Software Engineering

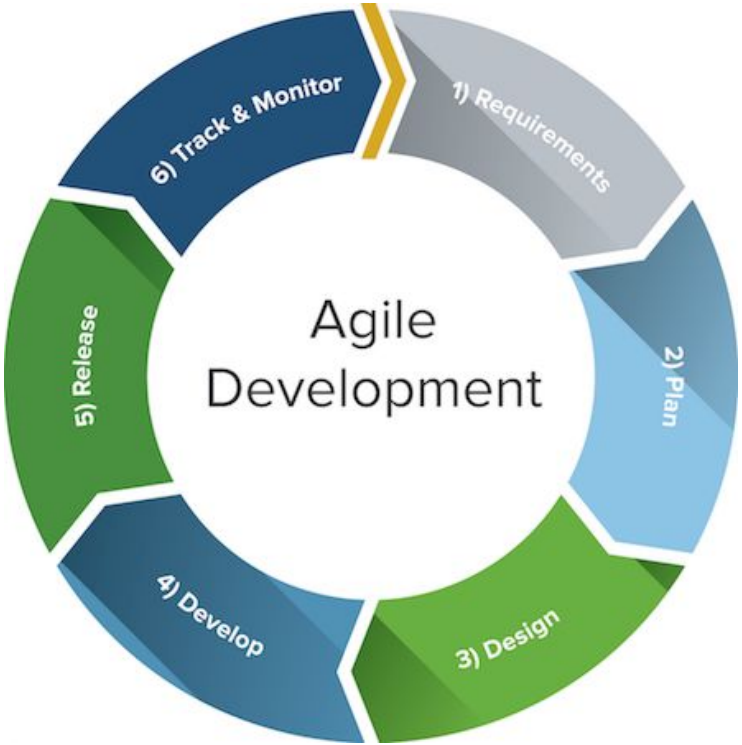
Mo.Agamia

Plan

- Scrum
  - Process
  - Tools, Artifacts, and Methods
- UML
  - Structural UML diagrams
  - Behavioral UML diagrams
- SRS

1 Scrum

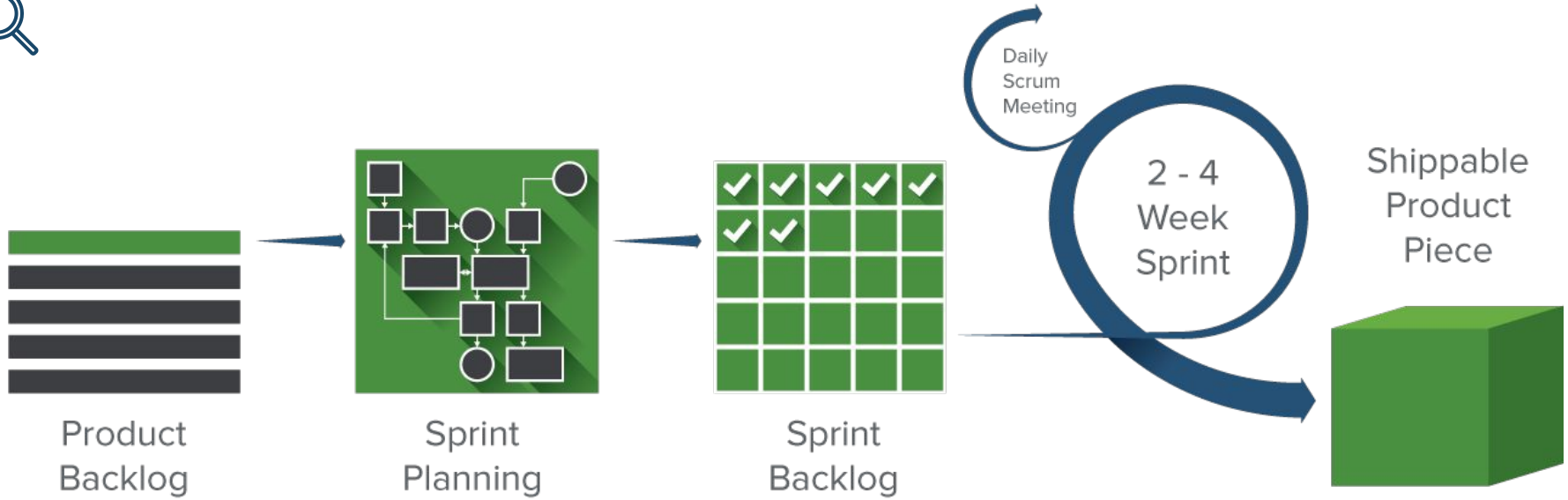
Agile Development Cycle



Mo.Agamia

1.1 Process

Scrum Process Steps



Scrum Process Steps

Product Charter

Is a high-level summary of the project's key success factors. ITt includes the major objectives of the project, scope boundaries, and reciprocal agreements.

Sprint planning

Is a list of the top items on the backlog to complete in the sprint.

Backlog refinement/grooming

Is to ensure the backlog only contains items that are relevant and detailed, and that meet the project's objectives.

Product backlog

Is a list of all the desired features for the product.

Sprint review meeting

At the end of each sprint, the team presents the work they have completed at a sprint review meeting.

Sprint retrospective meeting

At the end of each sprint, the team reflects on how well Scrum is working for them and talks about any changes that need to be made in the next sprint.

Project Charter Template

Agile Project Charter

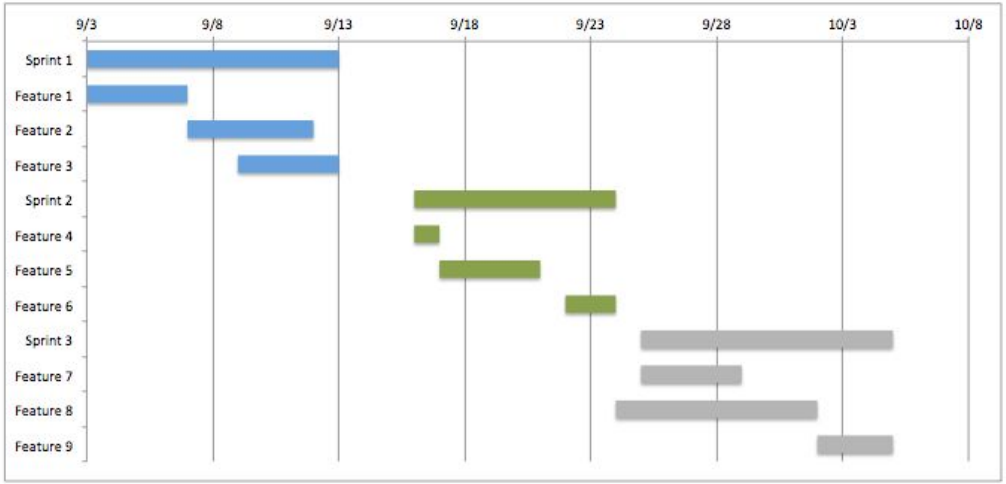
General Project Information	
Project Name	
Project Champion	
Project Sponsor	
Project Manager	
Stakeholders	
Expected Start Date	
Expected Completion Date	

Project Details
Mission
Vision
Scope
Success Metrics

Date:	Date:

Project Plan Template

Task Name	Responsible	Start	End	Days	Status
Sprint 1	Alex B.	9/3	9/13	10	Complete
Feature 1	Frank C.	9/3	9/7	4	Complete
Feature 2	Jacob S.	9/7	9/12	5	Complete
Feature 3	Jacob S.	9/9	9/13	4	Overdue
Sprint 2	Jacob S.	9/16	9/24	8	In progress
Feature 4	Alex B.	9/16	9/17	1	In progress
Feature 5	Frank C.	9/17	9/21	4	Not started
Feature 6	Shari W.	9/22	9/24	2	Not started
Sprint 3	Shari W.	9/25	10/5	10	Not started
Feature 7	Alex B.	9/25	9/29	4	Not started
Feature 8	Kennedy K.	9/24	10/2	8	Not started
Feature 9	Jacob S.	10/2	10/5	3	Not started



Release Plan Template

Sprint	Task	Start	End	Duration	Status	Release date	Goal
1	Task 1	2/5/2018	2/8/2018	3	Planned	2/12/18	
1	Task 2	2/12/2018	2/20/2018	8	Ongoing	2/24/18	
1	Task 3	2/19/2018	2/26/2018	7	Released	3/1/18	

Product Backlog Template

Task Name	Story	Sprint Ready	Priority	Status	Story Points	Assigned to Sprint
Sprint 1	No	No	High	In Progress	24	No
Task 1	Yes	Yes	Medium	Complete	8	Yes
Task 2	Yes	Yes	Medium	Complete	16	Yes
Task 3	Yes	Yes	Medium	Complete	0	Yes
Sprint 2	Yes	Yes	Medium	In Progress	96	Yes
Task 4	Yes	Yes	Low	Complete	32	Yes
Task 5	Yes	Yes	Low	Complete	48	Yes
Task 6	No	No	Medium	Not Started	16	No
Sprint 3	Yes	No	Medium	In Progress	32	No
Task 7	Yes	No	Low	In Progress	8	No
Task 8	No	Yes	Medium	In Progress	8	No
Task 9	Yes	No	Medium	In Progress	16	No
Sprint 4	Yes	Yes	Medium	In Progress	64	Yes
Task 10	Yes	No	Low	In Progress	32	No
Task 11	Yes	Yes	Low	Complete	32	Yes
Task 12	Yes	Yes	Medium	Complete	0	Yes
Sprint 5	No	No	Low	Not Started	64	No
Task 13	No	No	Low	Not Started	48	No
Task 14	No	No	Low	Not Started	8	No
Task 15	No	No	Low	Not Started	8	No

Sample Sprint Backlog

Sprint Backlog Template

Backlog Item	Story Points	Responsible	Status	Original Estimate	Day 1	Day 2	Day 3	Day 4	Day 5	Sprint Review
User Story #1	**8**									
Task				7	5	3	0	0	0	0
Task				3	1	1	5	0	1	0
Task				1	0.5	0	3	0	0	0
Task				0.5	1	2	3	1	0	0
User Story #2	**1**									
Task				3	3	0.5	0.5	0	0	2
Task				3	5	5	1	1	1	0
Task				2	2	5	0	1	0	1
Task				5	5	9	5	1	0	1
User Story #3	**5**									
Task				8	6	0	0	0	0	0
Task				3	1	3	3	3	0	0
Task				1.5	1	0.5	0.5	1	1	0
Task				2	0.5	0	0	0	0	3
User Story #4	**8**									
Task				9	4	2	2	1	1	0
Task				6	6	3	3	3	1	1
Task				6	2	8	8	1	0	1
Task				0.5	0.5	0.5	0.5	0	0	0
User Story #5	**3**									
Task				2	1	1	1	0.5	1	1
Task				6	6	6	0.5	3	9	0
Task				9	9	9	4	3	3	3
Task				0.5	0.5	0.5	1	0.5	0	1
Total				78	60	59	41	20	18	14

Sample Test Plan

Agile Test Plan

Project Name		Browser:	
Test Case ID		Version:	
Written By:		Description:	
Tested By:		Tested On:	

Test #	Date	Action	Expected Results	Actual Results	Pass?
1	6-Mar	Logging in	Should get to home screen	User directed to different page	☐
~~2~~	~~12-Mar~~	~~Signing up for webinar~~	~~Get confirmation email~~	~~Confirmation email received~~	☑
3	20-Mar	Click magnifying glass	Whole page gets bigger	Text changes sizes	☐
~~4~~	~~1-Apr~~	~~Click blog post hero image~~	~~Go to blog post~~	~~Go to blog post~~	☑
					☐
					☐
					☐
					☐
					☐
					☐
					☐
					☐
					☐
					☐
					☐

Tools, Artifacts, and Methods

Scrum Tools, Artifacts, and Methods

Scrum board

The Scrum board is usually divided into three categories: to do, work in progress, and done. The Scrum Team needs to update the board throughout the entire sprint.

User stories

A user story describes a software feature from the customer's perspective. It includes the type of user, what they want, and why they want it.

Burndown chart

Is a chart representing all the outstanding work. The backlog is usually on the vertical axis, with time along the horizontal axis.

Large-Scale Scrum

The principles are taken directly from Scrum, however focuses on scaling up without adding additional overhead (like adding more roles, artifacts, or processes).

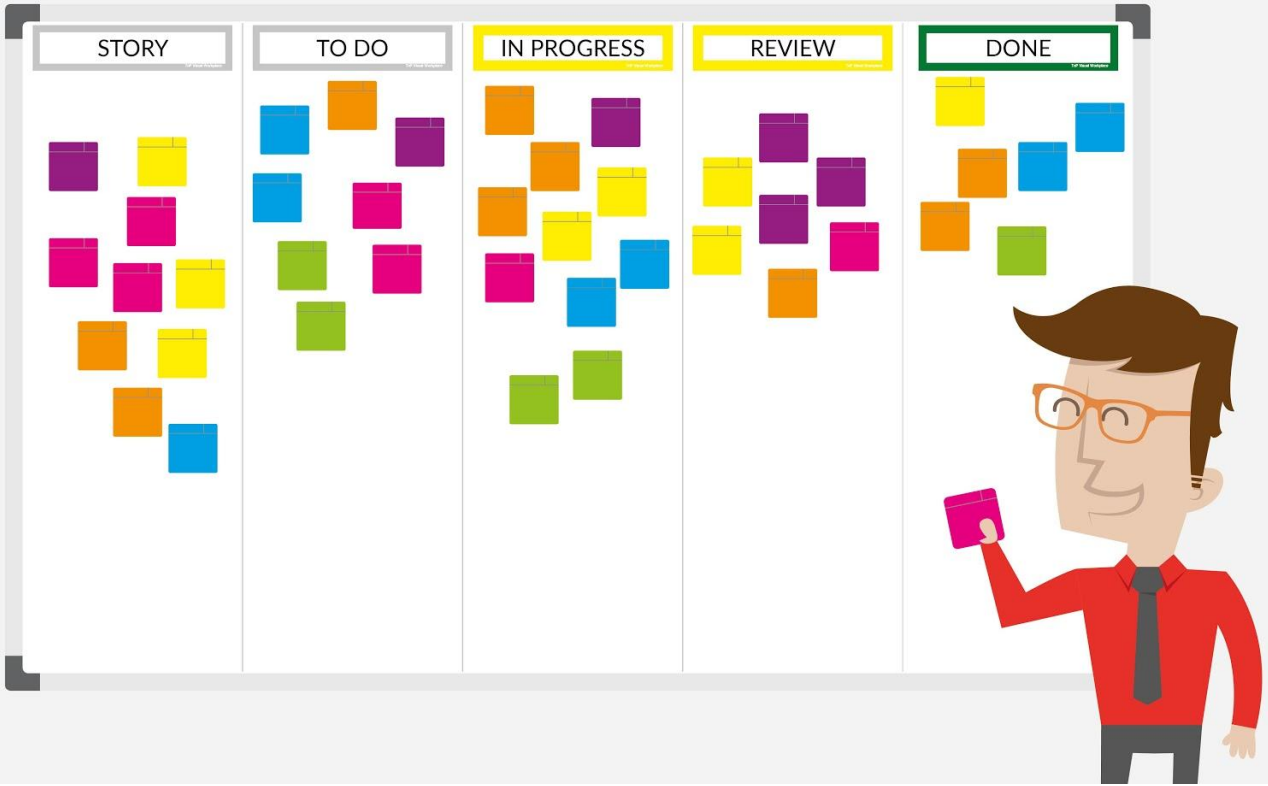
Timeboxing

A timebox is a set period of time during which a team works towards completing a goal. Instead of letting a team work until the goal is reached, the timebox approach stops work when the time limit is reached.

Icebox

Any user stories that are recorded but not moved to development are stored in the icebox.

The scrum board



Mo.Agamia

User Stories

User stories are simple, clear, brief descriptions of functionality that will be valuable to either a user or purchaser of a product. User stories should be:

- **Independent** - Dependencies between stories lead to prioritization and planning problems.
- **Negotiable** - they are not written story cards are short!
- **Valuable** - each story must bring some business value.
- **Estimable** - each story must have an estimated time & cost.
- **Small** - to estimate and track progress within a sprint.
- **Testable** -must be a criteria of “done”

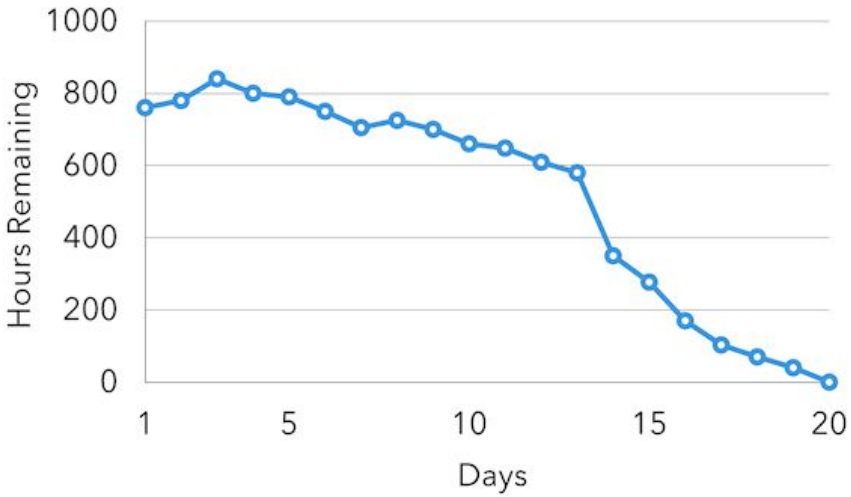
User Stories

User Story ID	As a <type of user>	I want to <perform some task>	So that i can <achieve some goal>
1	Project manager	View a status report from each team member	Ensure the project stays on track
2	Employee	Be reminded of upcoming deadlines	Complete my tasks on time
3	Director	See the big picture view of department work	Stay

User Stories

User Story	Tasks	Day 1	Day 2	Day 3	Day n
As a member, I can read profiles of other members so that i can find someone to date.	Code the...	8	4	8	...
	Design the UI	16	12	10	...
	Automate tests ...	4	4	1	...
	Meet with Mary about...	4	16	12	...

Burndown Chart



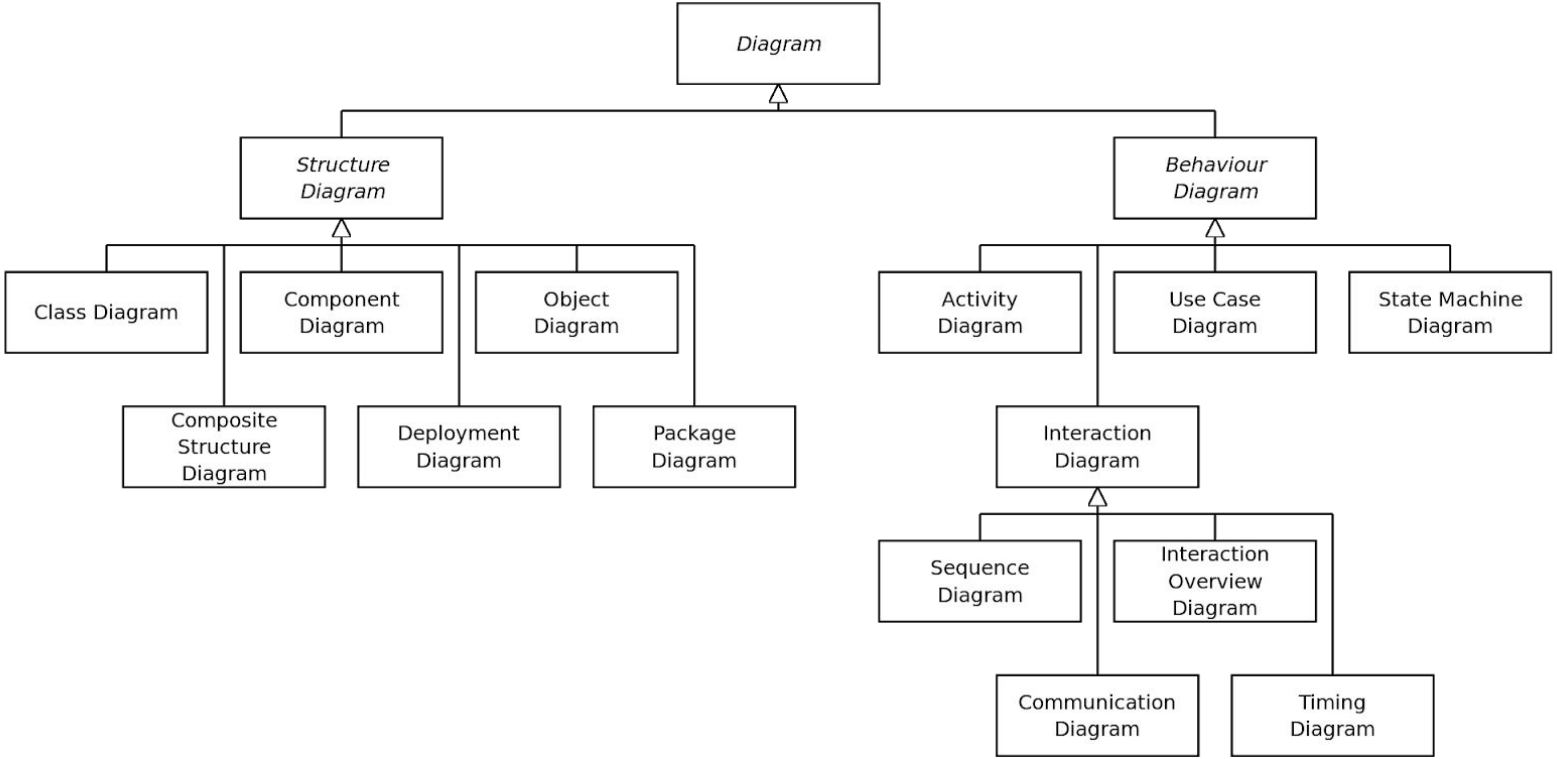
2 UML

“Unified Modeling Language, is a way of visualizing a software program using a collection of diagrams”

UML Benefits

- Illustrate data models for information systems, no matter how simple or complex.
- Better understand the general overview of the schematics of an application.
- Visually express any specific needs of a system and disseminate that information throughout the business.
- Create detailed charts that highlight any specific code needed to be programmed and implemented to the described structure.
- Provide an implementation-independent description of types used in a system that are later passed between its components.

Types of UML Diagrams



Class Diagrams

A Class Diagram defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity.

Benefits:

- Class diagrams are simple and fast to read.
- Class diagrams give you a sense of orientation.
- Class diagrams provide detailed insight into the structure of your systems.

MyClass Name
+attribute 1 : int -attribute 2 : float
+op1(in p1 : boolean) : string -op2(inout p3 : int) : float #op3(out p6) : circle

Class Diagrams

Class Name

- The name of the class appears in the first partition.

Class Attributes

- Attributes are shown in the second partition.
- The attribute type is shown after the colon.
- Attributes map onto member variables (data members) in code.

MyClass Name
+attribute 1 : int -attribute 2 : float
+op1(in p1 : boolean) : string -op2(inout p3 : int) : float #op3(out p6) : circle

Class Diagrams

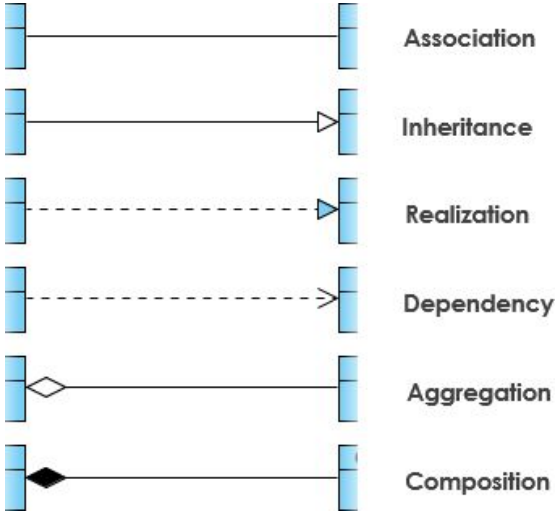
Class Operations (Methods)

- Operations are shown in the third partition. They are services the class provides.
- The return type of a method is shown after the colon at the end of the method signature.

MyClass Name
+attribute 1 : int -attribute 2 : float
+op1(in p1 : boolean) : string -op2(inout p3 : int) : float #op3(out p6) : circle

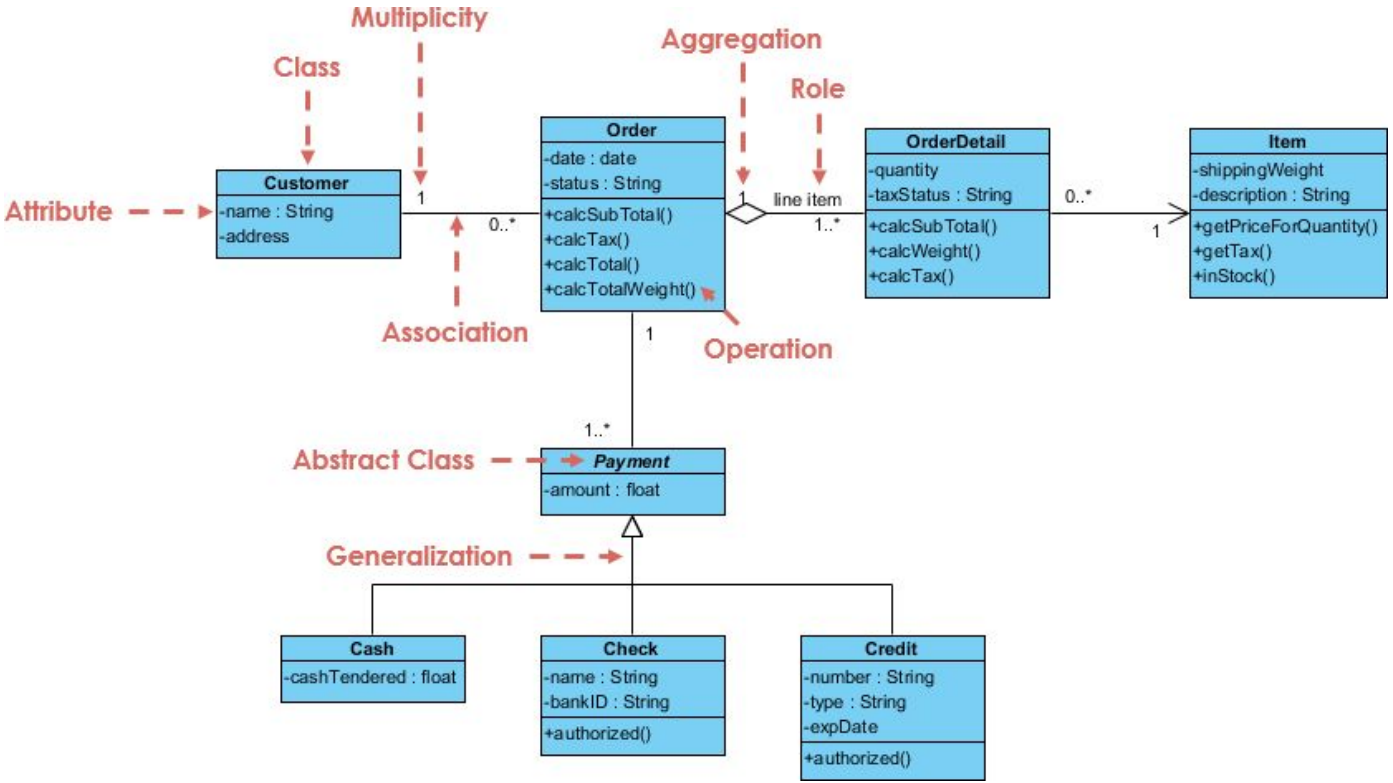
Class Diagrams

MyClass Name1
+attribute 1 : int -attribute 2 : float
+op1(in p1 : boolean) : string -op2(inout p3 : int) : float #op3(out p6) : circle



MyClass Name2
+attribute 1 : int -attribute 2 : float
+op1(in p1 : boolean) : string -op2(inout p3 : int) : float #op3(out p6) : circle

Class Diagrams



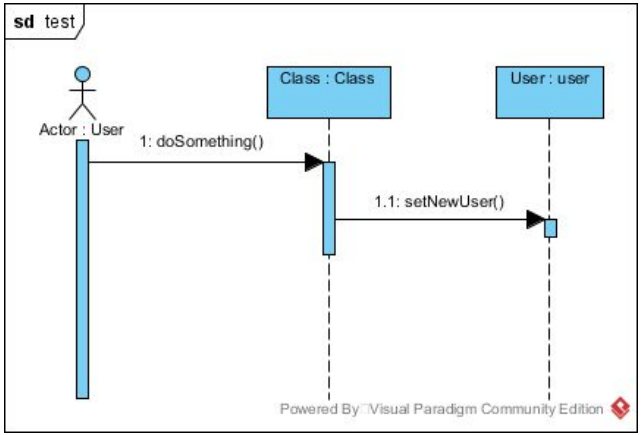
Mo.Agamia

Sequence Diagrams

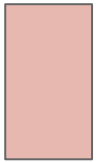
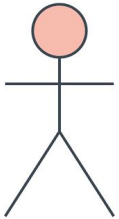
A Sequence Diagram is a type of interaction diagram that describes how—and in what order—a group of objects works together.

Benefits:

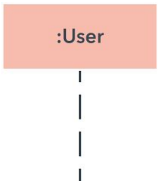
- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.



Sequence Diagrams

Symbol	Name	Description
	Object symbol	Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system. Class attributes should not be listed in this shape.
	Activation box	Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.
	Actor symbol	Shows entities that interact with or are external to the system.

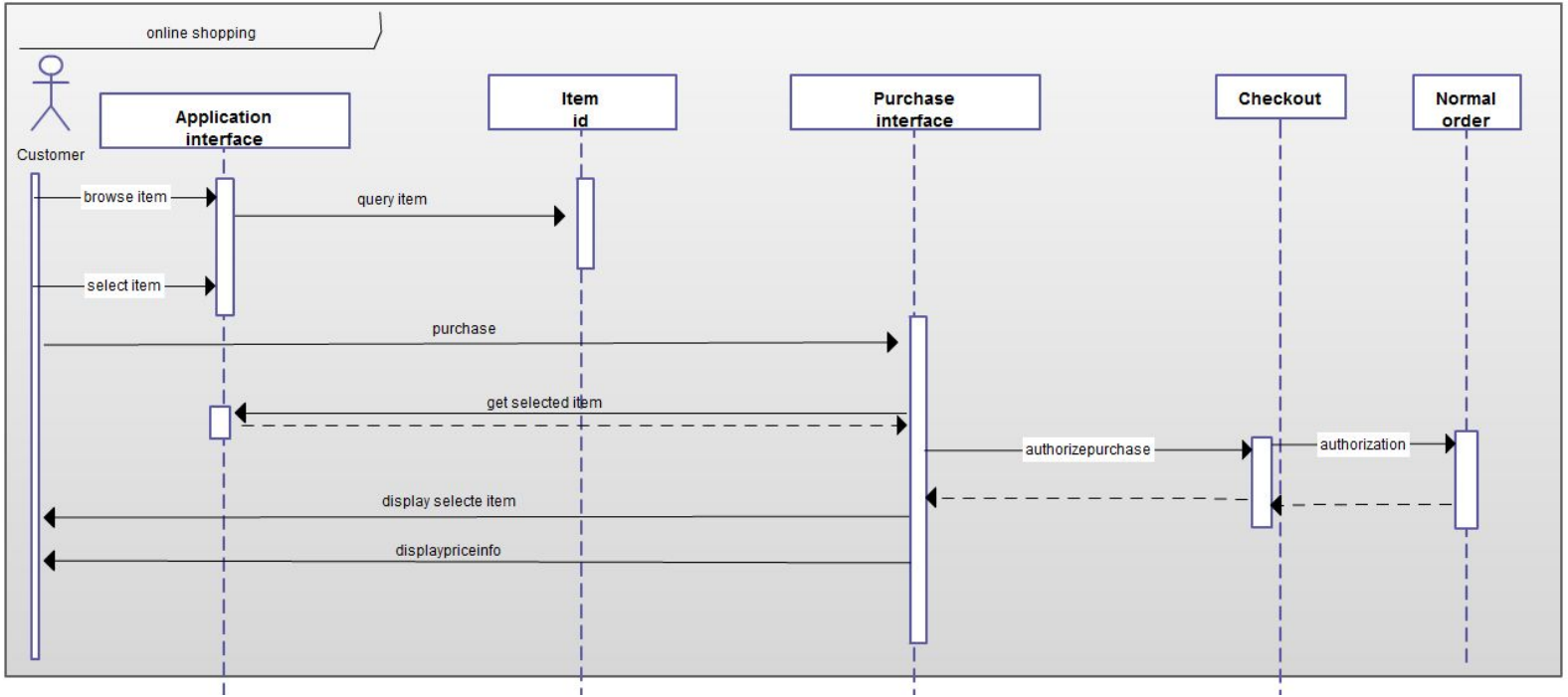
Sequence Diagrams

Symbol	Name	Description
	Package symbol	Used in UML 2.0 notation to contain interactive elements of the diagram. Also known as a frame, this rectangular shape has a small inner rectangle for labeling the diagram.
	Lifeline symbol	Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labeled rectangle shape or an actor symbol.

Sequence Diagrams

Symbol	Name	Description
	Synchronous message symbol	Represented by a solid line with a solid arrowhead. This symbol is used when a sender must wait for a response to a message before it continues. The diagram should show both the call and the reply.
	Asynchronous message symbol	Represented by a solid line with a lined arrowhead. Asynchronous messages don't require a response before the sender continues. Only the call should be included in the diagram.
	Delete message symbol	Represented by a solid line with a solid arrowhead, followed by an X. This message destroys an object.

Sequence Diagrams



SRS

“A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for software under development”

SRS

A Sequence Diagram fully describes what the software will do and how it will be expected to perform.

Benefits:

- It provides feedback to the customer.
- It decomposes the problem into component parts.
- It serves as an input to the design specification.
- It serves as the parent document

Includes:

- Interfaces
- Functional Capabilities
- Performance Levels
- Data Structures/Elements
- Safety
- Reliability
- Security/Privacy
- Quality
- Constraints and Limitations

SRS Structure

1. Introduction
    - 1.1. Purpose
    - 1.2. Document conventions
    - 1.3. Intended audience
    - 1.4. Additional information
    - 1.5. Contact information/SRS team members
    - 1.6. References
2. Overall Description
    - 2.1. Product perspective
    - 2.2. Product functions
    - 2.3. User classes and characteristics
    - 2.4. Operating environment
    - 2.5. User environment
    - 2.6. Design/implementation constraints
    - 2.7. Assumptions and dependencies
3. External Interface Requirements
    - 3.1. User interfaces
    - 3.2. Hardware interfaces
    - 3.3 Software interfaces
    - 3.4 Communication protocols and interfaces
4. System Features
    - 4.1. System feature A
        - 4.1.1. Description and priority
        - 4.1.2. Action/result
        - 4.1.3. Functional requirements
    - 4.2. System feature B
5. Other Nonfunctional Requirements
    - 5.1. Performance requirements
    - 5.2. Safety requirements
    - 5.3. Security requirements
    - 5.4. Software quality attributes
    - 5.5. Project documentation
    - 5.6. User documentation
6. 6. Other Requirements

Appendix A: Terminology/Glossary/Definitions list

Appendix B: To be determined

Thanks!

Any questions?

You can find me at:

- Github: https://github.com/MoAgamia/SE-Boot-Camp

References

- https://www.smartsheet.com/agile-vs-scrum-vs-waterfall-vs-kanban
- https://www.smartsheet.com/agile-project-management-excel-templates
- https://www.smartdraw.com/uml-diagram/
- https://techwhirl.com/writing-software-requirements-specifications/
- https://www.versionone.com/agile-101/
- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/
- https://www.lucidchart.com/pages/uml-sequence-diagram
- http://asingh.com.np/blog/ieee-srs-recommendations/
- http://www.cse.chalmers.se/~feldt/courses/reqeng/examples/srs_example_2010_group2.pdf