Analysis [Iteration 1, Phase 2]

Slide Series 3

Analysis

During analysis we try to create a model of the problem domain as a collection of interacting objects (classes)

- This is the **analysis model** (aka domain model)

Have to find...

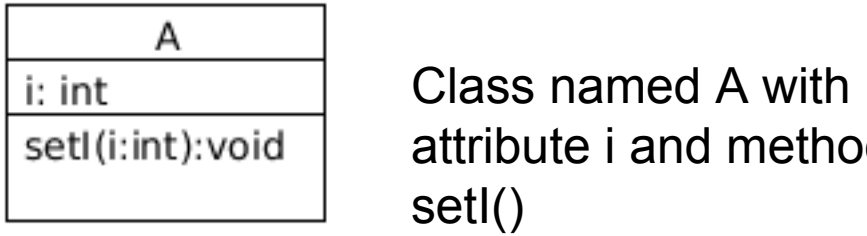
- Classes and how they are related
- To a lesser degree; possible attributes, possible behavior (methods) of the classes

Model represented as UML class diagram

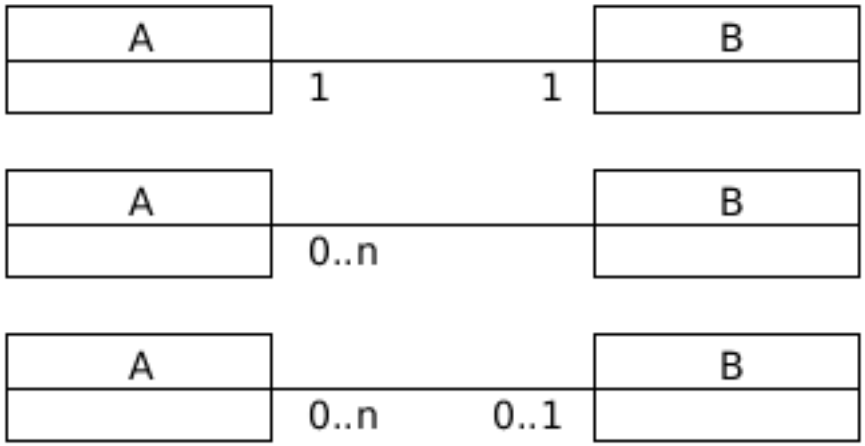
- A static view

Class Diagram for Analysis Model

Make it simple (no arrows, except possible specialization (extends))! Multiplicity is useful!



Class named A with attribute i and method setI()



Multiplicity

One instance of A is associated with one instance of B

Zero or many instances of A is associated with B

Zero or many instances of A is associated with zero or one B

Finding the Analysis Model

Have the UC's from RAD, simple method

- Underline nouns in use cases, will become be classes
- Underline verbs in use cases will become methods
- Find attributes/relationships from text (has, uses, is a, owns, knows, sends, receives, moves, rolls ...)
- **Include as much as possible**. Easy to skip later, ...

This is a critical activity

- If model wrong, not complete, inconsistent, ...
- ... **BIG** trouble later!!

This is a creative activity, few (no) rules ...

Automate a mess and you get an automated mess! // The 21 laws of programming

Remainder: Iterative Development

Getting a stable analysis model is a key issue

- Possible not stable after first iteration ...
- ... but if not stable after 2-3 ... problems!!!
- Don't assume you can fix it by coding, must solve the problem!

Work really hard to get model stable
(i.e. solve the problem)

Efficient Modelling

Optimal is to first draw on whiteboard!

- Very fast drawing and communication
- Use phone/camera to document
- Very fast communication, everyone can participate

Later, Tools to draw UML

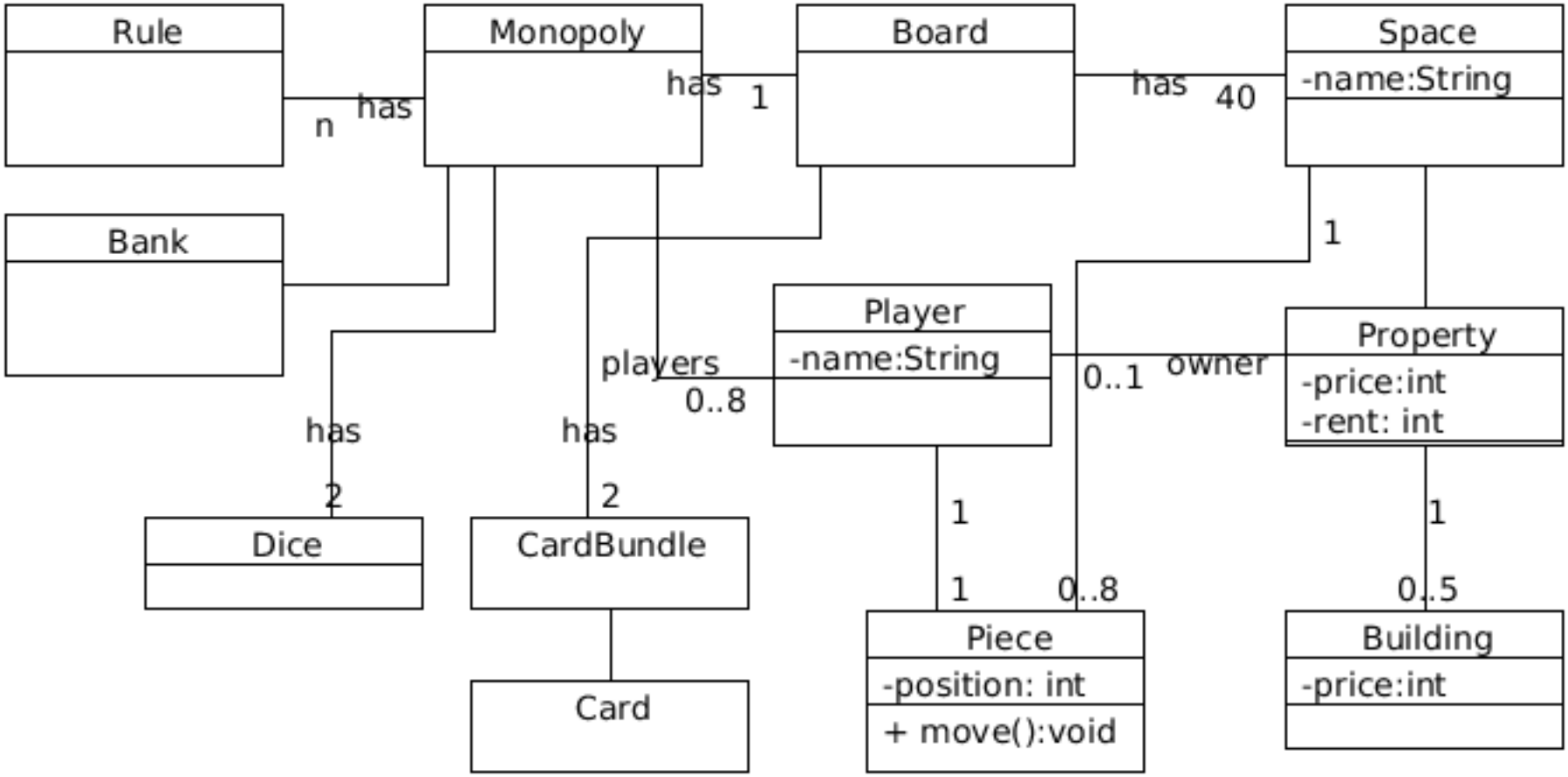
- When model getting more stable
- UMLet plugin to Eclipse, fastest possible (I use)
- Linux : Dia
- Mac/Win? ...

Picking Nouns for MP

From UC Move

- Dice
- Piece
- Board
- Space
- Jail
- Card
- Rent
- Player
- Balance

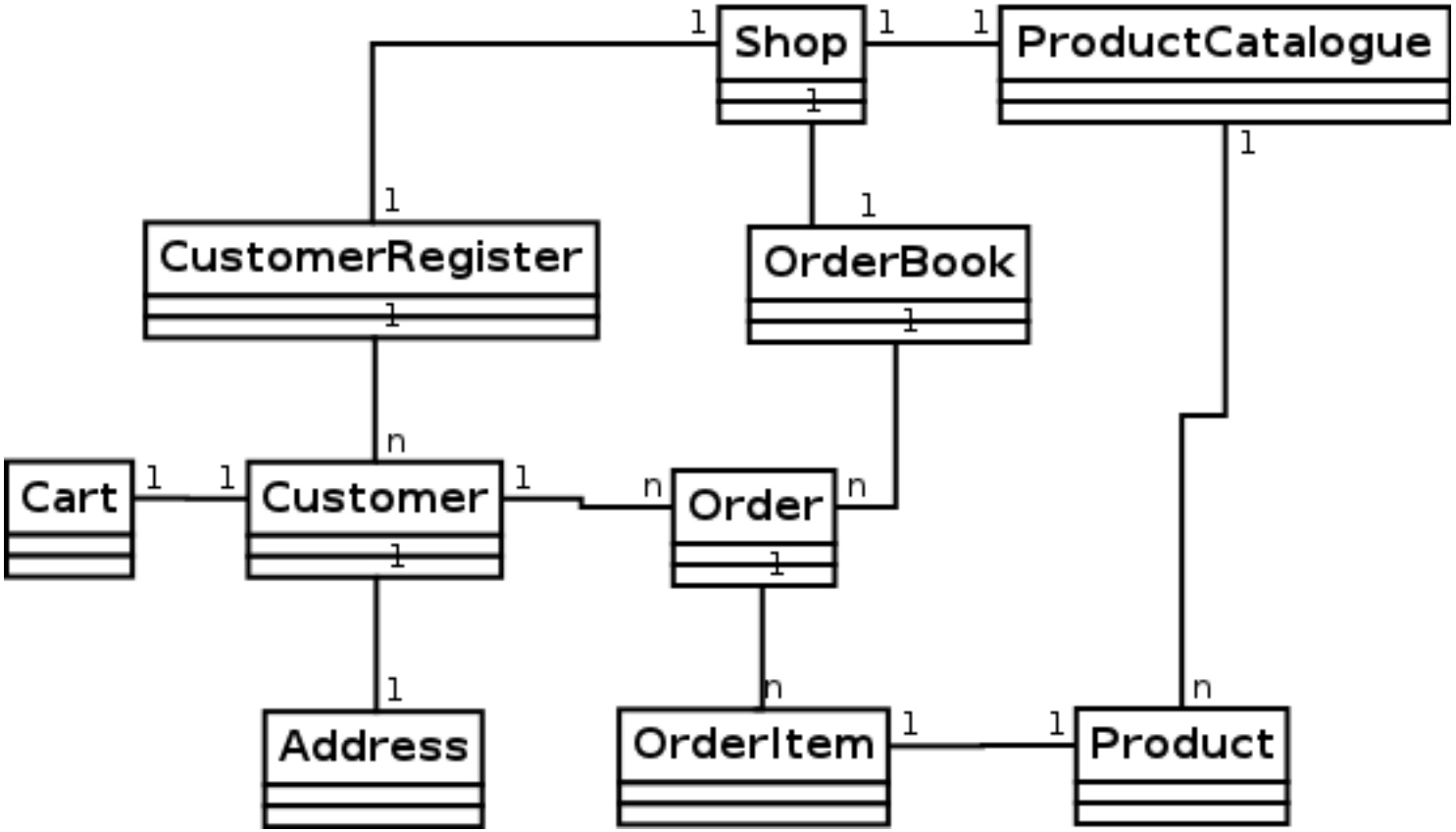
Preliminary Analysis Model for MP



This is not the ultimate truth, it's a starts of a solution...

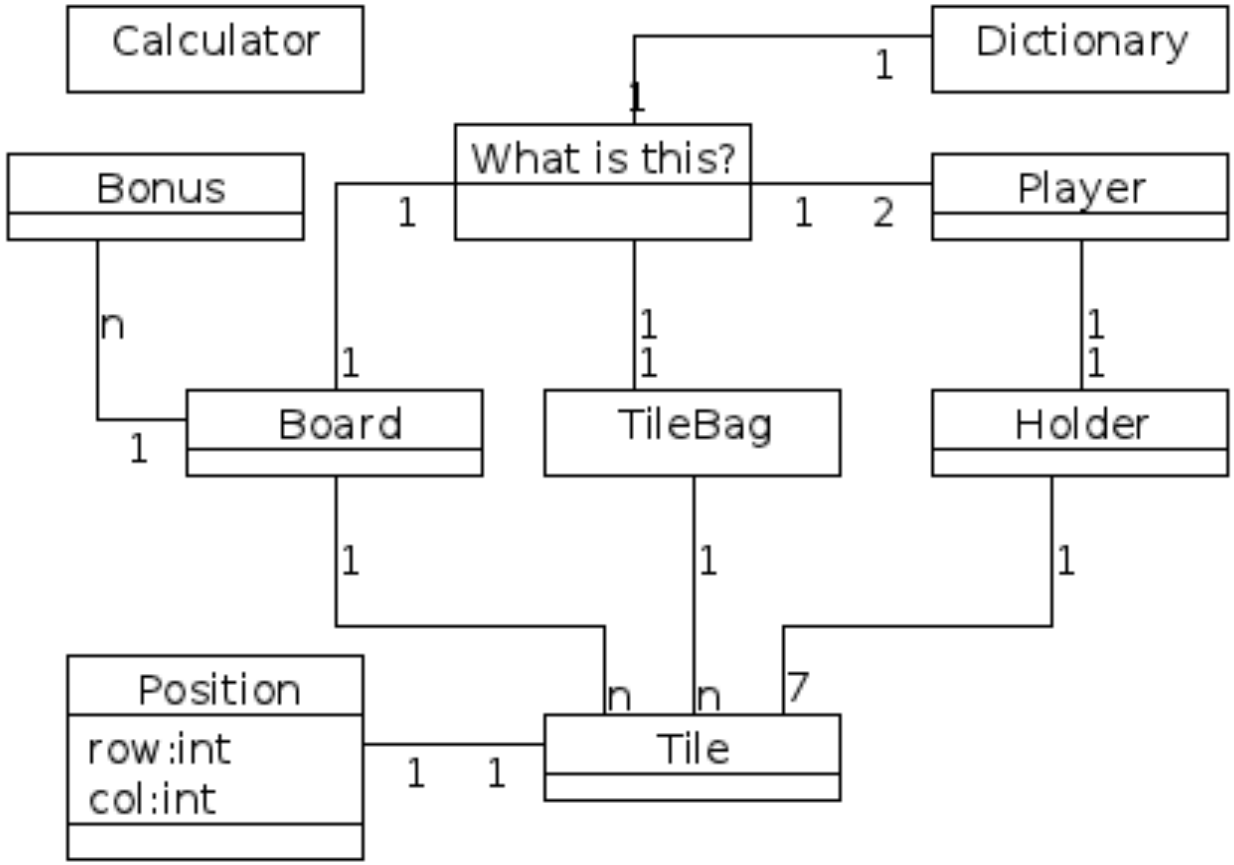
Other Example of Analysis Model

A Store

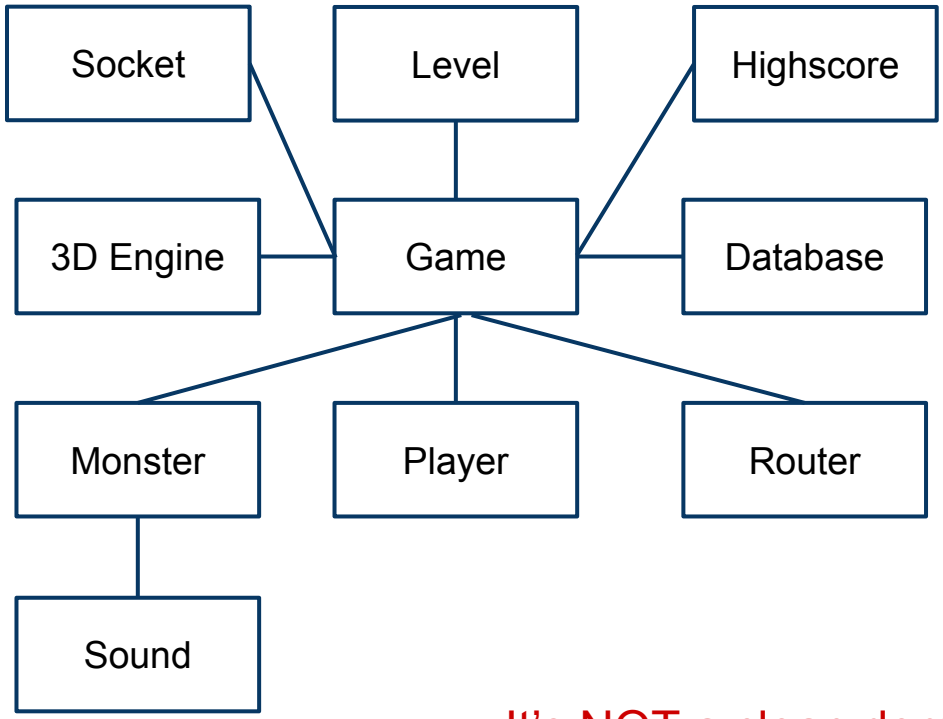


Yet Another Example of Analysis Model

A domain model of what?



How about this model?



It's NOT a clean domain model! Mixing services, techniques and model objects

Yet, Yet Another Example of Analysis Model

Let's assume a chat application. Some possible objects

- Inbox
- UserRegistry
- Chat
- TopicList
- Topic
- Message
- ChatRoom
- ...

How are these related?

Identity, Lifecycle and Absent Values

How do we identify objects?

- Do classes need to define identity (later equals() method)?

Will any objects survive the execution of the program?

- If so must be able to store objects
- Objects surviving = persistent objects
- Use stereotypes in class diagram to indicate <<Persistent>> (or similar) on persistent classes

Absent values: Value to indicate absence (similar to 0 in arithmetic)
- Avoid **null** as absent value

Identity and Life Cycle for MP

Player names must be unique!

Space names must be unique!

No objects will survive ... (for now)

Absent value: Player.NONE (alternative to null for missing owner)

More .. ?!?

Parallel Development

During this phase you can, in parallel, start implementing the GUI

- Have a mockup from RE
- Identify input/output elements from mockup
- Will change in response to the design and implementation of the model, so don't put too much effort
- Plan for techniques/tools to use

If using a different GUI-style (animated, 3d) you need to start "technical" prototyping right now!

- This is not covered in course

Technical Issues for GUI

Some Issues

- The GUI shouldn't be monolithic (i.e one huge JFrame)
- GUI is composed of many panels (custom components also good)
- Separate construction code (JButton b = new JButton()) and event code (listeners)
  - Use factories
- No model logic whatever in GUI
  - Don't use GUI for logical behaviour i.e. disable a button to prevent. Button disabled because of model state!
- GUI can use non-visible classes: Options, Preferences, etc. (holding "look"-data (non-model data, data not part of the domain problem))

GUI as Part of MVC

Some Issues

- EventListeners in GUI classes or...
- ..as separate classes outside (i.e. control classes in MVC)
- GUI will later be "observer" of model, technical solutions?
  - Recommended some kind of EventBus, upcoming...

GUI Tools

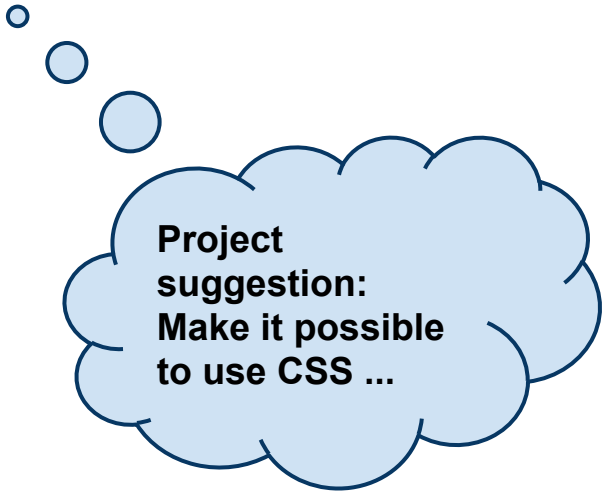
GUI-drawing tool often generate horrible code

- Possible many times as much code...
- Checkout before using in full scale (clean up generated code)

NetBeans have built in (unpopular?) GUI-builder

Using XML to define the GUI

- SwiXML
- BerylXML
- Links from course page
- Others...???



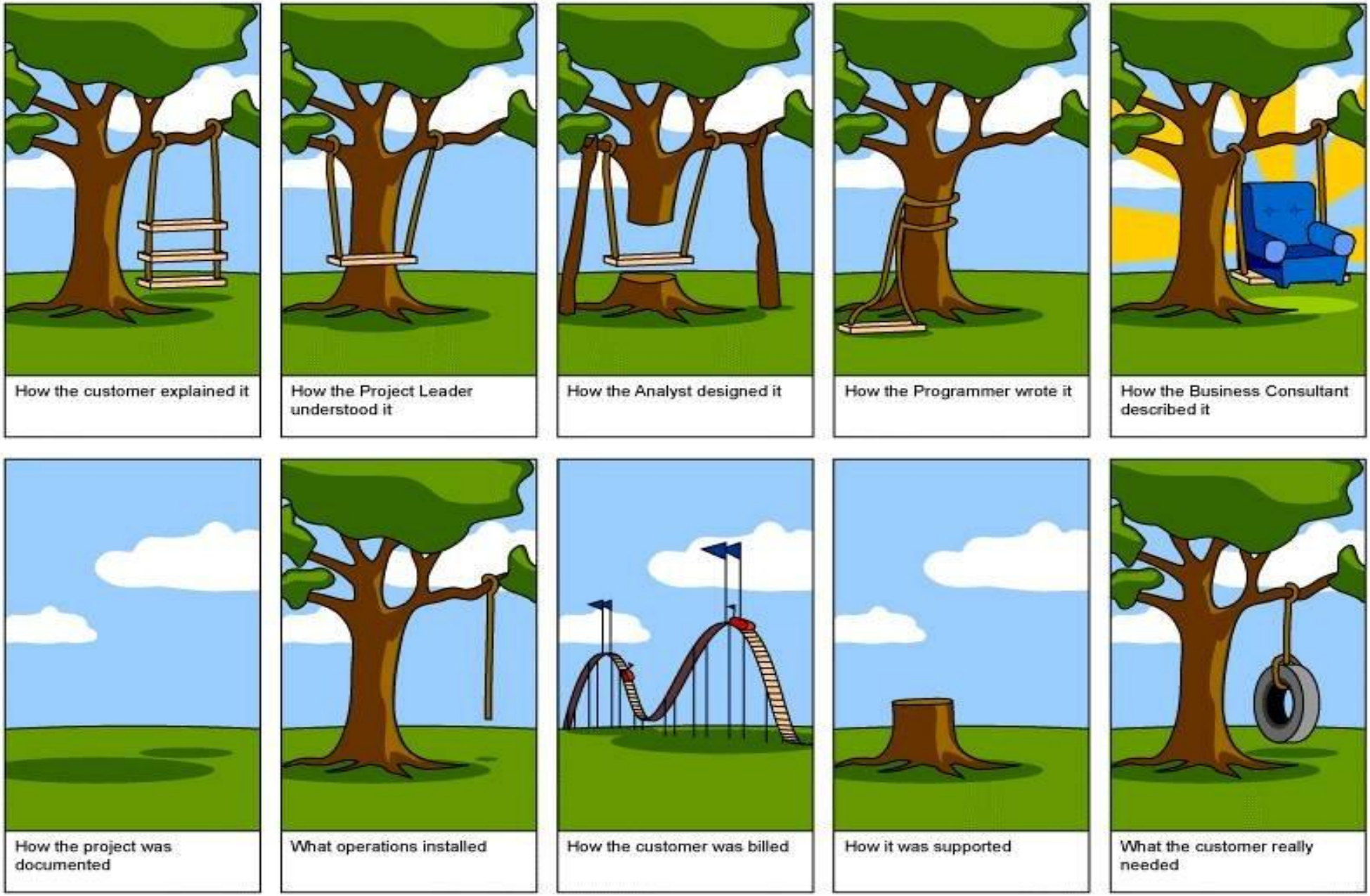
Documenting the Analysis

From previous phase we have recorded requirement elicitation in the RAD

Analysis is also documented in RAD

- Include the analysis model class diagram (possible updated later)
- List: Identity & Persistence
- Possible updated GUI

Hmmm...


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Summary

Analysis focus on building an analysis model

- We used the requirements from RAD (use cases) to find a model
  - Mostly classes, not much of attributes and methods
- We expressed the model as a simple UML-class diagram
- We documented model in RAD
- In parallel we develop a very basic GUI

Next: From analysis model to design model and first running increment