

Status Reports

- Please email your team's weekly status report to me by **8:00 a.m.** every Monday morning recapping the prior week's progress
- Received today:
 - Resurface
 - Caliber Public Safety #2
 - NASA/JPL
 - Insights Intervention
 - Lockheed-Martin # 2
 - Union Pacific PST
 - NCAR
 - Mind Be Well
 - Festo 1
 - Innovar
 - CU Psychology
 - Caliber Public Safety # 1
 - Terumo BCT
 - Leeds School of Business
 - CEAS Advancement
 - Full Contact
 - NMBL

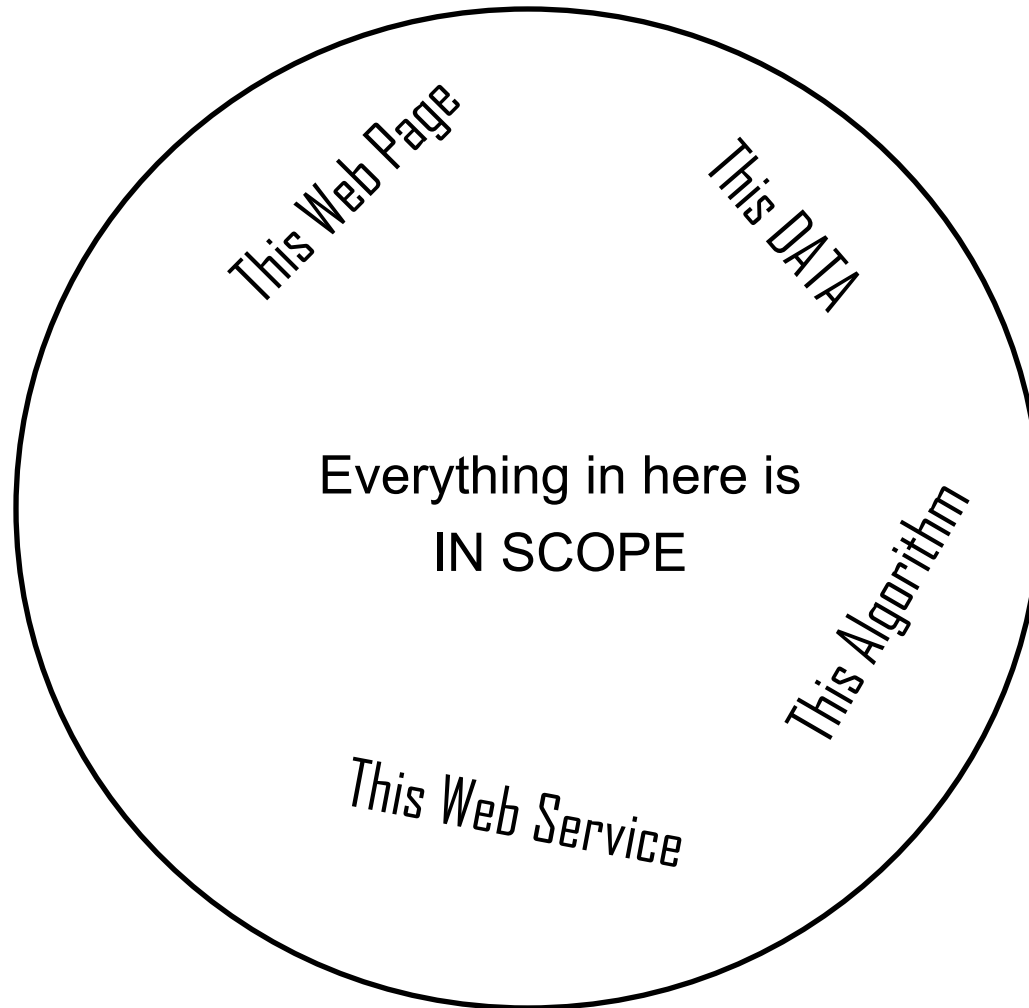
- Missing:
 - BI Inc. # 1
 - BI Inc. #2
 - CU E-BIO
 - Digiclips
 - Festo # 2
 - Trimble
 - UCAR
 - UR Turn
 - Gloo
 - Lockheed-Martin # 1

- Scope
 - Easily understood as a sum of REQUIREMENTS

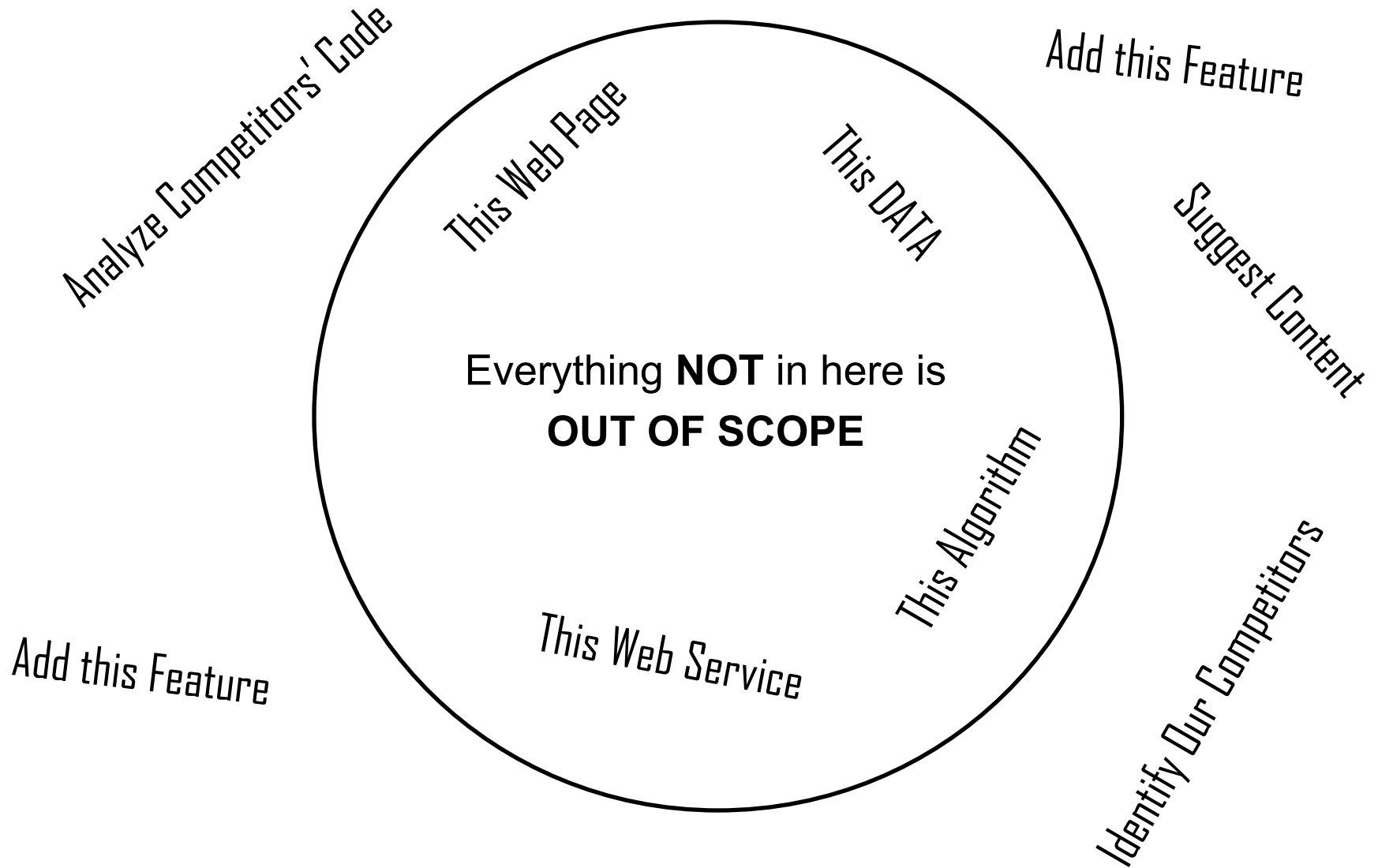
“If it can’t do [...this feature...], I don’t want it.”

“It must be able to [...this feature...]”

“I want a system that will [.... do this ...] ”



Requirements



High Level Requirements

- Are set in the **Project Charter**
- Established very **early on** in the life of the project
- **Signed Off** – Agreed to by Sponsor
- Become the baseline for **SCOPE management**

Why do a large percentage of projects fail ??

Failure to manage **SCOPE** ...

➔ Failure to manage **REQUIREMENTS**

You must learn to manage your sponsor's **expectations**.

- Put yourself in their shoes.
- What is motivating them?
- What's in it for the sponsor's organization?
- What do they really want you to do?

Managing sponsor's **expectations**.

If it isn't written down, it didn't happen...

Managing sponsor's **expectations**.

If it isn't written down, it didn't happen...

You cannot manage what you do not measure...

Managing sponsor's **expectations**.

If it isn't written down, it didn't happen...

You cannot manage what you do not measure...

People do what you inspect, not what you expect.

Scope must be written down and signed off.

Any changes to scope must be carefully managed.

Scope = Sum of requirements

Requirements must be written down and signed off.

Any changes to requirements must be carefully managed.

Managing Requirements

1. Write them down
2. Number them: unique identifiers for measuring and tracking
3. Define a change control process
 - Changes to scope/requirements may be considered, but require careful, formal, disciplined control
 - Document requests, discuss, vote, decide, document results
4. Keep requirements in a distinct place within your repository

Managing Requirements


AND, bring ANY questions or concerns to ME

So, let's talk about defining and managing requirements...

- Two types of Requirements:
 - Functional Requirements ➔ “what” – (Users’ view)
 - Non-Functional Requirements ➔ “how” – (Developers’ view)

Requirements

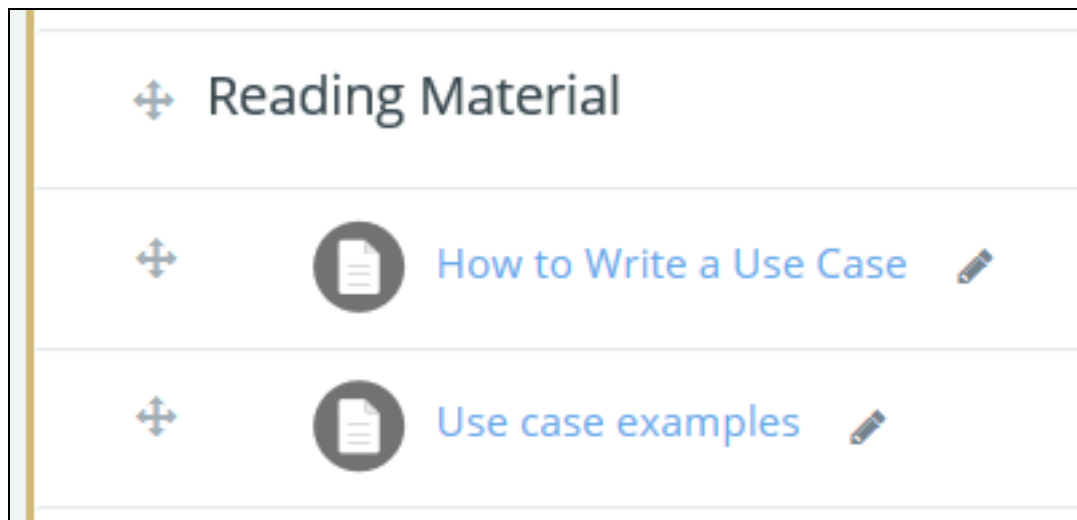
Often stated as User Stories

	A	B	C	D
1	Agile User Story Template			
2				
3				
4				
5	User Story ID	As a <type of user>	I want to <perform some task>	so that I can <achieve some goal>
6	1	Project manager	View a status report from each team member	Ensure the project stays on track.
7	2	Employee	Be reminded of upcoming deadlines	Complete my tasks on time.
8	3	Director	See the big picture view of department work	Stay in the loop.
9				
10				

Functional Requirements

- User Requirements, Business Requirements
 - Documented by **Use Case**
 - The sum of the Use Cases = “User Requirements”
 - These requirements describe the users’ expectations for **what** the system does for them
 - “**WHAT** I want the system to do for our organization”
 - A functional requirement for an everyday object like a mug would be: “ability to contain tea or coffee without leaking”.
- The definition of a functional requirement is:
 - “Any requirement which specifies **what** the system should do.”

- **Use Cases document your users' requirements**
- **Review this week's readings**



- **The format of your Requirements Document can be**
 - **A table (rows, columns, cells)**
 - **A text document with paragraphs/sentences or bullet points**
 - **A picture showing “actors” and “flows”**
 - **EXAMPLE: https://en.wikipedia.org/wiki/Use_case**

- [Martin Fowler](#) (leading industry expert) states:

"There is no standard way to write the content of a use case, and different formats work well in different cases."

He describes "a common style to use" as follows:

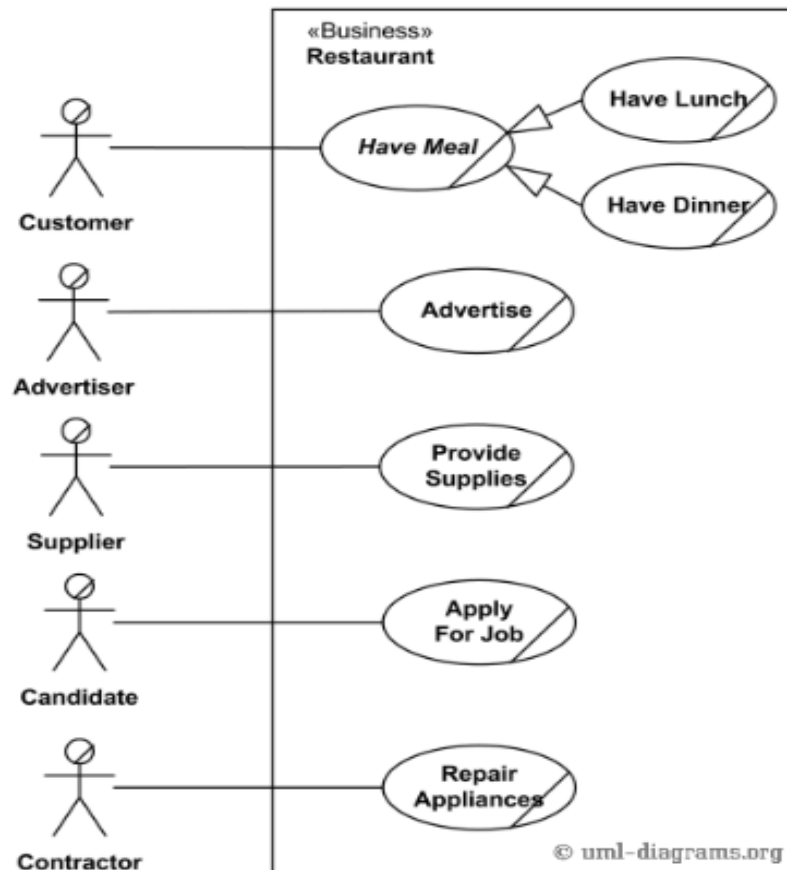
- Title: "goal the use case is trying to satisfy"
- Main Success Scenario: numbered list of steps
 - Step: "a simple statement of the interaction between the actor and a system"

Restaurant

UML Use Case Diagram Example

Here we provide two alternative examples of a **business use case** diagram for a Restaurant, rendered in a notation used by **Rational Unified Process** (RUP).

First example shows **external business view** of a restaurant. We can see several **business actors** having some needs and goals as related to the restaurant and **business use cases** expressing expectations of the actors from the business.



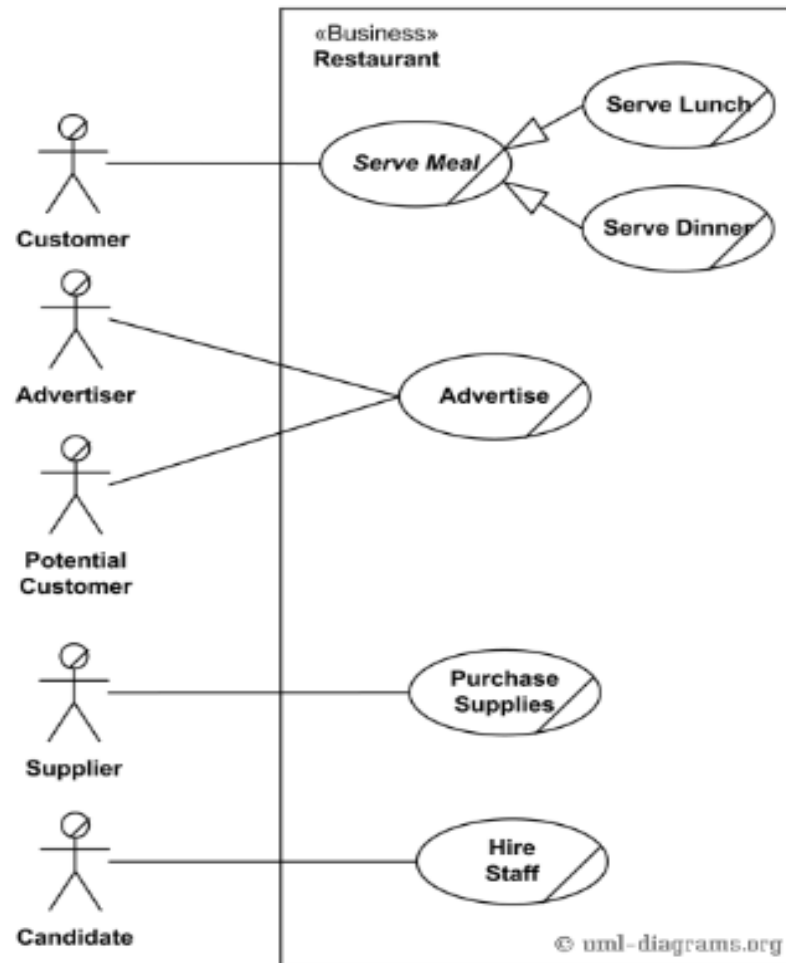
Business use case diagram for Restaurant - External view

For example, **Customer** wants to **Have Meal**, **Candidate** - to **Apply for Job**, and **Contractor** - to fix some appliances. Note, that we don't have such actors as **Chef** or **Waiter**. They are not external roles but part of the business we model - the Restaurant, thus - they are not actors. In terms of RUP Chef and Waiter are **business workers**.

Second example shows **internal business view** of a restaurant. In this case we can see that restaurant has several business processes represented by **business use cases** which provide some services to external **business actors**. As in the previous example, actors have some needs and goals as related to the restaurant.

This approach could be more useful to model services that the business provides to different types of customers, but reading this kind of business use case diagrams could be confusing.

For example, **Customer** is now connected to **Serve Meal** use case, **Supplier** - to **Purchase Supplies**. We have now new actor **Potential Customer** participating in **Advertise** use case by reading ads and getting some information about restaurant. At the same time, **Contractor** actor is gone because **Repair Appliances** is not a service usually provided by restaurants.



Business use case diagram for Restaurant - Internal view

Still, in this example we don't have actors as **Chef** or **Waiter** for the same reasons as before - they both are not external roles but part of the business we model.

Take a look at this:

http://web.cse.ohio-state.edu/~bair.41/616/Project/Example_Document/Req_Doc_Example.html

- Non- Functional Requirements
 - Ties Use Cases to Technical Implementation
 - Describes **HOW** the system should behave
 - A non-functional requirement for the cup mentioned previously would be:

“contain hot liquid without heating up to more than 45 ° C”.
 - These requirements impact the users’ experience
 - For example, “The home page must load within 1.4 seconds.”
 - Or, “If a document is flagged as private, only the user who created it can see it.”
- The definition of a non-functional requirement is:

“ Any requirement which specifies **how** the system performs a certain function.”

Examples of non-functional requirements

- **Interface requirements**
 - Field 1 accepts numeric data entry.
 - Field 2 only accepts dates before the current date.
 - Screen 1 can print on-screen data to the printer.
- **Regulatory/Compliance Requirements**
 - The database will have a functional audit trail.
 - The system will limit access to authorized users.

So where do we go from here?

- You must define a high-level **SCOPE** statement in your Project Charter.
- Your sponsor must sign off on the **Project Charter** which includes a high level statement of **SCOPE**
- The high-level scope statement must be drilled down to individual **REQUIREMENTS**, each identified and tracked.
- Each requirement must be documented and described at a detailed level.
- Requirements documents must be tracked carefully in a repo.
- Any changes to requirements must be carefully managed through a **Change Control** process

So where do we go from here?

- Once your Sponsor has signed off on your Charter:
- Begin the process of documenting your REQUIREMENTS
- How?
 - Start with your scope and objectives (from the charter)
 - Brainstorm, and capture all team members' inputs
 - Features
 - Functionality
 - User Stories, Use Cases
 - Data (what is it? where does it come from? who updates it, when?)
 - Audience (Users of your system)
 - User interface (who uses the system for what purpose, when?)
 - Organize – Functional versus Non-Functional
 - Organize – by Features, Stories, Use Cases, Data

Suggestion:

- Get started this week
- Talk about it
- Get some feedback from your TA (and/or your instructor)