CIS*3750 - System Analysis and Design in Applications

Luiza Antonie, Fall 2025, University of Guelph

What are Requirements?

Requirements

- What the software does
- What it does, not how it does it
- Functional or non-functional
- Affects all levels of development

Good Requirements…

- Define the **users**
- Define **non standard terms**
- Define **user terms**

Good Requirements…

- Are **categorized**
- Are **prioritized**
- Flow **logically**
- Have **time estimates**

Good Requirements...

- Come from the **client**
- Can also come from **user** needs
- May come from **programmer**

Things to consider...

- Specificity
- Importance

What is specificity?

- Clearly describe the scope of the requirement
- Include testable definitions of performance

- The <user/system type><MuSCoW>, <do one thing><for some measurable reason>.

Which of these requirements has specificity?

- The system must respond to a search query in less than 2 seconds to provide a list of 50 most relevant results

- The system must allow the user to search for a list of results

Which of these requirements has specificity?

- [✓ The system must respond to a search query in less than 2 seconds to provide a list of 50 most relevant results]

- The system must allow the user to search for a list of results

Requirements that have specificity

- Can be measured in some way
- Aren’t open to interpretation by others

What is importance?

- A clear understanding between developer and client of what the expectations of the final product are

How do we ensure clear expectations?

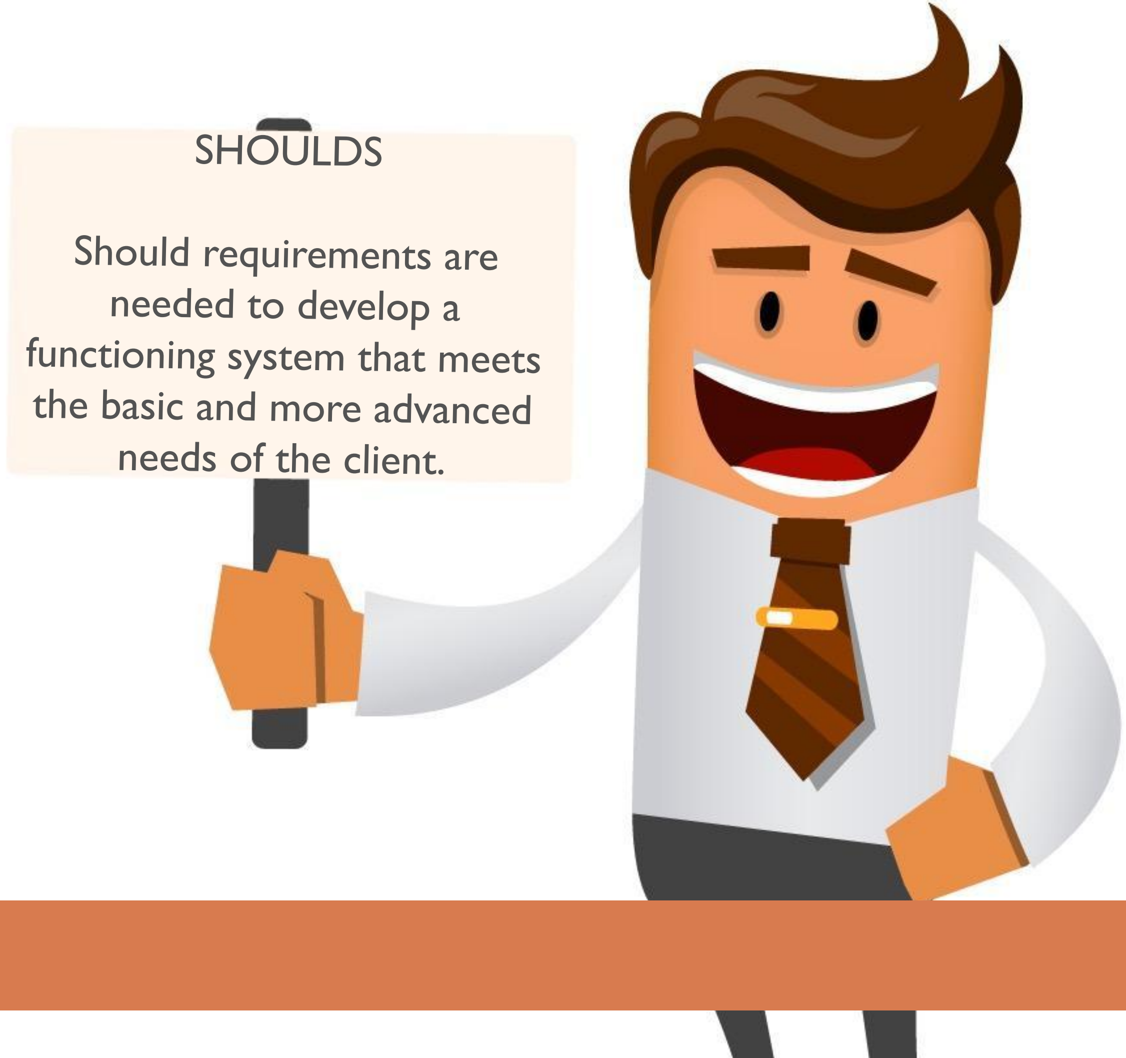
- We ask the client to help us rank our requirements so we have a better sense of what needs to happen first

Requirements categorization

We use the MuSCoW Method to Categorize our Requirements!

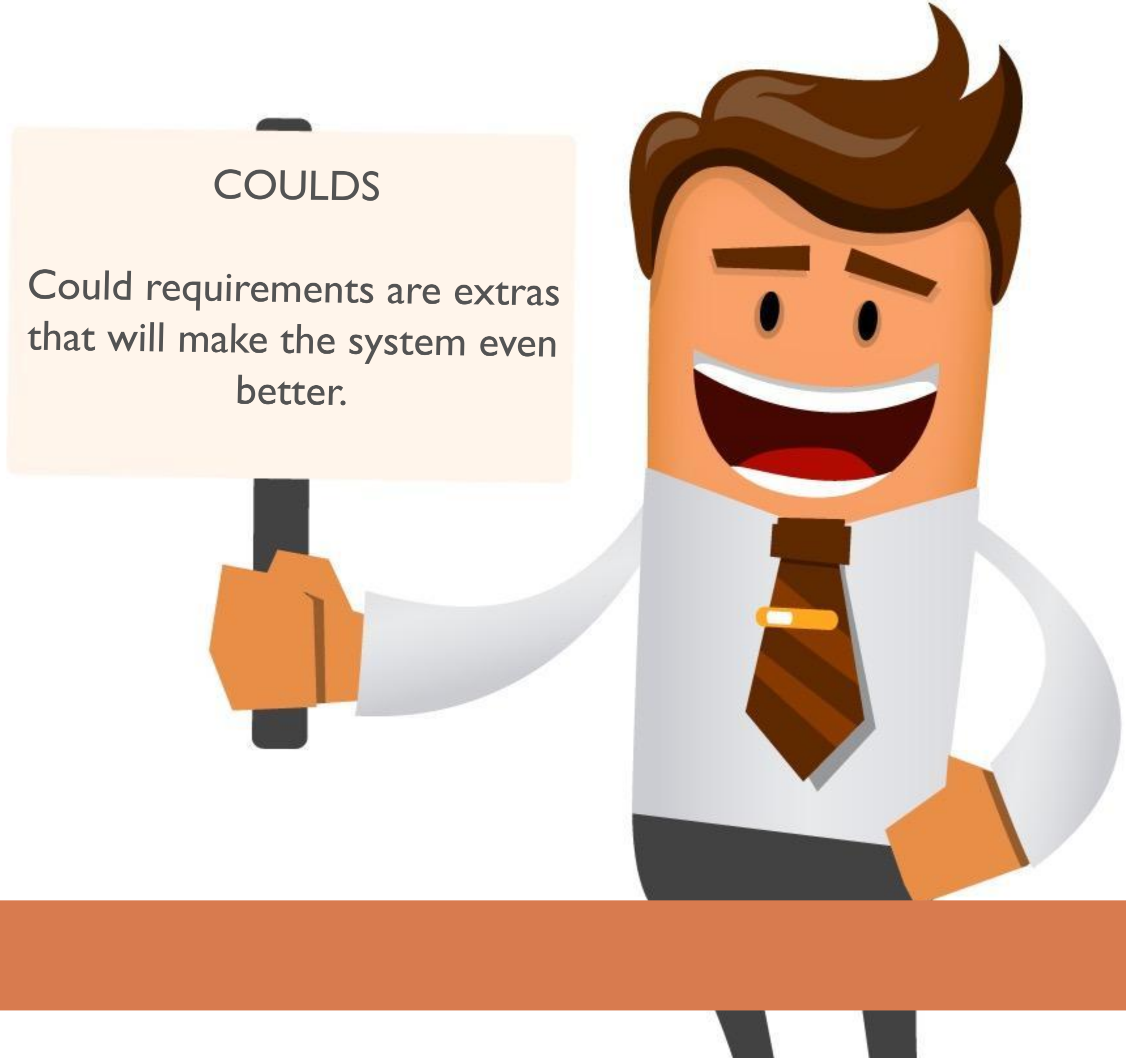
MUSTS

Must requirements are needed to develop a functioning system that meets the basic needs of the client.



SHOULDS

Should requirements are needed to develop a functioning system that meets the basic and more advanced needs of the client.

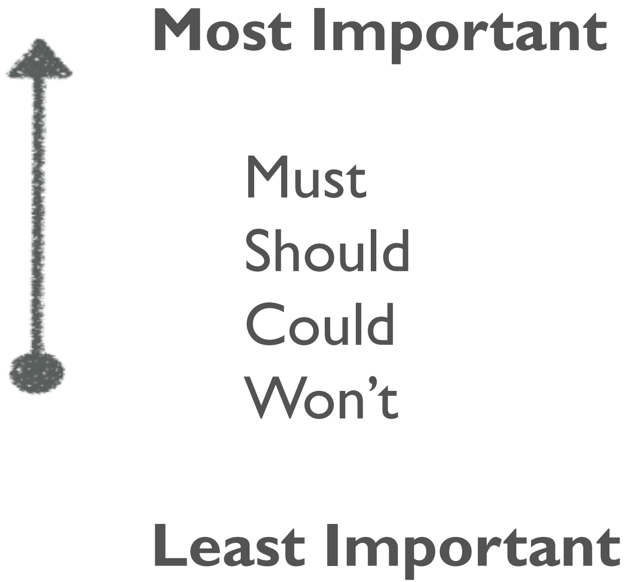


COULDS

Could requirements are extras that will make the system even better.

WONTS

Won’t requirements are those things that the system won’t do.



Most Important

Must

Should

Could

Won’t

Least Important

Requirements need

- ★ Categorization
- Prioritization
- Logic
- Time Estimates

Importance!

Requirements need

- ★ **Categorization**
- **Prioritization**
- **Logic**
- **Time Estimates**

MuSCoW!

Requirements vs User Stories

	Requirements	User Stories
Actors Involved	Users, System	Users
Scope	Functional/ Non-functional	Functional
Extent	What it does, not how it does it	What it does, not how it does it

From User Stories to Requirements

- As an instructor I want to be able to create and update announcements on the course website so that students can see those announcements.
- The instructor user must be able to create an announcement that will appear on the Home page.
- The instructor user should be able to update an announcement that will appear on the Home page.

Prioritization

What is the most important requirement?

Picking the most important requirement is a challenge, especially in a team.

To prioritize
we assign values
of 10, 20, 30, 40, or 50
to each of our
requirements

Most Important

10
20
30
40
50

Least Important

Good prioritization might look like this:

- Must: 10, 20, 30
- Should: 30, 40
- Could: 40, 50
- Won’t: no priority

Good prioritization might also look like this:

- Must: 10, 20
- Should: 30, 40, 50
- Could: 50
- Won't: no priority

Bad prioritization might look like this:

- Must: 10, 20, 30
- Should: 20, 30, 40
- Could: 40, 50
- Won’t: 50

Bad prioritization might look like this:

- Must: 10, 20, 30
- Should: 20, 30, 40
- Could: 40, 50
- Won't: 50

Requirements need

- ★ Categorization
- ★ Prioritization
- Logic
- Time Estimates

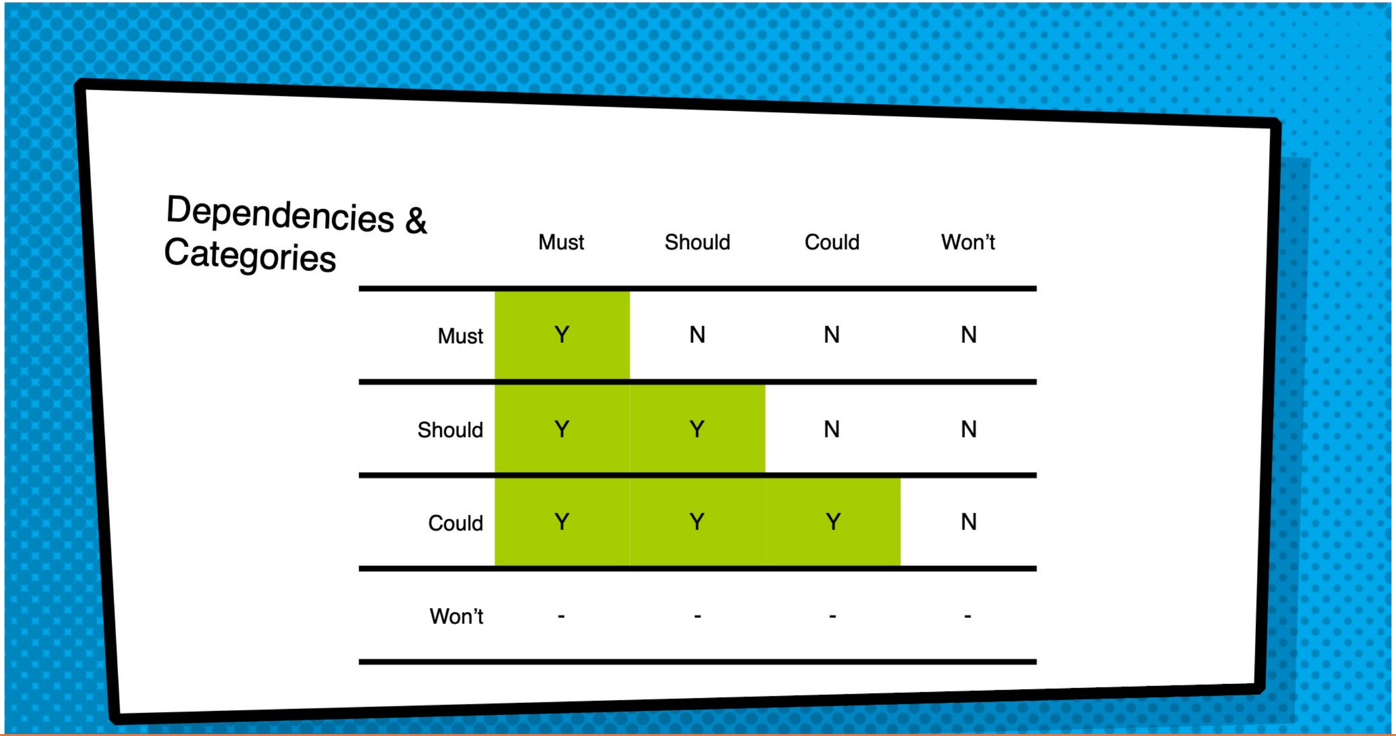
Flow Logically

Do any of the requirements depend on the successful completion of other requirements?

Requirement X cannot depend on Requirement Y, if ID(X)<ID(Y).

Requirement X cannot depend on Requirement Y, if category(X)>category(Y)

Dependencies



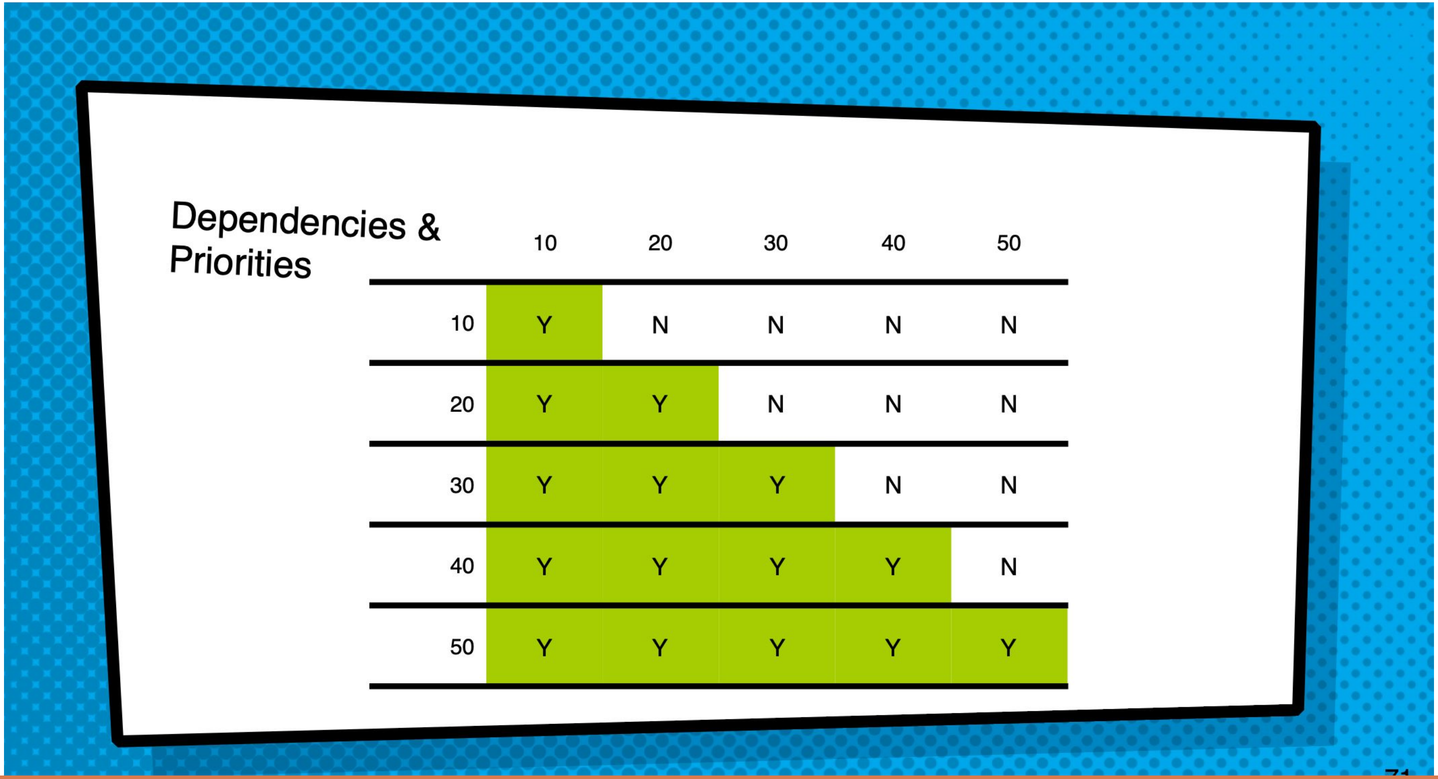
Dependencies & Categories

	Must	Should	Could	Won’t
Must	Y	N	N	N
Should	Y	Y	N	N
Could	Y	Y	Y	N
Won’t	-	-	-	-

MUST>SHOULD>COULD>WON’T

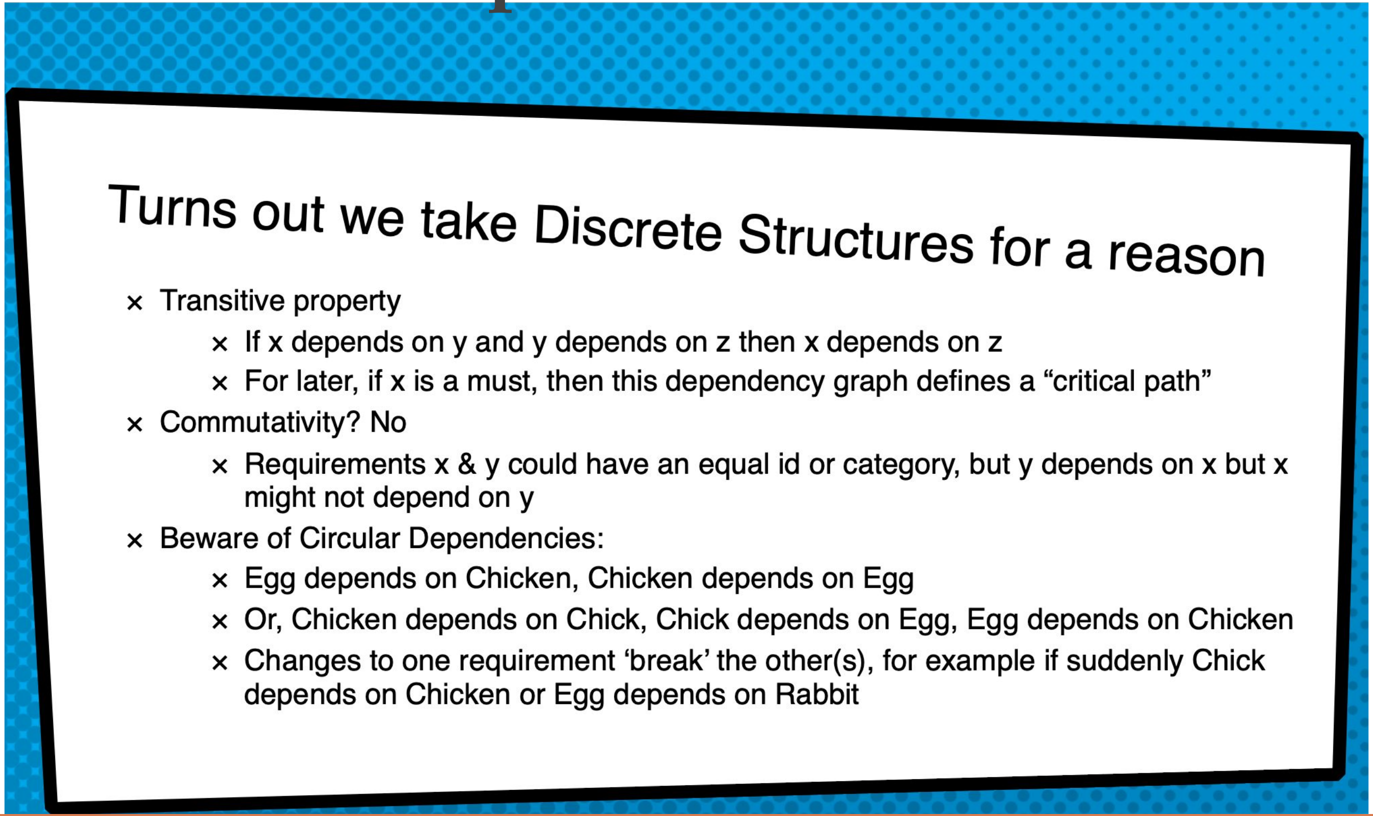
Requirement X cannot depend on Requirement Y, if priority(X)<priority(Y).

Dependencies



Dependencies & Priorities	10	20	30	40	50
10	Y	N	N	N	N
20	Y	Y	N	N	N
30	Y	Y	Y	N	N
40	Y	Y	Y	Y	N
50	Y	Y	Y	Y	Y

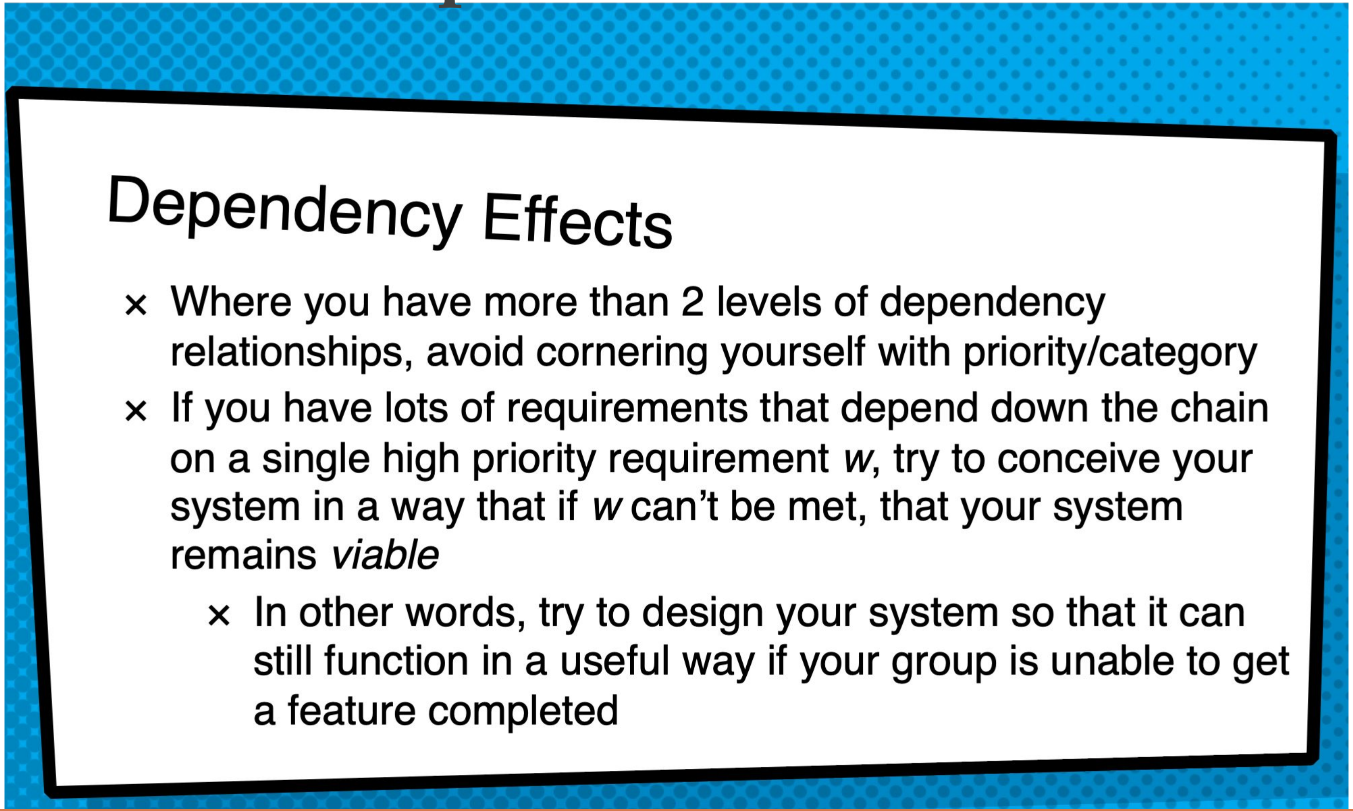
Dependencies



Turns out we take Discrete Structures for a reason

- Transitive property
  - If x depends on y and y depends on z then x depends on z
  - For later, if x is a must, then this dependency graph defines a “critical path”
- Commutativity? No
  - Requirements x & y could have an equal id or category, but y depends on x but x might not depend on y
- Beware of Circular Dependencies:
  - Egg depends on Chicken, Chicken depends on Egg
  - Or, Chicken depends on Chick, Chick depends on Egg, Egg depends on Chicken
  - Changes to one requirement ‘break’ the other(s), for example if suddenly Chick depends on Chicken or Egg depends on Rabbit

Dependencies



Dependency Effects

- Where you have more than 2 levels of dependency relationships, avoid cornering yourself with priority/category
- If you have lots of requirements that depend down the chain on a single high priority requirement *w*, try to conceive your system in a way that if *w* can’t be met, that your system remains *viable*
  - In other words, try to design your system so that it can still function in a useful way if your group is unable to get a feature completed

Time Estimates

Read the requirement on the next slide and ...

Write down how long you think this would take one programmer to complete.

A researcher
must be able to generate
an alphabetized list
(by last name)
of all salary records,
sorted by sector,
who are earning more than 250k.

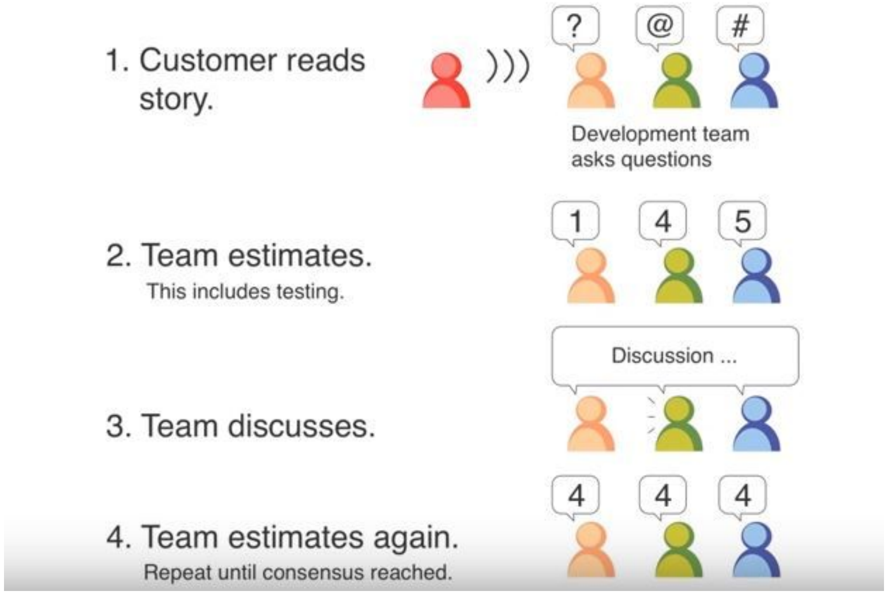
Time Estimates should be based on the question:

What is the minimum work that someone could do and still claim they satisfied the requirement?

- Assume everything else that needs to be complete is complete

- Estimate time *only* for the requirement, not for the requirement and its dependencies

Planning Poker



1. Customer reads story.

2. Team estimates.

This includes testing.

3. Team discusses.

4. Team estimates again.

Repeat until consensus reached.

Image Source: https://www.c-sharpcorner.com/article/agile-story-point-estimation-techniques-planning-poker/

Hand out the cards

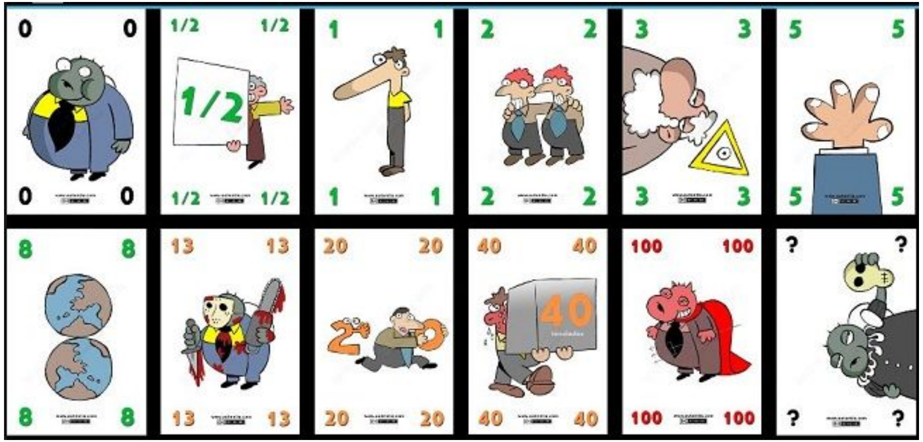
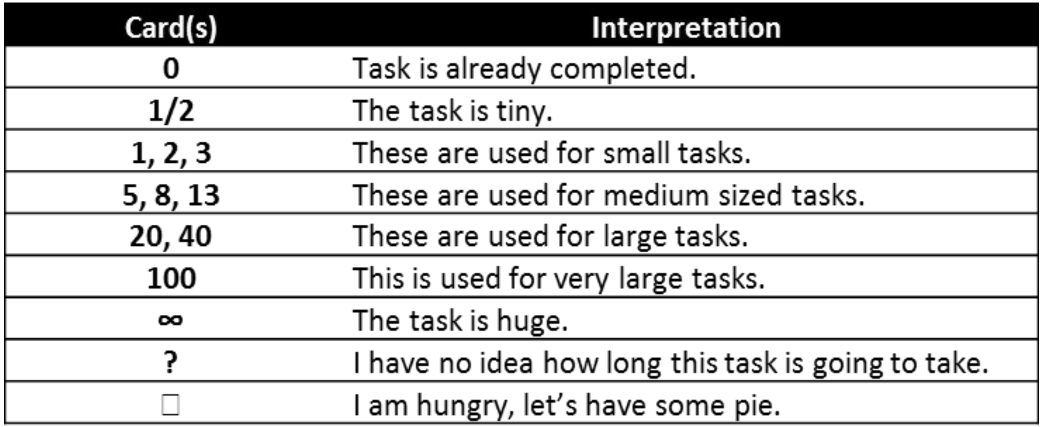


Image Source: Autentia Planning Poker

Hand out the cards



Card(s)	Interpretation
0	Task is already completed.
1/2	The task is tiny.
1, 2, 3	These are used for small tasks.
5, 8, 13	These are used for medium sized tasks.
20, 40	These are used for large tasks.
100	This is used for very large tasks.
∞	The task is huge.
?	I have no idea how long this task is going to take.
□	I am hungry, let’s have some pie.

Image Source: http://www.leanmath.com/blog-entry/agile-planning-projects-poker-and-pi

Read the story

As an instructor I want to be able to create announcements on the course website so that students can see those announcements on the landing page.

Discuss the story

- How many people will work on this story?
- What skills are needed to work on this story?
- Do we have any design elements already developed?
- Are there any dependencies? Are these dependencies complete?

Vote and share your vote

5
5
5
8
Susan
Vadim
Ann
Chris

Image Source: https://medium.com/@scrum.master/play-planning-poker-in-scrum-team-c5c12b47bd55

If everyone agrees,
what does this mean?

If nobody agrees,
what does this mean?

- In either case, explain the assumptions that brought you to your decision

- Discussing assumptions helps identify **new requirements**, **misunderstandings**, or things that not everyone has thought about

- Discussing assumptions might also identify a **requirement** that you hadn’t thought to include

- New requirements mean you have to check their priority and dependencies…
- This means you might have to restructure your requirements document!
- Careful planning prevents this

Planning poker flow

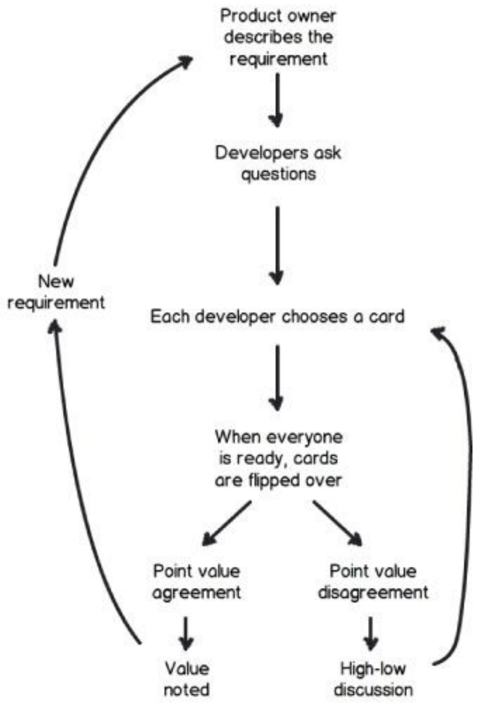

Product owner describes the requirement
Developers ask questions
Each developer chooses a card
When everyone is ready, cards are flipped over
Point value agreement
Point value disagreement
Value noted
High-low discussion
New requirement

Image Source: https://www.axisagile.com.au/blog/estimation/planning-poker-at-pace/