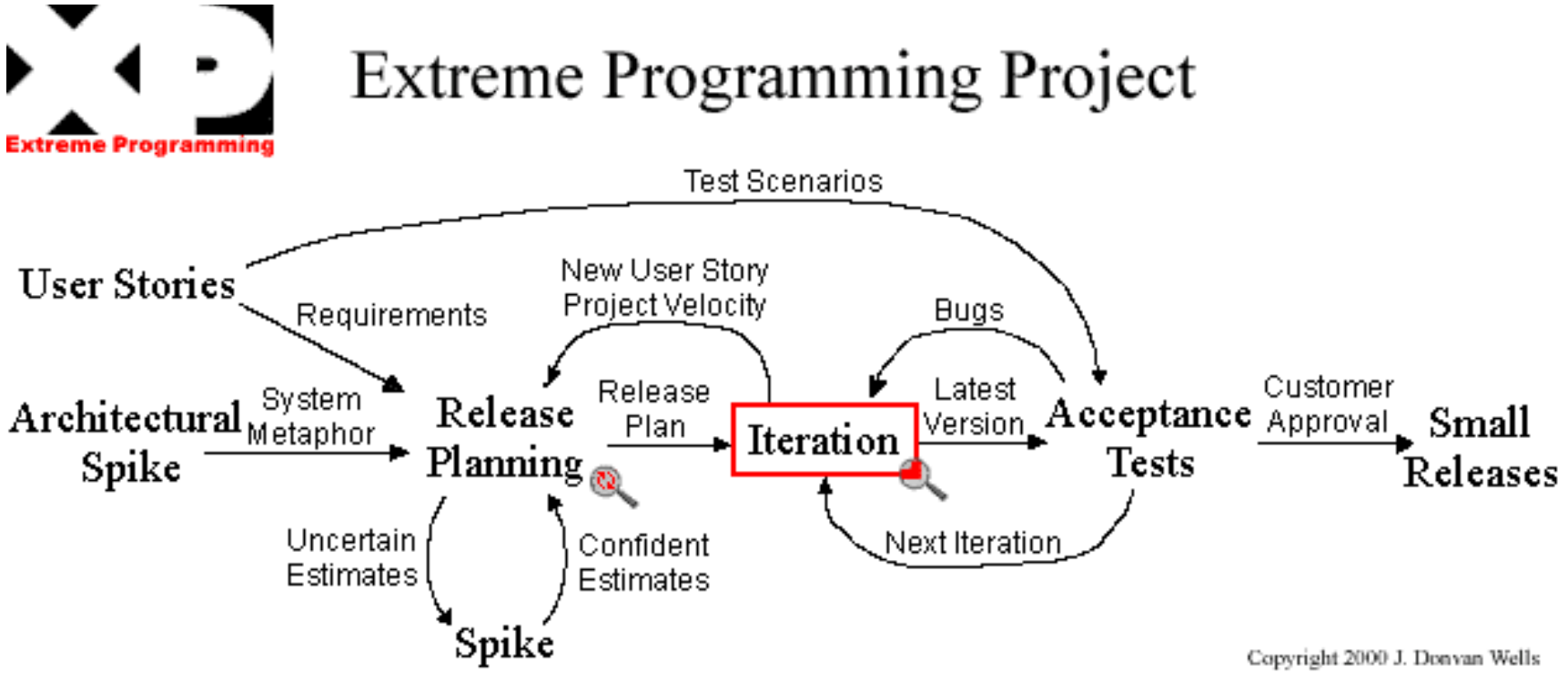


Scrum is implemented by the development team, led by the scrum master, directed by the product owner



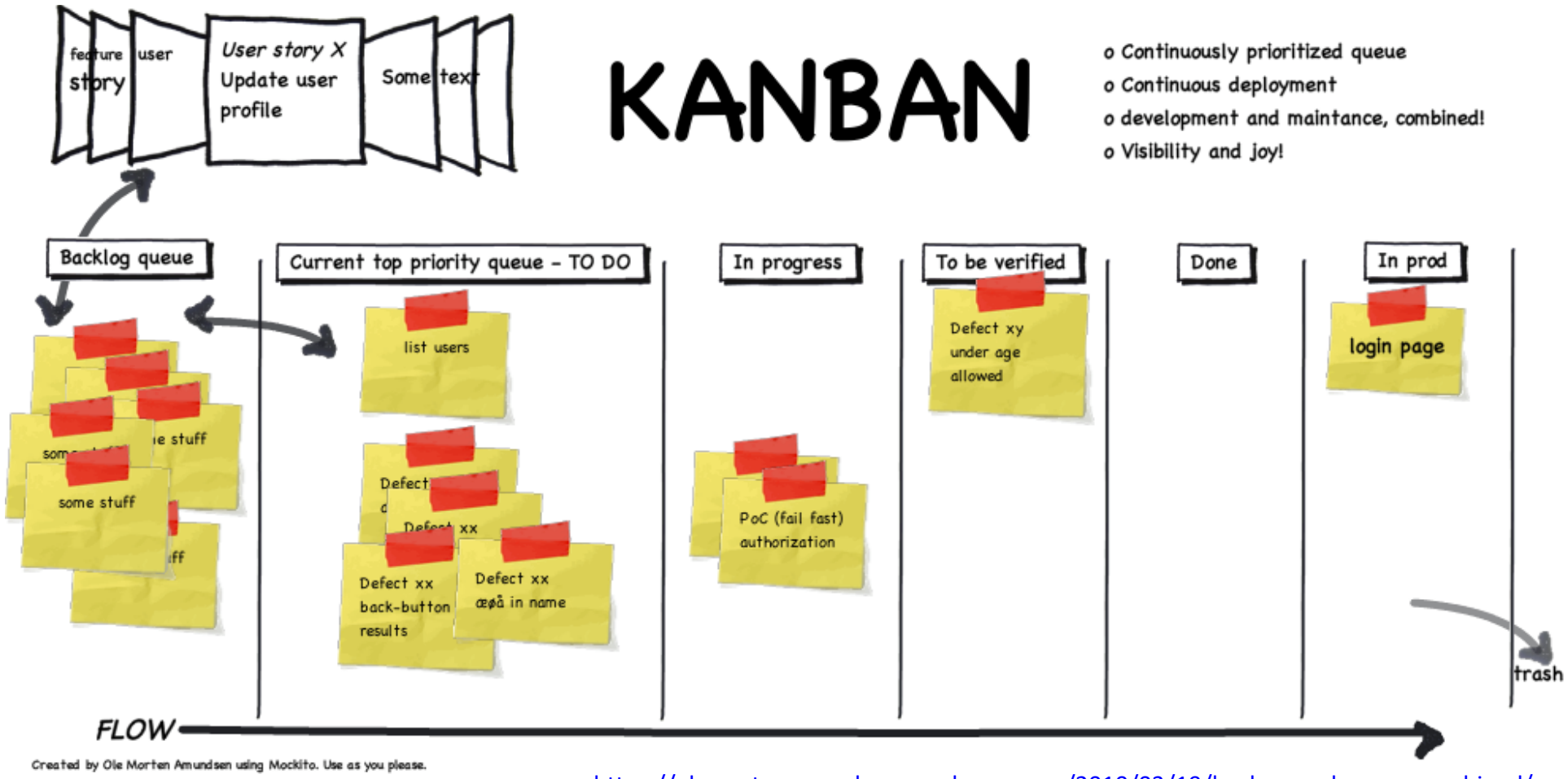
http://repone.net/wp-content/uploads/2015/03/Scrum.jpg

XP is also iterative but is built on the use of key programming practices such as pair programming, unit test, test-driven dev., etc.



http://www.extremeprogramming.org/map/project.html

Kanban emphasizes continuous checking of priorities, continuous deployment and continuous improvement of the process



https://olemortenamundsen.wordpress.com/2010/03/19/kanban-and-scrum-combined/

Advantages and disadvantages of Agile methodologies

- ☺ Can be very responsive to change
- ☺ Early review of actual code with client
- ☺ Not as much up-front documentation work
- ☺ Deals well with uncertainty in the requirements or in feasibility of approaches
- ☺ Risk reduction through early and continuous releases
- ☺ Accommodates the social aspects of software development

Advantages and disadvantages of Agile methodologies

- ☺ Can be very responsive to change
- ☺ Early review of actual code with client
- ☺ Not as much up-front documentation work
- ☺ Deals well with uncertainty in the requirements or in feasibility of approaches
- ☺ Risk reduction through early and continuous releases
- ☺ Accommodates the social aspects of software development

- ☹ Schedule is less predictable
- ☹ New methodology, must be learned and practiced well
- ☹ Management is often uncomfortable with it
- ☹ May require more skills of more team members
- ☹ Prioritizing requirements can be difficult
- ☹ Difficult to incorporate non-functional requirements
- ☹ High commitment and dependence on customer representative

Advantages and disadvantages of Agile methodologies

- ☺ **Can be very responsive to change**
- ☺ Early review of actual code with client
- ☺ Not as much up-front documentation work
- ☺ **Deals well with uncertainty in the requirements or in feasibility of approaches**
- ☺ **Risk reduction through early and continuous releases**
- ☺ **Accommodates the social aspects of software development**

- ☹ **Schedule is less predictable**
- ☹ New methodology, must be learned and practiced well
- ☹ Management is often uncomfortable with it
- ☹ May require more skills of more team members
- ☹ Prioritizing requirements can be difficult
- ☹ Difficult to coordinate across large projects
- ☹ Difficult to incorporate non-functional requirements
- ☹ **High commitment and dependence on customer representative**

Topics for Today

- Introduction to the Course
  - Syllabus
  - Course Schedule
- Software for Engineering Systems
- What is a Software Development Methodology
  - Waterfall Methodology
  - Agile Methodologies