Analysing the old specification

Vagueness and ambiguity

Can you satisfy any requirements document?

- What if the specification is vague or ambiguous?
- Is it possible to write a piece of software satisfying unclear or contradictory requirements?
- Always be thorough and careful when reading specifications
- We’ll look at some of the oddities with the last assignment specifications

Requirement - what’s problematic about this?

“The Cave table contains the basic information for a room”

Ideas?

“The Cave table contains the basic information for a room”

- What is a “Room”?
- The name of the table is “Cave” - but what is a “Room” in a “Cave”?
- Some “Room”s are outdoors - like in a forest
- Isn’t the table really the data for the “World” or the map of the game?
- Isn’t the text description part of the basic information for a “Room”?
  - The text information is, however, in a separate table Lines
  - Furthermore, a Room can have many Lines
- Isn’t the Thing in a room a part of the basic information for a “Room”?
  - Things are described, however, in a separate table Things

Requirement - what's problematic about this?

The way the database is designed.

How did the database design influence your Java design decisions?

How many had an int roomID in the Room class? (Hands up)

How many rooms in the real world have an ID?

Problems with the database

- Don’t let the database influence your design too much!
- The database might not be very object oriented, which might lead to a less object oriented design of your system
- Having multiple lines for a room makes initializing the “Cave” difficult and tedious
- Why should the database be concerned with linewidths?
- The things table contains the columns Visible and Attainable, what are the purposes of those columns?
  - What does it mean that a thing is invisible or unattainable?

How did you manage the Lines?

Did anyone implement a method appendLine(String) in the Room class?

Is it a common behavior for a Room to have an appendLine capability?

Remember this?

```
SELECT cave.roomid, north, south, east, west, line, linenr, thing FROM cave
 JOIN lines
  ON cave.roomid=lines.roomid LEFT JOIN things
  ON cave.roomid = things.roomid
LIMIT 20;
1|5|4|3|2|You are standing at the end of a road before a small brick building.|1|||
1|5|4|3|2|Around you is a forest.  A small stream flows out of the building and|2|||
1|5|4|3|2|down a gully.|3|||
2|0|5|1|0|You have walked up a hill, still in the forest.  The road slopes back|1|||
2|0|5|1|0|down the other side of the hill.  There is a building in the distance.|2|||
3|6|6|6|1|You are inside a building, a well house for a large spring.|1|Skeleton Key|1|1
...
```

Left join? Why? Not every room has a Thing. Notice the nulls above.
Why is roomid repeating? Because a Room (Cave?) can have more than one line.

Implementing the rules for things and rooms

In what table could you find information on the rules for e.g. the Bird?

In what table could you find information on the rules for e.g. the Snake?

Could you create the rules on-the-fly while reading from the database?

Problems with the database, continued

- The rules of the game are not encoded in the database
    - This makes programmatic initialization of the rules impossible or at least forces us to hard code the rules somehow
- The special rooms with creatures (the dragon and the snake) are not encoded in the database
    - The rooms have no database information about the snake or the dragon
- This means that these rules must be hardcoded by some other means

Requirement - what’s problematic about this?

Use ArrayList<Room> for the Cave.

Is there a natural order between Rooms? If you sort the list, what key would you use? RoomID?

Is there a correspondence between the index of the ArrayList and the roomid?

What are IDs used for?

How many of you used List<Room> rather than ArrayList<Room> when declaring the list of rooms?

What’s in the other end of the e.g. West exit from a Room?

Specification dictates choice of data types

The specification dictates that the tables should be read into “suitable ArrayLists”. How is an ArrayList suitable for the Room graph?

The IDs of the various rooms are not a sequence - the rooms are rather connected via routes to North, South, East or West. The connections are not always bi-directional - going North then South is not guaranteed to lead you back to the same room you came from.

Letting a Room have an ID is not very object oriented - the ID only makes sense in terms of the terms of the Cave table entries.

Specification dictates choice of data types, cont.

What other data types can you think of, in order to represent a graph like the world of the game where “rooms” are connected to each other?

You did have lectures on the Collections Framework?

The Map interface could be useful (if we decide to keep the concept of IDs for rooms).

Can a Room object have an instance variable of type Room? Can it have four such variables?

Could they be named north, south, east and west?

What data type did you use for the thing(s)?

How many used a String as the datatype for a Thing in a Room?

How many used a class Thing?

In the Room class, how many used a List<Thing> for the things?

How many used an ArrayList<Thing>?

Specification dictates choice of data types, cont.

If we decide that a Room can have more than one Thing, a List<Thing> could be suitable. Using String for the Thing variable is at least not a good choice, if we do allow more than one Thing.

The Player could have a List<Thing> to represent the inventory.

But the Cave should really not be a List<Room>. Two adjacent Rooms in the List do not necessarily share a connection to each other. There is absolutely nothing gained from using a List to store the Rooms.

Requirement - what's problematic about this?

"... commands in the form of buttons, icons or something else"

How did you implement the UI component for the command "Pick up"?

How did you implement the UI component for the command "Drop"?

If you used a button for pickup, how did the UI know what thing to pick up?

The button for drop, then, how did it know what thing to drop?

Specification mandates UI components

But the UI components for the various “commands” do not always make sense.

The player should be able to carry more than one thing in the inventory. For instance four keys are needed to open the chest and win the game.

But there should be a “command” for “dropping a thing in a room”. Which thing? Most students interpreted the specification as using a button for “Drop” - but a button can’t know what object you want to drop...

Specification mandates UI components, continued

The Things table is described to have “The thing in the room” - singularis.

Can a room only have one thing? If so, what happens if we want to drop a thing in a room which already has a thing? If this was an unfortunate description, and a room indeed can have more than one thing, it becomes very hard to implement a button for “Pick up thing” since the button can’t know which of many things you want to pick up.

Requirement - what’s problematic about this?

“The application should understand the following commands:

…

- Check if there are any thing**s** in the room
- *Print* the contents of the backpack (thing**s** we have picked up)

…

“

Command for listing things in the room

Why would we need a command to list things in a room in a *graphical* user interface? What would it mean? That we have to ask the room to reveal its things?

Couldn’t things in a room be visible by default? According to the Things table, only the Glass Key is invisible - couldn’t it just as well not be in the room until the rules make it “suddenly appear”?

Same thing about the inventory - why keep the inventory hidden until a button is press, again in a *graphical* user interface?

Requirement - what’s problematic about this?

“The application should understand the following commands:

…

- Pick up **the thing** which is in the room

…

“

Vagueness in command descriptions

One command is described as “put down a thing in the room”.

Does that mean:

- Only one thing can be put down
- One particular thing can be put down (somehow we should indicate what thing we want to put down)
- A random thing from the inventory can be put down
- Since a room might be able to hold only one thing, what happens when we try to put down another thing in the room?
- Et cetera

Requirement - what’s problematic about this?

“The application should understand the following commands:

…

- *Print* the room information **again**

…

“

Print room information again

- What is part of the room information? The “Lines” alone? The lines and information about a potential snake or dragon?
- Why should we have to ask the room for the information again? Shouldn’t we be made aware about any changes without having to press a button?
- If the dragon and snake are part of their respective room information, why aren’t they part of the lines for their rooms?
- What are the snake and dragon? They can’t be things, because they are not in the Things table either.
- “Print” - what does it mean? Send text to a laser printer?

Requirement - what's problematic about this?

“The application should understand the following commands:

…

- *Open* the pirate chest (requires the four keys)

…

Room 250 - if we have all four keys, we can *open* the chest and we have *won*.

“

Open the chest

The chest is interesting. It is is both a thing “Pirate Chest” and it is also mentioned in the Lines information about the last room (250):

“You are in an immense room. On a shelf of rock high above you there are marvelous treasures. Unfortunately, the wall is smooth and too slippery to climb. Before the wall there is a rather large wooden chest. It is locked with four huge and rusty locks.”

But since it is a Thing, should we be able to pick it up?

If we have the four keys we can “open” the chest and then we “win”.

Open, how? What’s inside? Skansholm’s book? How do we win?

Requirement - what’s problematic about this?

“Room 13, the Bird can only be picked up if we **already picked up** the Cage from Room 10. Additionally, the Bird *gets scared* and *flies away* if we **carry** the Rod from Room 11.

Room 19, here’s **a Snake** *blocking* the South exit. The Snake *disappears* if we *carry* the Cage **with** the Bird and drop them down - then the Southern exit *will be set to* 29.

Room 120. In the room is **a Dragon**. If we drop down Gold, Jewels, Diamonds and Silver, the Dragon will *disappear* and the West exit *will be set to* 0. Finally, a Glass Key *will appear*.”

Vagueness about the rules

Some things have special rules. The Bird is such a thing.

We need the Cage in order to be able to pick up the bird.

But if we have the Rod, “the bird gets scared and flies away”. Flies away where? For how long? Can we put down the Rod in the same room as the Bird without scaring it? What does it mean “to have the Rod”? In the hand, in the inventory?

When we get to the Snake, the snake disappears if we put down “The cage with the bird”. Have they merged? Is the Bird inside the Cage? How did it get there?

The Dragon

The Dragon, unlike the Bird, is not a Thing (since it is not present in the Things table). In order to get rid of the Dragon (whatever it is), we need to put down (actually Drop - is there a difference?) gold, jewels, diamonds and silver.

This speaks against the previous notion that a Room only can have one Thing.

Anyway, what happens to the Gold, Jewels, Diamonds and Silver once we put them down? Are they taken by the Dragon?

Note also, that if you opted for a Button for “Drop/Put down”, you have the problem of selecting what to put down.

Does the order in which we put them down matter?

The Snake and the Dragon

The Snake, like the Dragon, is not a Thing (not in the Things table) and not part of the room’s Lines of description. So both of them must be handled without any help of the database.

Both the Snake and the Dragon “disappear” if we put down certain Things.

What does disappear mean? For good? What happens to the Things when we put them down? Do they too disappear?

When the Dragon disappears, a Glass Key “appears”. The glass key is both invisible and unattainable according to the Things table.

Why can’t the Glass Key simply Appear (like they said) instead?

Requirement - what’s problematic about this?

“The application should understand the following commands:

…

- Go to the room *which is* to the North
- Go to the room *which is* to the South
- Go to the room *which is* to the East
- Go to the room *which is* to the West

…

“

Motion buttons

It is suggested that movement is implemented via buttons for the directions North, South, East and West.

- But if a Room lacks a passage to e.g. West, what should happen when we press the button?
- Why should we be able to press a button for a direction which is illegal?
- If we need to press a button to see if that direction is legal, how shall we discover that the South room gets valid after the Snake goes away? Press the button again?

It’s better to disable the buttons you cannot use.

Requirement - what's problematic about this?

Rules:

- Room 34, if we go North, **we** *will die* and the game *terminates*
- Room 113, if we go West, **we** *will die* and the game *terminates*

Game over situations

Some rooms have exits to death trap rooms which will kill the player (actually it says “we”) and terminate the game.

We should also have a button which terminates the game. But terminating a graphical program usually shuts it down and closes its window.

What does it mean that

- the player (or “we”) dies?
- the game terminates?

And who are “we”? ;-)

Requirement - what’s problematic about this?

From assignment 1 - the text version - we find this:

“Thing**s** - in some rooms you can find thing**s**. If there is no **thing**, it will *say* “No Objects”, otherwise it will *say* what **thing** is in the room.

The Room class has the following instance variables:

…

String **thing**;

“

Data type for Things

In the text version (assignment 1), the specification said that the Room class should have a String instance variable called `thing` . That too, implies that a Room only can have one thing (otherwise, having a String with all the Things would be a nightmare to deal with, if we want to pick up one of the things).

The description of things from the page before talks about Things (plural) and Thing (singular). Which is is? Can a room have **a thing** or **many** thing**s**?

Requirement - what’s problematic about this?

If any exit direction has the value 0, there is no Room in this direction. If there is a negative value, further rules should be applied.

Room 34 and 113 will kill “us” if we go in some directions.

```
select roomid, north, south, east, west from cave
 where north < 0 or south<0 or east<0 or west<0;
19|15|-2|74|30 /* Snake room */
34|-1|33|0|33
113|0|109|0|-1
120|69|0|74|-3 /* Dragon room */
```

Negative room ids

Rooms with an exit to ID 0 should be interpreted that this direction is invalid.

But some rooms have a negative ID for some directions. The room with the dragon is one such room. Its West exit is -3. If we manage to get rid of the Dragon, the Glass Key should appear and the West exit should be set to 0.

It is very hard to interpret what going West while the ID is -3 should mean. The DragonRules table isn't very helpful (and it's a challenge task to implement it).

If we **don't implement** the challenge task, what would going to room -3 mean?

Negative room ids, continued

Rooms with an exit to ID -1 have special rules in a table which could be implemented as a challenge. Otherwise, -1 means a death room.

The MinusOneRules table have messages which are hard to interpret, however, if we'd like to implement them as a challenge.

What does for instance the following message mean? "A wraith chased you out of the tunnel". What tunnel? Where did we go?

What should you do with specs like these?

- You must contact the customer and clear out the vaguenesses
- If the customer is a teacher, you must ask for clarification
- It is not possible to implement a stupid or contradictory specification

We will give you our suggestions for clearer rules in the next lecture

Discussion

- What did you think was hard to understand about the specification?
- What did you think was hard with the implementation?
  - Why did you think it was hard?
- What did you learn while interpreting the specification?
- What did you learn while implementing the game?
- What concepts from the course(s) did you find usable in the implementation of the game?
- What concepts did you think was missing from the courses, which if taught would have been usable for implementing the game?