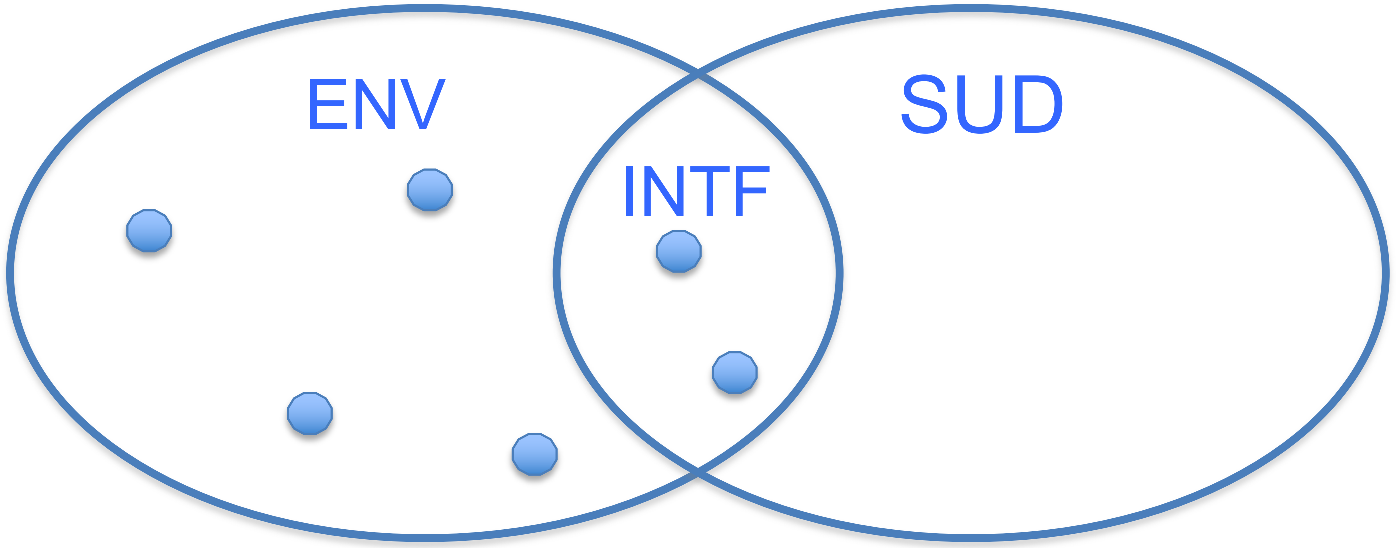
CS445 / SE463 / ECE 451 / CS645
Software requirements specification & analysis

A reference model for requirements engineering

Spring/Summer 2023

Mike Godfrey & Daniel Berry & Richard Trefler

Reference model



R – Reqs live in ENV (incl. INTF)
S – Spec lives in INTF, describes behavior of SUD
D – Domain knowledge lives in ENV (incl. INTF)

Reference model

Thus, if we enlarge our model to include domain knowledge, then the ZJVF must hold:

$$\mathbf{D, S \vdash R}$$

- D is domain knowledge
- S is the spec
- R is the reqs

Reference model

$$\mathbf{D, S \vdash R}$$

- The spec describes the behavior of a system that is supposed to realize the reqs.
- The domain assumptions are needed to argue that any system that meets the spec, and that manipulates the interface phenomena, will satisfy the original reqs.

Scenarios and Use Cases

Daniel M. Berry

Use Case

Each such particular way to use *S* is called a *use case*.

It is one case of the many ways to use *S*.

A use case (UC) is expressed in natural language as a simple imperative sentence, e.g.:

- **Insert a coin into the coinSlot**
- **Push and walk through the barrier**

Scenario

A UC should not be confused with a closely related concept called a *scenario*. A scenario of *S* is a particular sequence of interaction steps between a user of *S* and *S*.

Use Cases and Scenarios

The relation between UCs and scenarios can make both terms clearer.

A single use case contains many, many scenarios.

A UC *U* of *S* has a so-called *typical* scenario. This scenario is that identified by the stakeholders of *S* as being the normal case of *U* that proceeds with all decisions being made in the so-called normal or typical way.

Variations

A UC consists also of variations called *alternatives* and *exceptions*.

Alternatives (As)

An alternative of UC *U* is a sub-use-case that achieves the main goal of *U* through different sequences of steps or fails to achieve the goals of *U* although it follows most of the steps of *U*.

Exceptions (Es)

An exception of UC *U* is a sub-use-case that deals with the conditions along the typical scenario and other sub-use-cases that differ from the norm and those already covered.

As vs Es

The distinction between alternatives and exceptions is not really important, …

so long as you find all of *both* of them.

So think of them as prompts that help you find all of them.

Brainstorm to Find As & Es!

Brainstorm to find as many alternatives and exceptions as you can!

You can always throw out ones that prove useless later on.

It helps to have a twisted psyche, a devious, diabolical mind!

Heh heh heh!!

Deliverable: List of Domain Assumptions, Exceptions, and Variations for Your System

This deliverable is a complete list of assumptions, exceptions, and variations for your system.

An *assumption* is anything that must be true about the real world in order for the use cases of your sytem to work as expected. For now, write down *all* the assumptions that you can find,

- whether or not they are explicit in all documents you have produced before, and
- *whether or not they are true of the real world*.

(Later, i.e., **not in this deliverable, but in the specification that is Deliverables 4 and 5**, you will decide what your system does about each assumption.

To help you understand what an assumption is, know now that what you will say that your system does about each assumption, *A*, is one of the three following:

- your system does nothing because *A* is true of the real world,
- if *A* is not true of the real world, your system acts as if *A is* true, or
- if *A* is not true of the real world, your system provides functionality that eliminates the need for *A*.)

An *exception* is any condition that may cause a use case in your system not to work as intended or expected. Exceptions may overlap with assumptions in the sense that the failure for an assumption to hold may give rise to exceptions.

A *variation* is a use case that is slightly different from another use case, which itself might be a variation. It might represent a different way to achieve the other use case's goals or a way to achieve a different goal with approximately the same behavior as the other use case.

In making this list, don't hold back. Treat it like brainstorming for assumptions, exceptions, and variations, i.e., *anything* that could be different from the normal behavior of *any* use case.

Later, as you decide your system's responses to the items on this list, you will trim the list back to the truly relevant items.

Please make one numbered sublist for assumptions, one numbered sublist for exceptions, and one numbered sublist for variations. If you cannot decide which sublist an item should be in, just pick one sublist, and go on!

You may include in this deliverable a list of *any* question that you have for your customer about your system. These should be clearly marked as questions.