Model-based(ish) stuff

Laulasmaa 28.10.2013

TUT – Department of Pervasive Computing
Samuel Lahtinen

The stuff

- Commercial: Viking plop 2014

- History: Model-based Engineering of Command Interfaces
- (Prehistory: Embedded software Simulation in Workstation Environment)
- Current work: Service Composition for end-users (EasiClouds project)
- Bubbling under: Freeform software architecture design

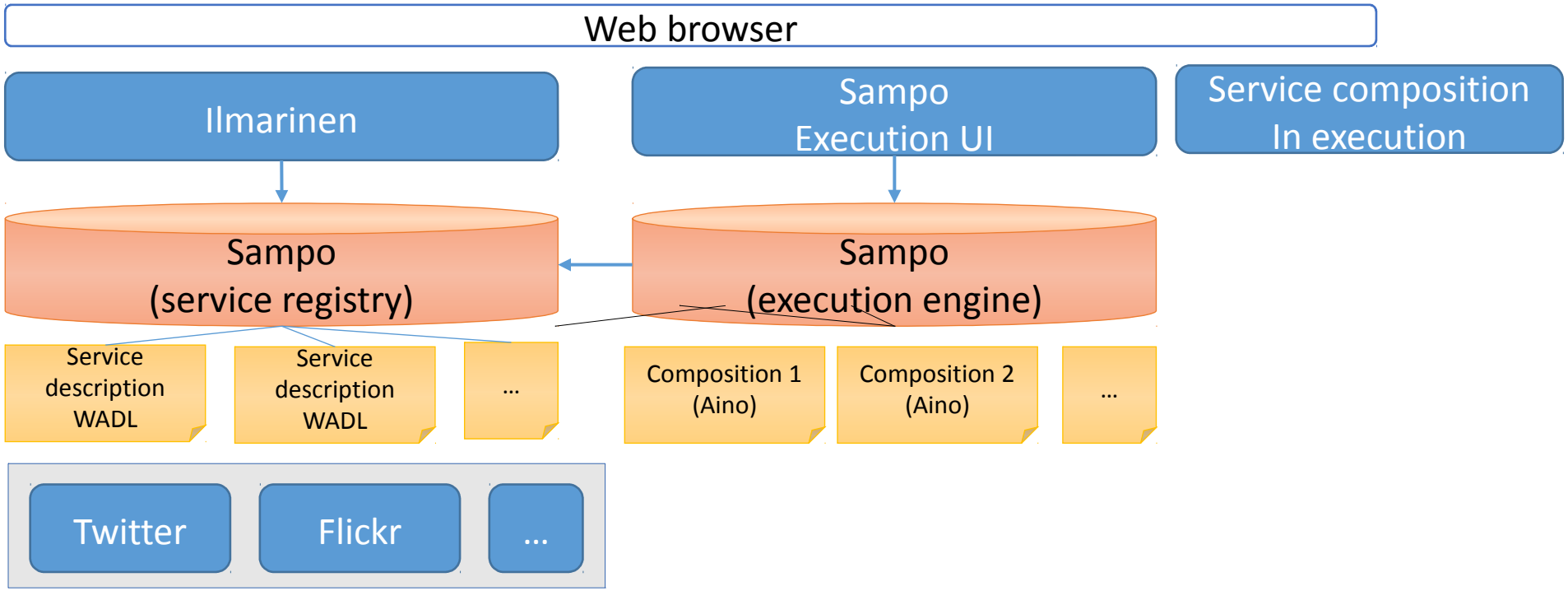
EasiClouds: Use case: photos from Twitter to Flickr

- A Twitter user has been sending many photos from his phone to Twitter
- She opens a Flickr account to keep photos organized.
- Manual transfer of photos between services tedious -> automation with a service composition
- Composition steps: service usage authorization, gets photos from Twitter, lets the user choose those what he wants, uploads selected photos to Flickr
- Task execution, run once or e.g. make the composition start weekly

Service composition for end-users

- Idea, allow end-users to combine web services to achieve a collaborative higher level task
- Research questions:
  - What should service descriptions contain to allow:
    - another service / application to use the service (automatically) only based on the description
    - a human user to understand the idea of the interface (how to abstract the info)
    - services to be connected to each other (collaboration)
  - How to describe services and the resources so that we can generate a user friendly way to use/combine the services

Architecture



Model-based derivation of command interfaces

- A command interface (speech interface) is generated from a model
- The model can be used to control active commands, verify them etc.
- Target application is attached to the control application by implementing a control component
- A model is used to describe the target application: different states, features and operations etc.
- Speech control language is automatically derived from the model, model is used to verify the commands, the application is controlled through the control interface

Model-based Command Interfaces, principles

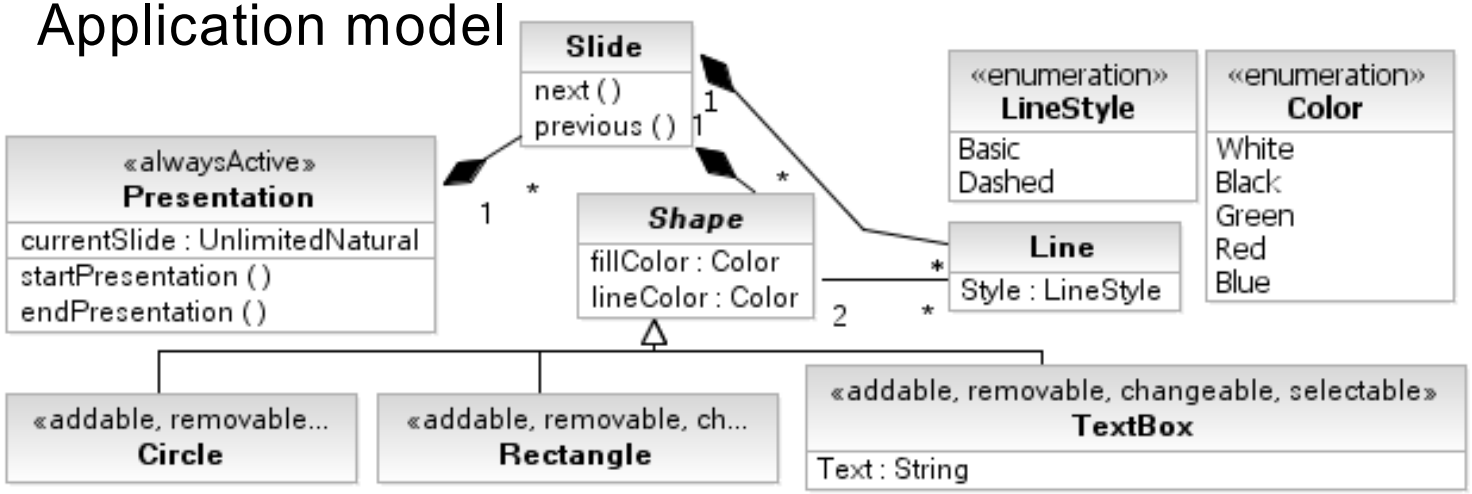
- We try to view applications functionality as creating and manipulating objects, changing their properties, and activating operations related to them.
- An essential state of an application is presented as a collection of objects and their relations
- The application allows selecting / activating (of these objects)
- A command can only be given if the set of active objects (*active context*) is correct

Terminology:

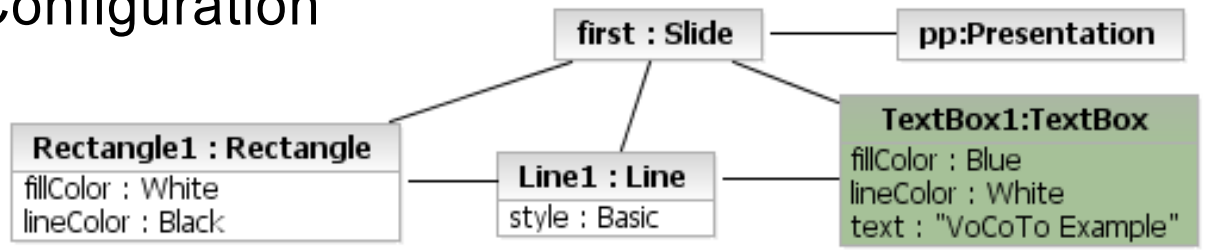
- *State configuration* : runtime presentation of an application: objects, links, attributes and their values.
- *Application models* : Describe the valid state configurations, state configuration is an instance of an application model.
- *Valid context* : A ”correct” set of objects for a command.

Models in applications

Application model

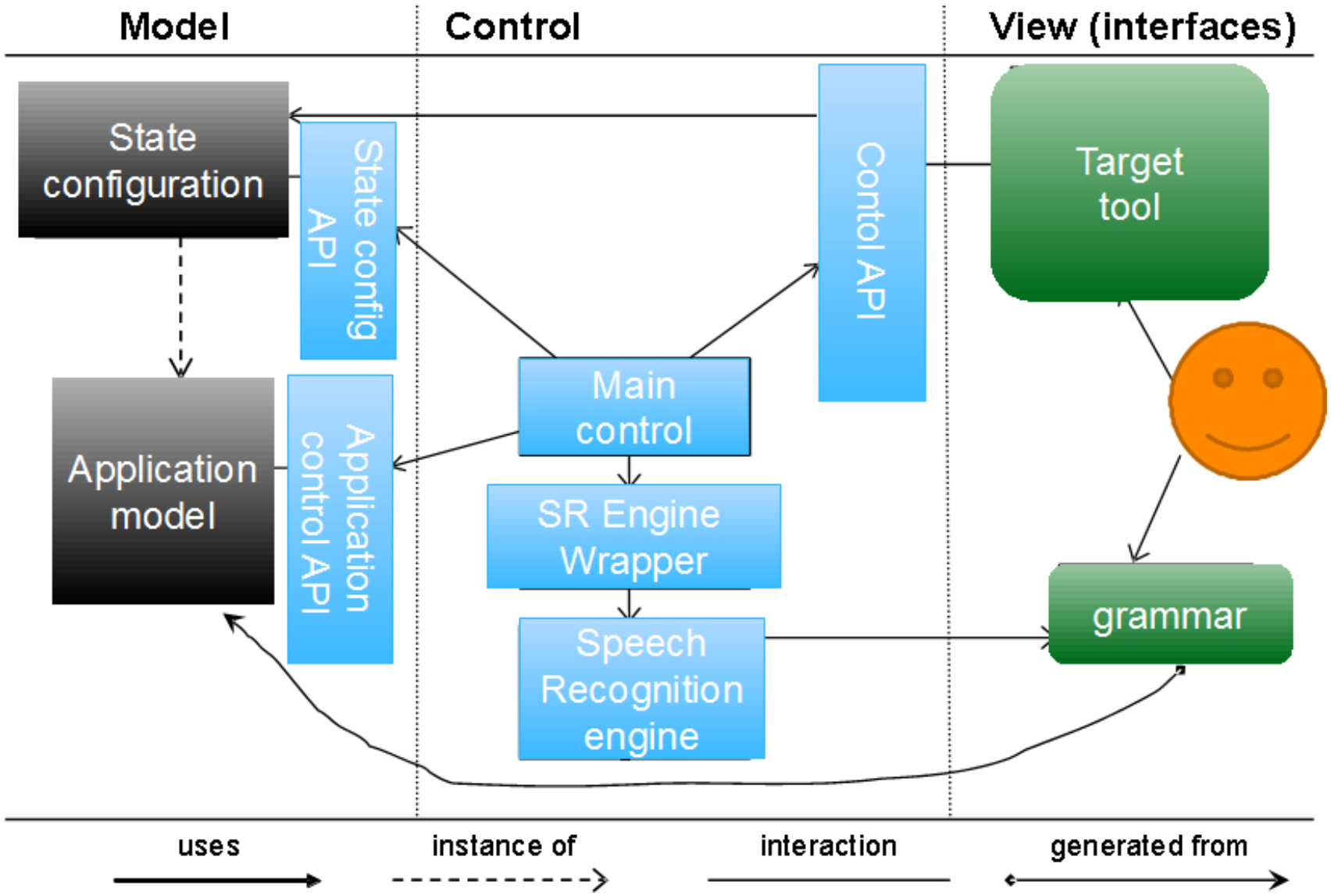


State Configuration



PowerPoint

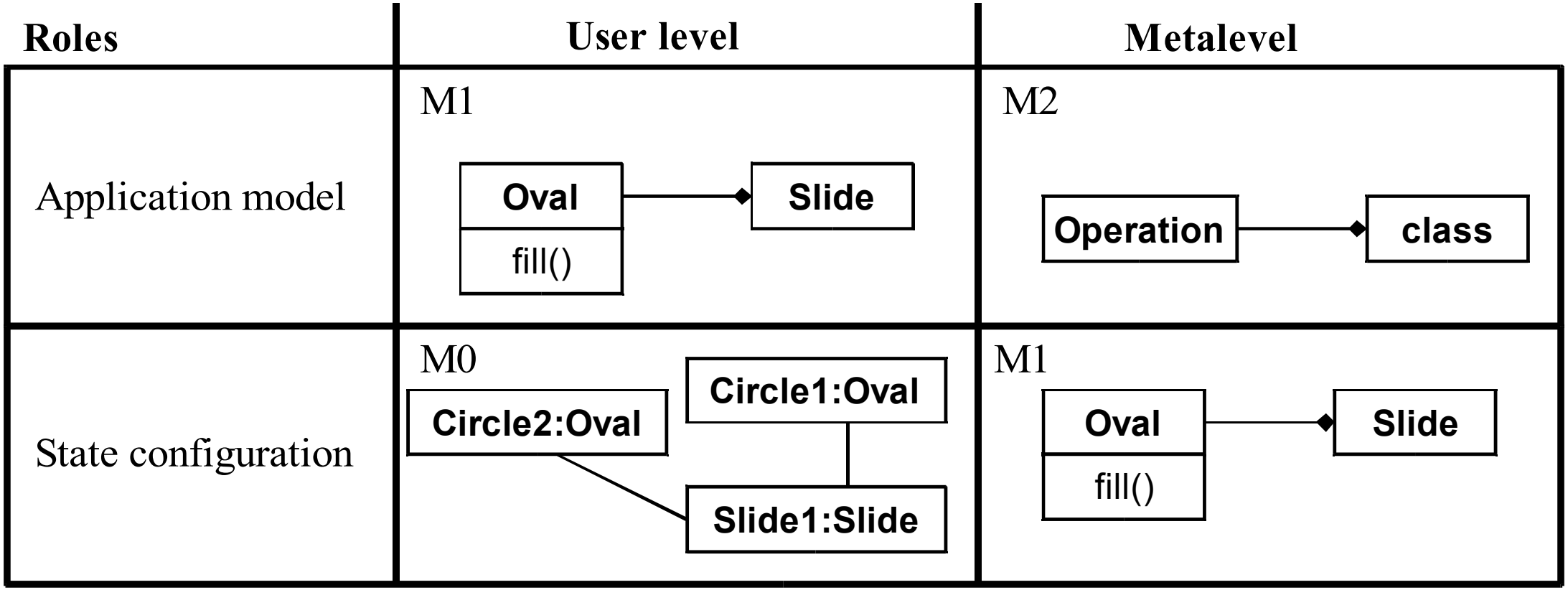
Architecture



Reflective interfaces

- We can define rules on how to modify the models with a metamodel → we get rules on how to change the grammar/language that is in use
- A metamodel can be used to derive the commands for changing the model
- →reflective commands, using them change the language in use

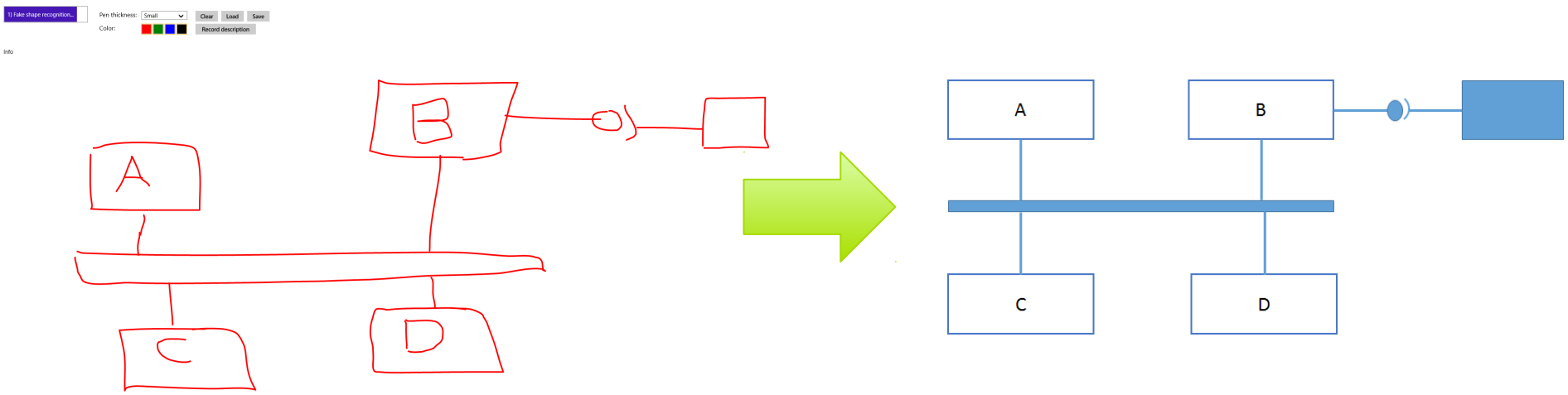
Different levels of models



Freeform Software Design

- Software design != using a modelling tool
- Sketching, collaborating, exchanging ideas…
- Design decisions are easily lost during the process
  - why, what, how?
- Updating early designs, concretizing them requires drawing them from the beginning
- Modelling tools terrible for beginners

Idea...



- Draw, design, explain
- (Recognize shapes, speech…)
- Concretize
- Playback, import to…

Questions, discussion etc.