AARHUS UNIVERSITET

Software Architecture in Practice

Architectural Prototyping

Henrik Bærbak Christensen

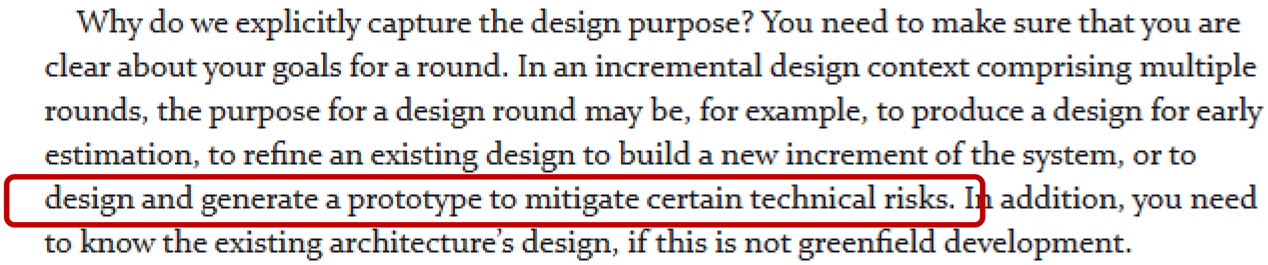
AARHUS UNIVERSITET

Patterns and Tactics

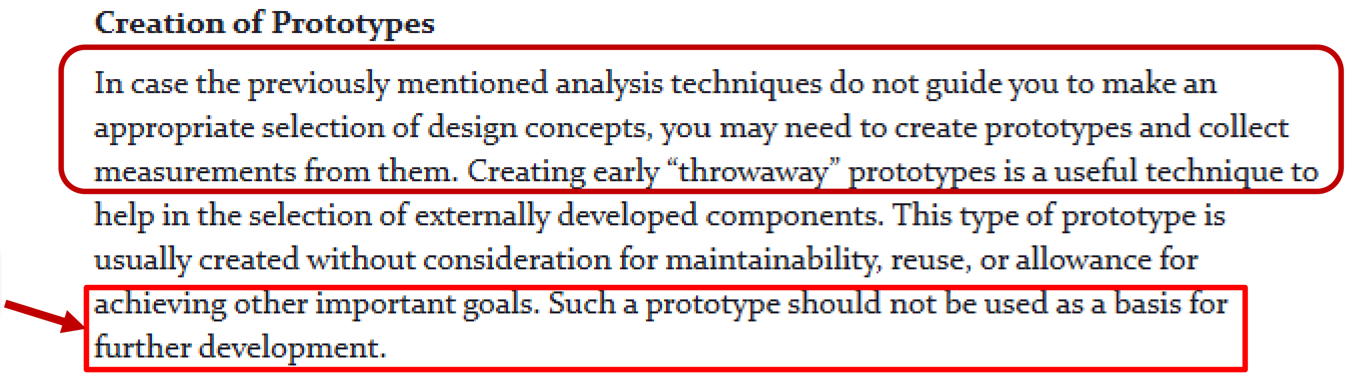
- All well, but…
- … what do we do when...
  - we are *uncertain* whether one or the other style / pattern / tactic is the better to choose?
  - when we are *uncertain* whether the favorite architecture will have the right balance of conflicting quality attributes?
  - when we are *uncertain* that the specs of the third party vendor are *real* and not just empty sales talk?
  - when we want to *explore* the design space for learning – and becoming better architects?

Bass et al. §20.2

- Prototyping has included into Bass over the years ☺



Why do we explicitly capture the design purpose? You need to make sure that you are clear about your goals for a round. In an incremental design context comprising multiple rounds, the purpose for a design round may be, for example, to produce a design for early estimation, to refine an existing design to build a new increment of the system, or to design and generate a prototype to mitigate certain technical risks. In addition, you need to know the existing architecture's design, if this is not greenfield development.



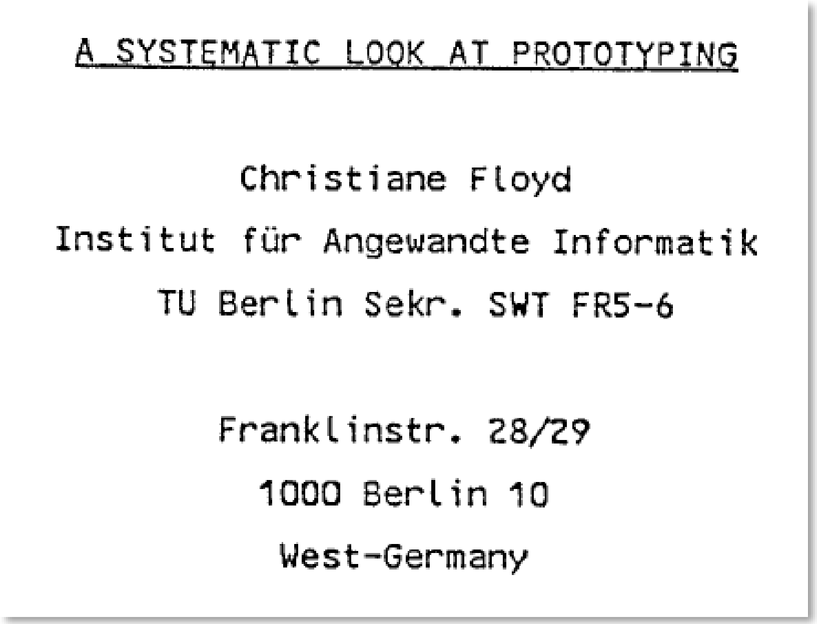
Creation of Prototypes

In case the previously mentioned analysis techniques do not guide you to make an appropriate selection of design concepts, you may need to create prototypes and collect measurements from them. Creating early "throwaway" prototypes is a useful technique to help in the selection of externally developed components. This type of prototype is usually created without consideration for maintainability, reuse, or allowance for achieving other important goals. Such a prototype should not be used as a basis for further development.

We do not totally agree with Bass on that note…

Prototyping

- Seminal paper by Floyd, 1984.
  - *executable systems that “involve an early practical demonstration of relevant parts of the desired software”.*
  - *“a learning vehicle providing more precise ideas about what the target system should be like.”*
  - *“the discussion focuses on software intended as direct support for human work.”*



A SYSTEMATIC LOOK AT PROTOTYPING

Christiane Floyd

Institut für Angewandte Informatik

TU Berlin Sekr. SWT FR5-6

Franklinstr. 28/29

1000 Berlin 10

West-Germany

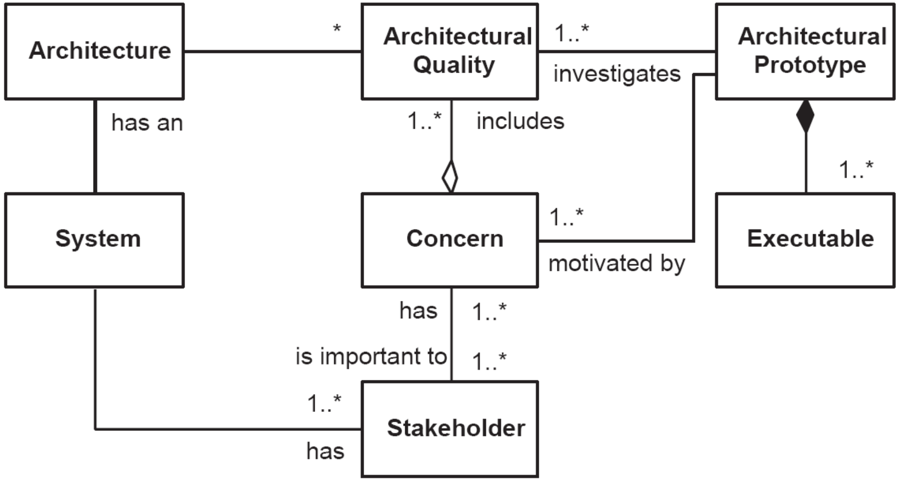
Prototyping

- Prototyping has mainly been used in the user interface community
  - *“the discussion focuses on software intended as direct support for human work.”*

- But why is *building executable systems that involve an early practical demonstration of relevant parts of the desired software* only interesting for HCI folks?
  - **Why not for software architects?**

So – What is it?

- **Definition:**
- An *architectural prototype* consists of a set of executables created to investigate architectural qualities related to concerns raised by stakeholders of a system under development. *Architectural prototyping* is the process of designing, building, and evaluating architectural prototypes.



- **Bardram, Christensen, Hansen: WICSA 2004**

Characteristics

- architectural prototypes are constructed for ***exploration and learning*** of the ***architectural design space.***
- architectural prototyping addresses issues regarding architectural ***quality attributes*** in the target system.
- architectural prototypes ***do not provide functionality*** per se.
- architectural prototypes typically address architectural ***risks***.
- architectural prototypes address the problem of ***knowledge transfer*** and ***architectural conformance***.

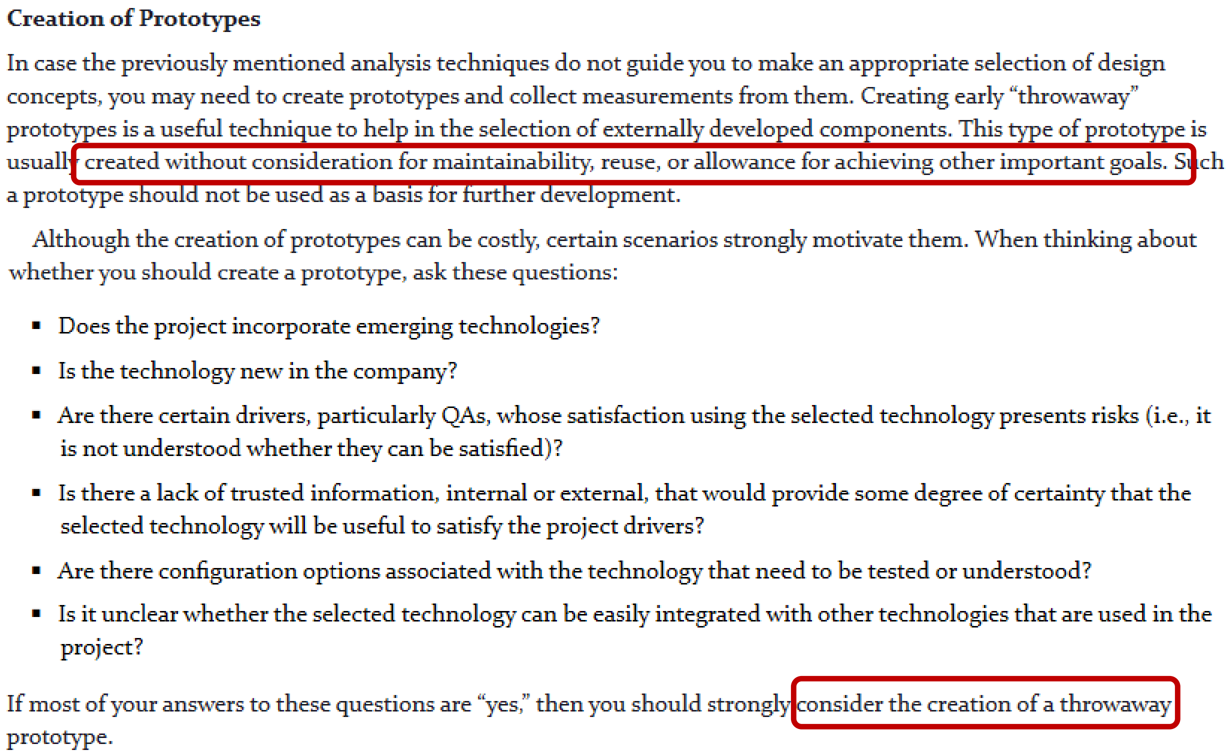
No functionality

- architectural prototypes ***do not provide functionality*** per se.

- I.e. no *business logic* and *no user interface*
- Example: Performance
  - Pump data from one end to the other fast enough
  - Simulate “business algorithms” by delays
- Example: Availability
  - Measure ‘reintroduction’ time but no real functionality

In Contrast to Bass

- We disagree in certain respects with Bass et al.’s idea of prototypes
  - Early APs may form *walking skeletons*
  - High quality on the QAs they set out to measure



Creation of Prototypes

In case the previously mentioned analysis techniques do not guide you to make an appropriate selection of design concepts, you may need to create prototypes and collect measurements from them. Creating early “throwaway” prototypes is a useful technique to help in the selection of externally developed components. This type of prototype is usually created without consideration for maintainability, reuse, or allowance for achieving other important goals. Such a prototype should not be used as a basis for further development.

Although the creation of prototypes can be costly, certain scenarios strongly motivate them. When thinking about whether you should create a prototype, ask these questions:

- Does the project incorporate emerging technologies?
- Is the technology new in the company?
- Are there certain drivers, particularly QAs, whose satisfaction using the selected technology presents risks (i.e., it is not understood whether they can be satisfied)?
- Is there a lack of trusted information, internal or external, that would provide some degree of certainty that the selected technology will be useful to satisfy the project drivers?
- Are there configuration options associated with the technology that need to be tested or understood?
- Is it unclear whether the selected technology can be easily integrated with other technologies that are used in the project?

If most of your answers to these questions are “yes,” then you should strongly consider the creation of a throwaway prototype.

Classes of APs

- Explorative
  - *Constructed in order to explore design space*
- Experimental
  - *Constructed in order to evaluate specific architectural decisions, typically trade-offs between qualities*
- Evolutionary
  - *Constructed one after another to produce a skeletal system*

Examples

Example 1: Net4Care AP06

- Net4Care: Support telemedicin for SMBs
  - *Questions:*
    - *Is Forwarder/Receiver a solid communication pattern?*
    - *Is KronikerDataSet a solid foundation as storage format?*
    - *Is BaseX a solid database system?*
    - *Is a compositional Observation flexible enough?*
    - *Is a manual serialization method easy enough?*
- AP06:
  - End-to-end testing (This was the precursor for TeleMed ☺)
  - No GUI, no error handling, ...
- Total staff hours: 23.5 hours

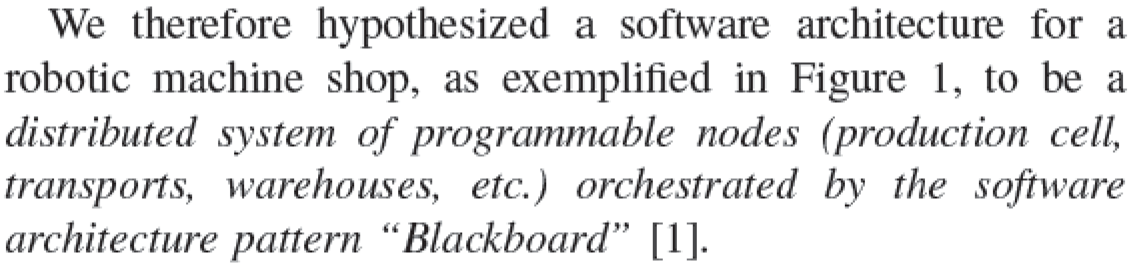
Example 1: Net4Care AP06

- Net4Care: Support telemedicin for SMBs

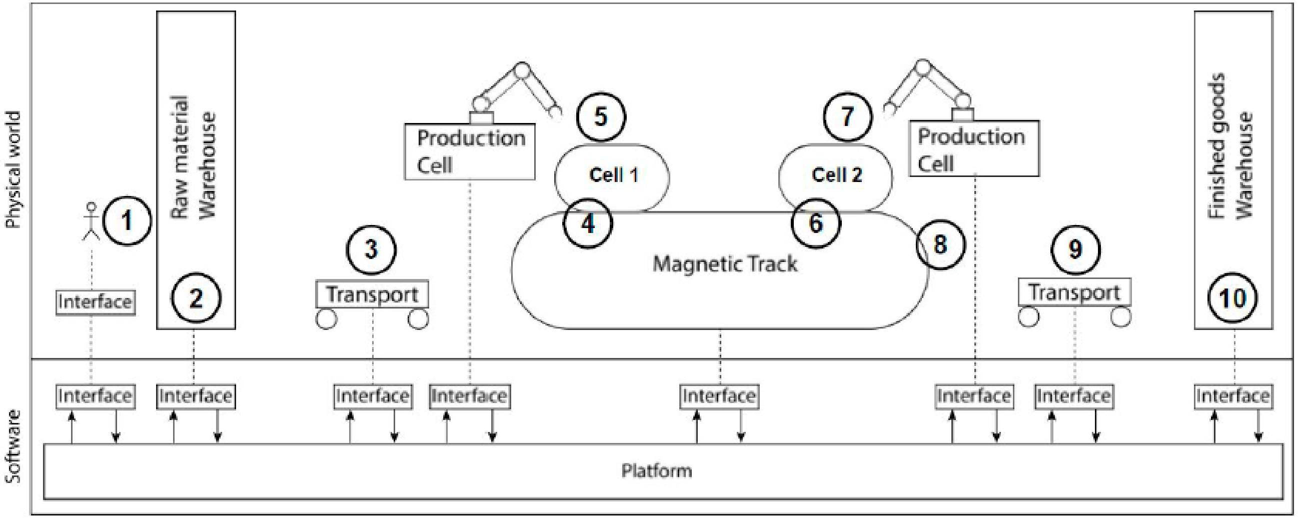
- The output
  - *Questions:*
    - *Is Forwarder/Receiver a solid communication pattern?* ☺
    - *Is KronikerDataSet a solid foundation as storage format?* ⚡
    - *Is BaseX a solid database system?* ⚡
    - *Is a compositional Observation flexible enough?* ☺
    - *Is a manual serialization method easy enough?* ⚡

Example 2: SMB Industry 4.0

- Goal: Flexible Robotic Manufacturing for SMBs
- Hypothesis:

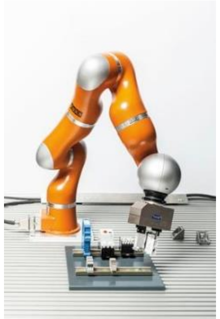


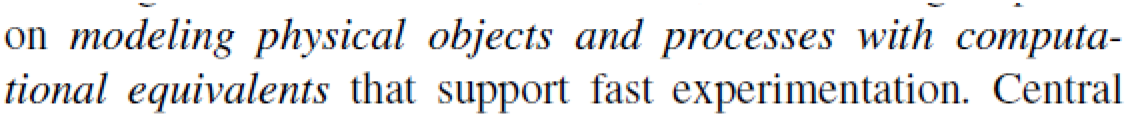
We therefore hypothesized a software architecture for a robotic machine shop, as exemplified in Figure 1, to be a *distributed system of programmable nodes (production cell, transports, warehouses, etc.) orchestrated by the software architecture pattern "Blackboard"* [1].



Example 2: SMB Industry 4.0

- Goal: *Explore the design space of*
  - *Is the blackboard architecture well suited?*
  - *And what tactics/external components to use?*
- *No functionality per se…*



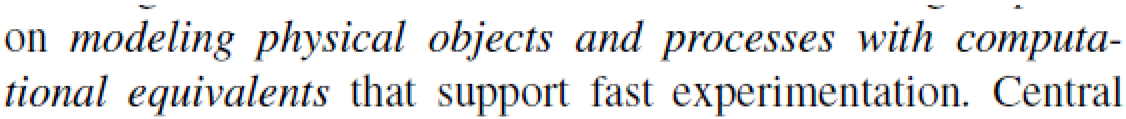


on *modeling physical objects and processes with computational equivalents* that support fast experimentation. Central

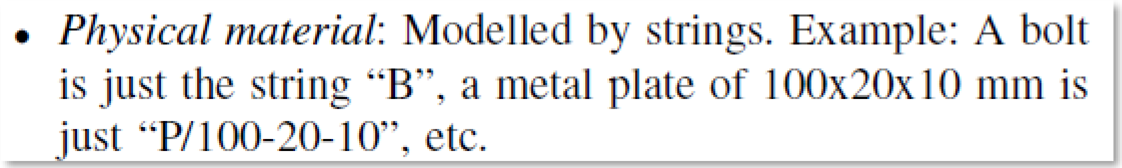
- *Frederik Brooks analogy*
  - *Essential and accidental complexity*

Example 2: SMB Industry 4.0

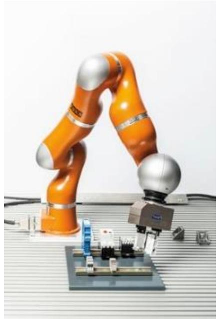
- So…

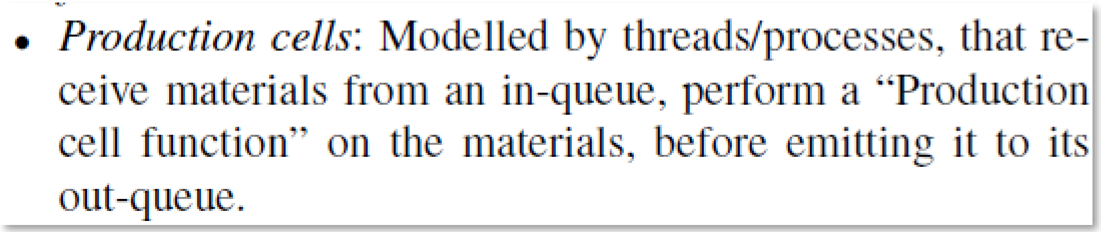


on *modeling physical objects and processes with computational equivalents* that support fast experimentation. Central

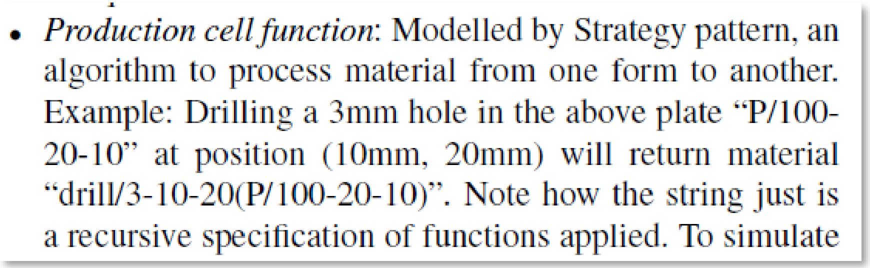


- *Physical material*: Modelled by strings. Example: A bolt is just the string “B”, a metal plate of 100x20x10 mm is just “P/100-20-10”, etc.





- *Production cells*: Modelled by threads/processes, that receive materials from an in-queue, perform a “Production cell function” on the materials, before emitting it to its out-queue.

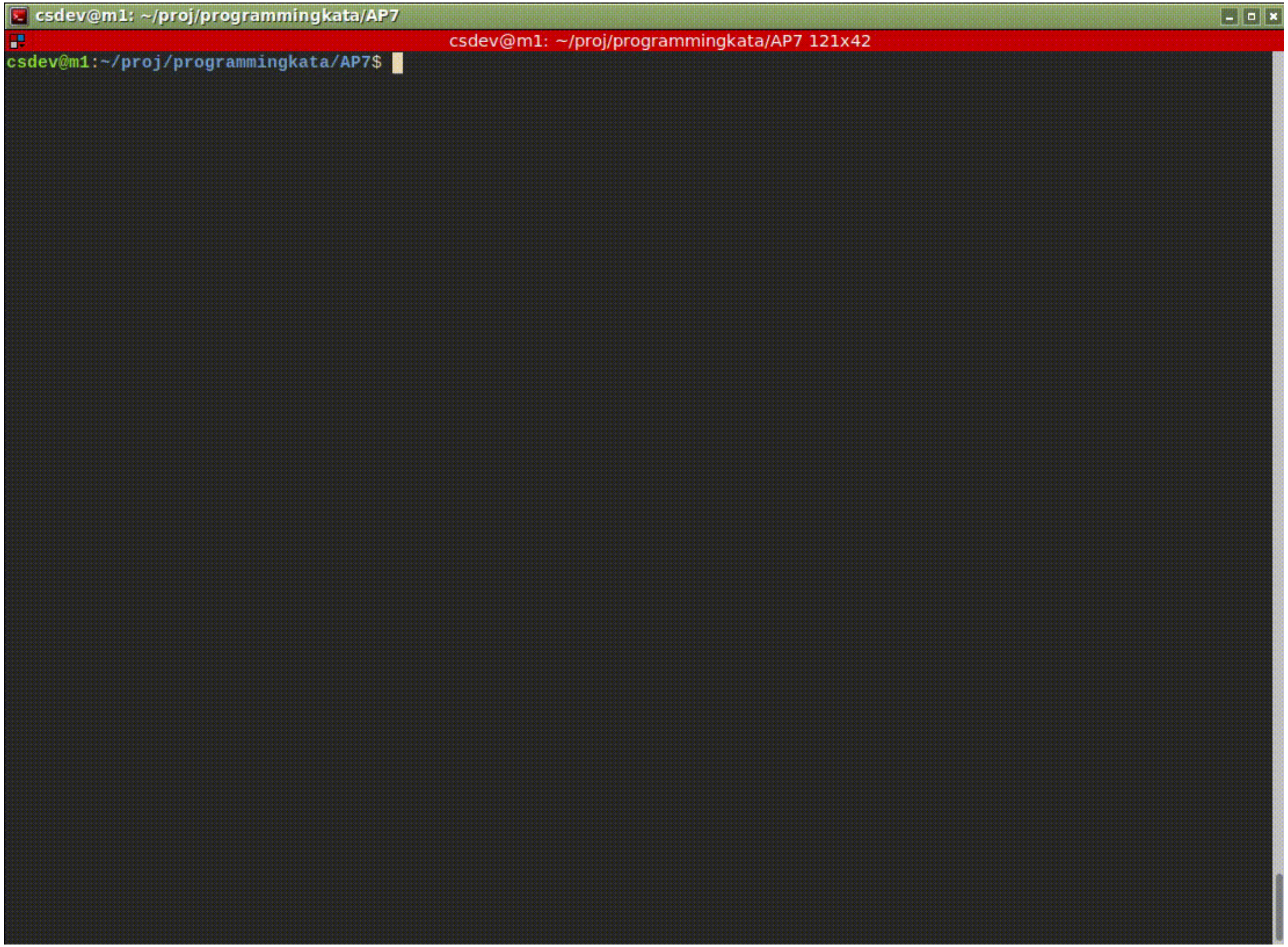


- *Production cell function*: Modelled by Strategy pattern, an algorithm to process material from one form to another. Example: Drilling a 3mm hole in the above plate “P/100-20-10” at position (10mm, 20mm) will return material “drill/3-10-20(P/100-20-10)”. Note how the string just is a recursive specification of functions applied. To simulate

Etc…

Example 2: SMB Industry 4.0

- Floor is then simulated by simple log output…

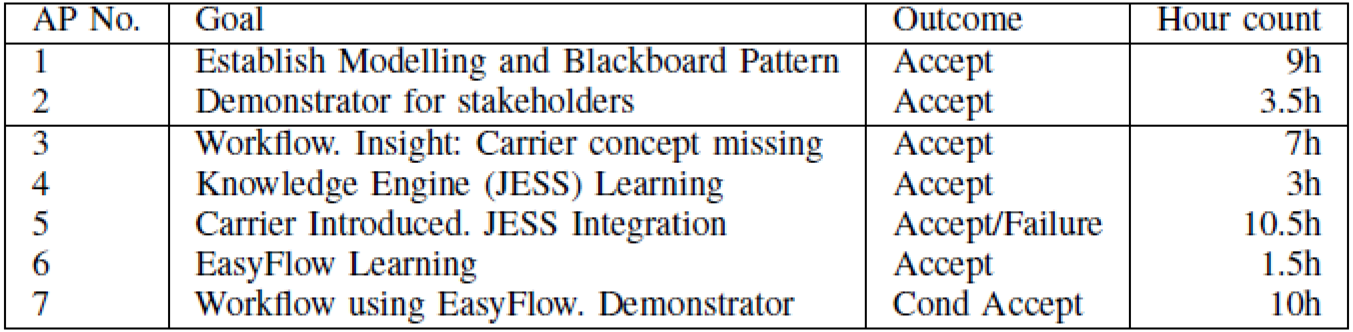


csdev@m1: ~/proj/programmingkata/AP7
csdev@m1: ~/proj/programmingkata/AP7 121x42

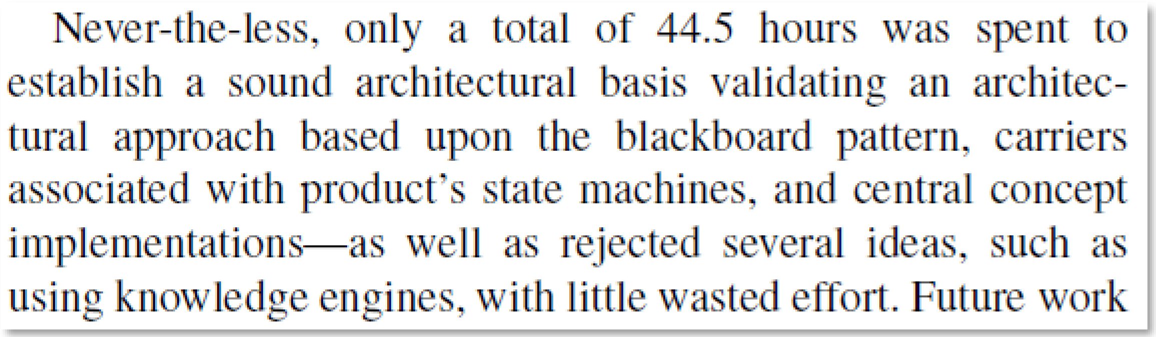
```
csdev@m1:~/proj/programmingkata/AP7$ 
```

Example 2: SMB Industry 4.0

- Learnings
  - 44½ h



AP No.	Goal	Outcome	Hour count
1	Establish Modelling and Blackboard Pattern	Accept	9h
2	Demonstrator for stakeholders	Accept	3.5h
3	Workflow. Insight: Carrier concept missing	Accept	7h
4	Knowledge Engine (JESS) Learning	Accept	3h
5	Carrier Introduced. JESS Integration	Accept/Failure	10.5h
6	EasyFlow Learning	Accept	1.5h
7	Workflow using EasyFlow. Demonstrator	Cond Accept	10h



Never-the-less, only a total of 44.5 hours was spent to establish a sound architectural basis validating an architectural approach based upon the blackboard pattern, carriers associated with product's state machines, and central concept implementations—as well as rejected several ideas, such as using knowledge engines, with little wasted effort. Future work

AARHUS UNIVERSITET

Anatomy of an AP

So – how do we code them?

Keypoints

- Some key goals dictate how to code them
  - Potentially *evolutionary prototype* which will transition into the real system: **produce high quality code!!!**
- That is:
  - An AP is normally produced when you *think* you have the proper architectural tactic
    - i.e. we are pretty confident
  - **Thus the code will likely transition into production!**
- Comparison:
  - User interface prototypes are usually discarded!!! (Ha ha ???)

Keypoints

- Some key goals dictate how to code them
  - Answers an architectural question (hypothesis) and nothing else: **include only enough code to answer the question!**
- That is
  - *Keep focus!* Keep this TDD principle in mind
- General rule
  - No real DB. No real business logic. No user interface. No real domain objects.
    - But the *real interfaces* that Façade these core roles
  - Or rather – include only ‘real’ implementation if required to test the architectural hypothesis *Does this tactic work?*

To do that, you need to…

- How to keep focus?
  - **Fake it till you make it**. *Program to an interface.* Any responsibility that your architecture may interact with but that is not architectural, should be accessed via interface with stub/fake implementation.

- That is:
  - *I need to store X in the database (db access not arch. issue)*
    - *storage.store(x)*
    - *Storage storage = new FakeObjectStorage();*

Keypoints

- Some key goals dictate how to code them
  - The found answer must be applicable to the target system: **the code context must be ”true” to the target system**
- That is:
  - Use the technology, platform, language, …, of the target system
    - Especially for run-time QAs, like performance, …
  - Do not invent interfaces that will not hold in the target
    - storage.store(x)   if the domain demand another interface
- There may be exceptions e.g. for exploratory AP
  - We explored CAN-bus issues on embedded systems – in a Java prototype…

Keypoints

- Some key goals dictate how to code them
  - The answer must be found quickly: **the development process must be fast, code should be minimized**

- That is
  - Again, keep focus.
    - Do not spend ages on the DB interaction if it is not the architectural issue studied…
  - Use best-of-breed tooling
    - Fast development environments, package management and dependency systems, reuse common libraries, etc.

Key techniques

- Techniques/tactics to achieve this are
  - Modifiability tactics
    - **Increase semantic coherence + Defer binding**
  - Testability tactics
    - **Specialized Interfaces, Localize .., Abstract …, Sandbox**
- More programming near terms…
  - Flexible design (Gamma et al. §1.6/Christensen §16)
    - **Encapsulate what varies, program to interface, favor object composition – SOLID principles**
  - Test doubles (Meszaros (online)/Christensen § 12)
    - **Test stubs, fake objects**
  - **Fake it ‘till you make it** (Beck / Christensen §5)

AARHUS UNIVERSITET

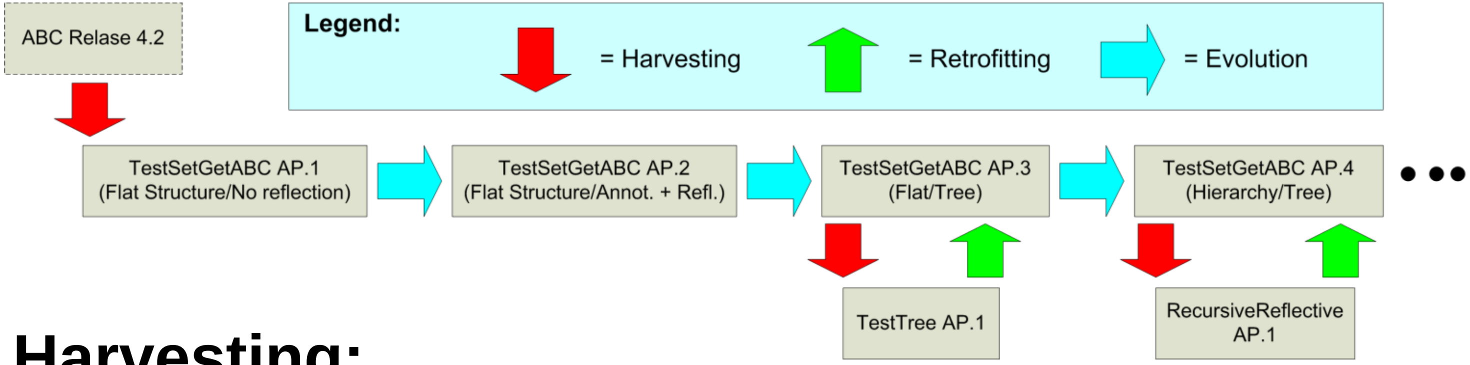
Process

For Architectural Experiments with *Existing systems*

Experiments with Legacy

- Legacy systems usually have *low affordance* for experimentation with architectural issues
  - Defined as *any minor change to the issue studied leads to a ripple effect of major changes to unrelated issues*
  - Experimental turn-around time is way too high
    - Hypotesis – Experiment (takes ages!) – Give up ☹
- In comes *architectural harvesting*

Architectural process


ABC Relase 4.2
Legend:
= Harvesting
= Retrofitting
= Evolution
TestSetGetABC AP.1 (Flat Structure/No reflection)
TestSetGetABC AP.2 (Flat Structure/Annot. + Refl.)
TestSetGetABC AP.3 (Flat/Tree)
TestSetGetABC AP.4 (Hierarchy/Tree)
• • •
TestTree AP.1
RecursiveReflective AP.1

Harvesting:

- *The process of extracting architecturally significant code/behavior or knowledge from a super-AP to a sub-AP*

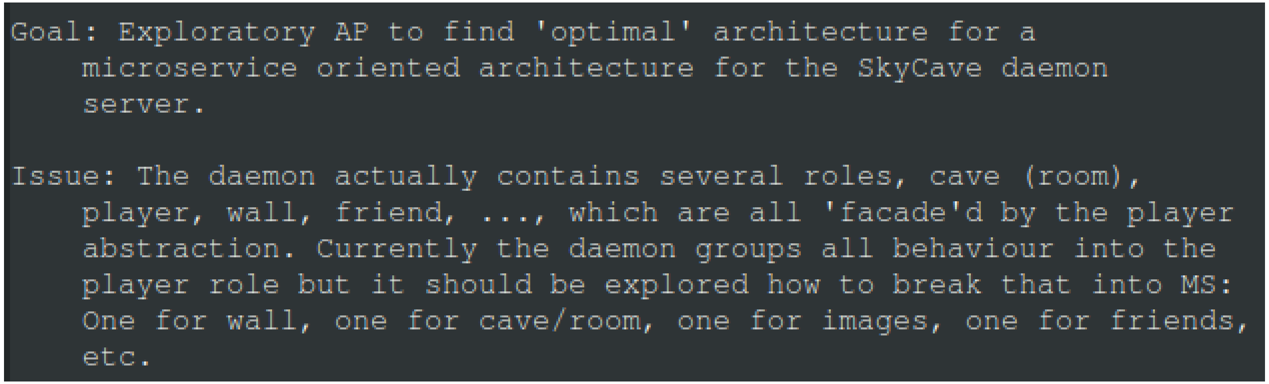
Retrofitting:

- *The process of inserting architecturally significant code/behavior or knowledge from a sub-AP into a super-AP.*

Example

Example 1: SkyCave API AP

- Context: Migration of architecture
  - Current architecture: 3-tiered architecture
    - Client – Server – Storage
  - *Hypothesis Future architecture*: MicroService architecture
    - Client – Gateway server – [cave service, room service, message service]
- [Strangler pattern]



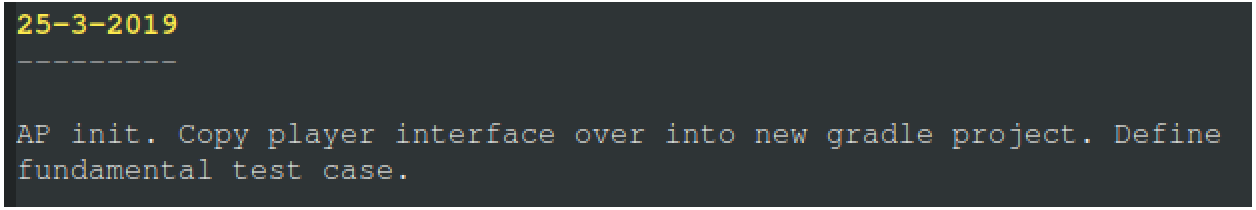
```
Goal: Exploratory AP to find 'optimal' architecture for a
    microservice oriented architecture for the SkyCave daemon
    server.

Issue: The daemon actually contains several roles, cave (room),
    player, wall, friend, ..., which are all 'facade'd by the player
    abstraction. Currently the daemon groups all behaviour into the
    player role but it should be explored how to break that into MS:
    One for wall, one for cave/room, one for images, one for friends,
    etc.
```

SkyCave AP

- Furthermore the AP studied
  - REST APIs
    - API Gateway must talk REST with three underlying services
  - Proper boundaries (bounded context) of each service
    - What are the proper interfaces? Coupling between them?
  - Meta:
    - *How to formulate the exercise associated?*
    - *Estimate student workload and evaluate feasibility of exercise*

Process

- Started by *harvesting process*



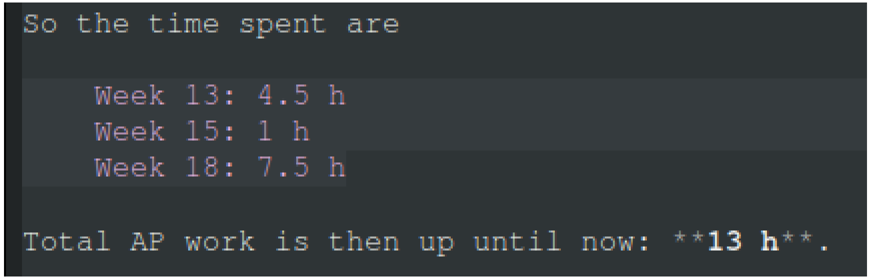
```
25-3-2019
---------

AP init. Copy player interface over into new gradle project. Define
fundamental test case.
```

- That is,
  - Copy the central gateway API (java interface *Player*) into a *new project, and only that and a few associated test cases.*
  - Resolve any *missing dependencies* by copying these interfaces/classes into the project as well
    - Sometimes *entity chain snipping* (Meszaros) when dependencies are irrelevant
      - *That is, cut of deep dependency tree with single level dummy interface/class*

Outcome

- Results
  - It is **doable**! So there is an exercise!
  - Boundaries are ok. Bounded contexts, no high coupling
  - REST APIs defined I can provide it / or not

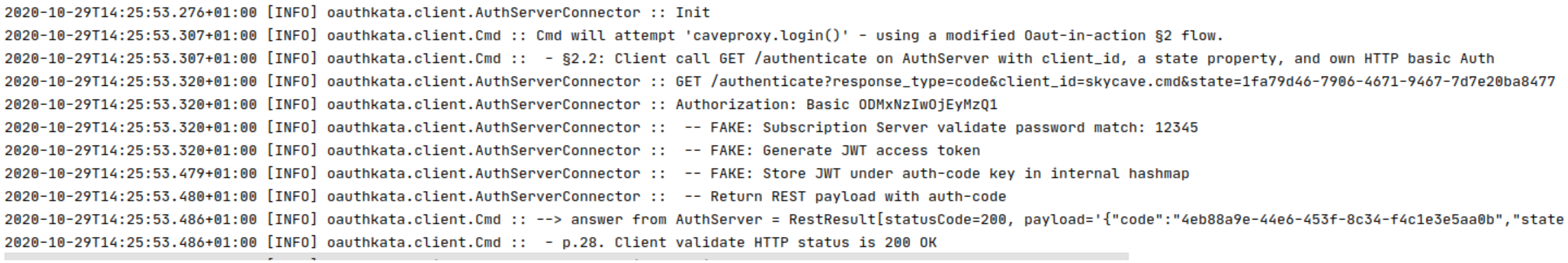
- Learning outcomes were then *retrofitted* into SkyCave
  - And the AP orphaned…



```
So the time spent are

    Week 13: 4.5 h
    Week 15: 1 h
    Week 18: 7.5 h

Total AP work is then up until now: **13 h**.
```

Example 2: SkyCave OAuth2

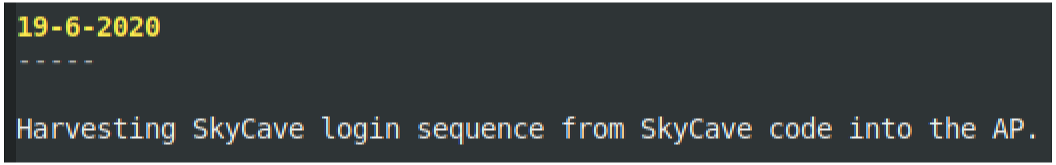
- Another example
  - Current Design: Homebrewed login system
  - Goal: Refactor SkyCave login system to mimic OAuth 2
- Process – *Take small steps*
  - **AP1**: Learn OAuth in a skycave *inspired* scenario
    - Protocol shown by log output
    - AuthenticationServer just a Test Stub



```
2020-10-29T14:25:53.276+01:00 [INFO] oauthkata.client.AuthServerConnector :: Init
2020-10-29T14:25:53.307+01:00 [INFO] oauthkata.client.Cmd :: Cmd will attempt 'caveproxy.login()' - using a modified Oaut-in-action §2 flow.
2020-10-29T14:25:53.307+01:00 [INFO] oauthkata.client.Cmd ::  - §2.2: Client call GET /authenticate on AuthServer with client_id, a state property, and own HTTP basic Auth
2020-10-29T14:25:53.320+01:00 [INFO] oauthkata.client.AuthServerConnector :: GET /authenticate?response_type=code&client_id=skycave.cmd&state=1fa79d46-7906-4671-9467-7d7e20ba8477
2020-10-29T14:25:53.320+01:00 [INFO] oauthkata.client.AuthServerConnector :: Authorization: Basic ODMxNzIwOjEyMzQ1
2020-10-29T14:25:53.320+01:00 [INFO] oauthkata.client.AuthServerConnector ::  -- FAKE: Subscription Server validate password match: 12345
2020-10-29T14:25:53.320+01:00 [INFO] oauthkata.client.AuthServerConnector ::  -- FAKE: Generate JWT access token
2020-10-29T14:25:53.479+01:00 [INFO] oauthkata.client.AuthServerConnector ::  -- FAKE: Store JWT under auth-code key in internal hashmap
2020-10-29T14:25:53.480+01:00 [INFO] oauthkata.client.AuthServerConnector ::  -- Return REST payload with auth-code
2020-10-29T14:25:53.486+01:00 [INFO] oauthkata.client.Cmd :: --> answer from AuthServer = RestResult[statusCode=200, payload='{"code":"4eb88a9e-44e6-453f-8c34-f4c1e3e5aa0b","state
2020-10-29T14:25:53.486+01:00 [INFO] oauthkata.client.Cmd ::  - p.28. Client validate HTTP status is 200 OK
```

SkyCave OAuth2

- Process – *Take small steps*
  - **AP2**: Harvest SkyCave login and adapt to OAuth2

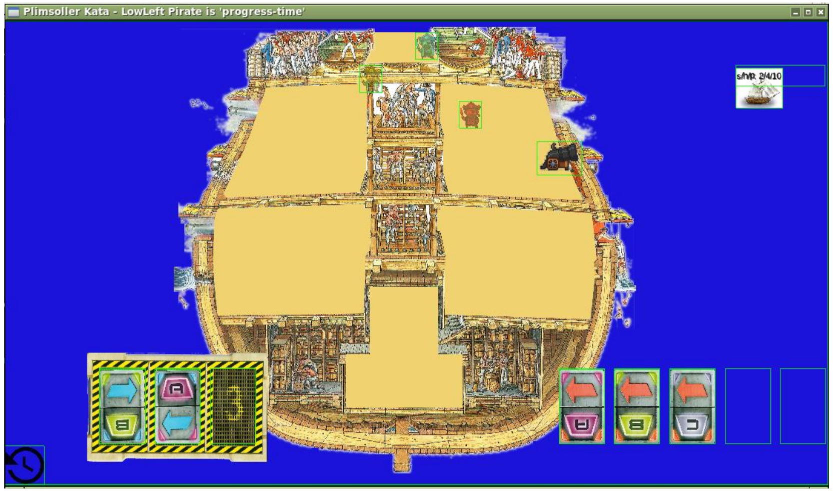


```
19-6-2020
-----

Harvesting SkyCave login sequence from SkyCave code into the AP.
```

  - That is, *copy ‘cave.login(user,pwd)’* and a test case that log in ‘magnus’, snip off all other irrelevant code, and make test case pass.
- Next
  - Gradually refactor login algorithms to conform to protocol learned from the first AP.
- Conclusion: Doable. (Retrofitting pending ☺)

Example 3: Strikketøj / Kodetøj

- What do a programmer do when the kids are grown up, and the wife spends the evening knitting?
  - Become an Indie Game Developer (?)
    - Success criteria:
      - Sell at least 5 copies at a price of 0 kr.
      - Have fun ☺
  - Turn-based, collaborative game play…
    - "Space Alert" inspired…
  - LibGdx Java game engine



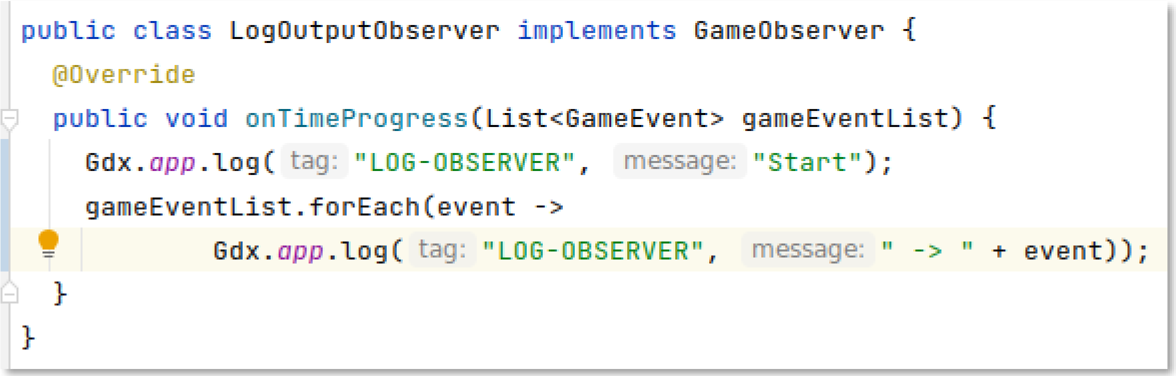
But…

- First draft of part of the game domain code in place
- Haphazard junk pile of web scraped PNG files on the UI
  - I am no graphical artist ☺
- *And a Game Event protocol*
  - *Domain (server) executes round => List of GameEvents*
  - *Send list to all player UI (clients)*
  - *Replay animations that mimic these game events*

- Issue:
  - Game code base is in the way of animation experimentation…

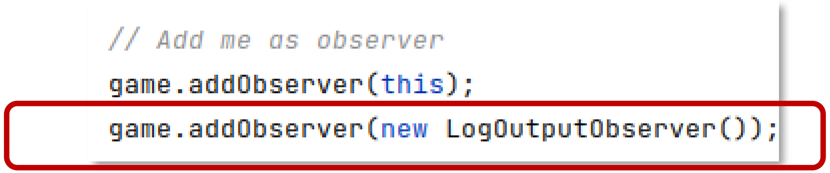
Instrumentation

- Create a LogObserver, to get the *ground truth*



```
public class LogOutputObserver implements GameObserver {
  @Override
  public void onTimeProgress(List<GameEvent> gameEventList) {
    Gdx.app.log( tag: "LOG-OBSERVER",  message: "Start");
    gameEventList.forEach(event ->
            Gdx.app.log( tag: "LOG-OBSERVER",  message: " -> " + event));
  }
}
```

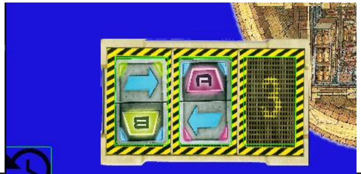
  - The UI observes what happens in the domain
    - And also add this logging observer

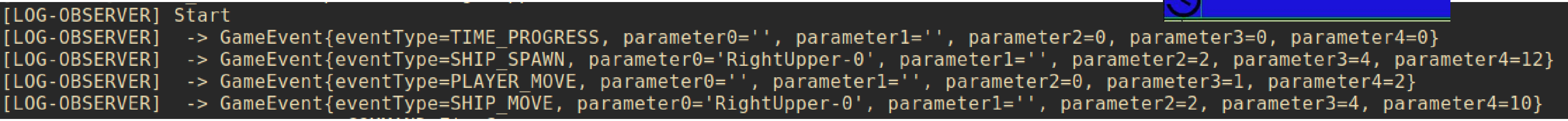


```
// Add me as observer
game.addObserver(this);
game.addObserver(new LogOutputObserver());
```

Harvesting

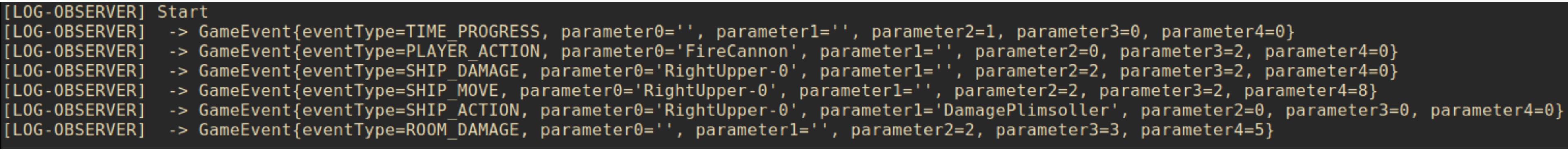
- And play a scenario (Record and Playback tactic ☺)
  - Pirate moves to right deck
    - An enemy ship appears, and moves





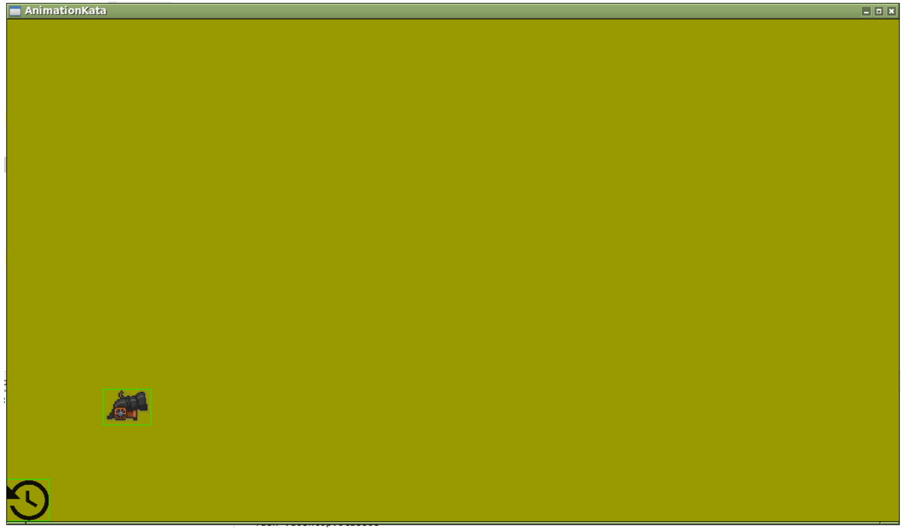
```
[LOG-OBSERVER] Start
[LOG-OBSERVER]  -> GameEvent{eventType=TIME_PROGRESS, parameter0='', parameter1='', parameter2=0, parameter3=0, parameter4=0}
[LOG-OBSERVER]  -> GameEvent{eventType=SHIP_SPAWN, parameter0='RightUpper-0', parameter1='', parameter2=2, parameter3=4, parameter4=12}
[LOG-OBSERVER]  -> GameEvent{eventType=PLAYER_MOVE, parameter0='', parameter1='', parameter2=0, parameter3=1, parameter4=2}
[LOG-OBSERVER]  -> GameEvent{eventType=SHIP_MOVE, parameter0='RightUpper-0', parameter1='', parameter2=2, parameter3=4, parameter4=10}
```

- Pirate fires the cannon and damage enemy
  - Enemy ship moves, fires, and right deck is damaged

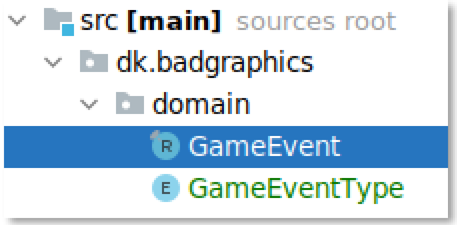


```
[LOG-OBSERVER] Start
[LOG-OBSERVER]  -> GameEvent{eventType=TIME_PROGRESS, parameter0='', parameter1='', parameter2=1, parameter3=0, parameter4=0}
[LOG-OBSERVER]  -> GameEvent{eventType=PLAYER_ACTION, parameter0='FireCannon', parameter1='', parameter2=0, parameter3=2, parameter4=0}
[LOG-OBSERVER]  -> GameEvent{eventType=SHIP_DAMAGE, parameter0='RightUpper-0', parameter1='', parameter2=2, parameter3=2, parameter4=0}
[LOG-OBSERVER]  -> GameEvent{eventType=SHIP_MOVE, parameter0='RightUpper-0', parameter1='', parameter2=2, parameter3=2, parameter4=8}
[LOG-OBSERVER]  -> GameEvent{eventType=SHIP_ACTION, parameter0='RightUpper-0', parameter1='DamagePlimsoller', parameter2=0, parameter3=0, parameter4=0}
[LOG-OBSERVER]  -> GameEvent{eventType=ROOM_DAMAGE, parameter0='', parameter1='', parameter2=2, parameter3=3, parameter4=5}
```

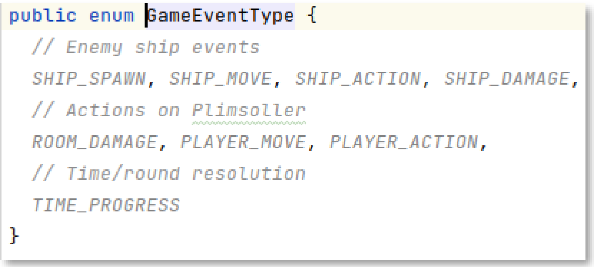
Harvesting



- Now I can
  - Make a new LibGdx project
  - Make a simple ‘replay()’ method that construct the two event lists, one for each turn
  - Requires *harvesting* code from original to the AP



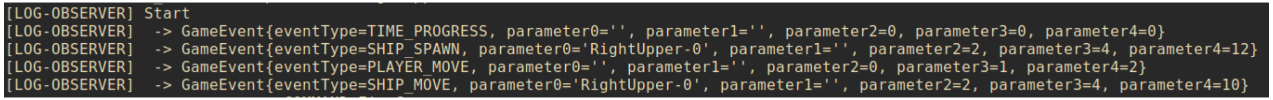
```
src [main] sources root
  dk.badgraphics
    domain
      GameEvent
      GameEventType
```



```java
public enum GameEventType {
  // Enemy ship events
  SHIP_SPAWN, SHIP_MOVE, SHIP_ACTION, SHIP_DAMAGE,
  // Actions on Plimsoller
  ROOM_DAMAGE, PLAYER_MOVE, PLAYER_ACTION,
  // Time/round resolution
  TIME_PROGRESS
}
```

Harvesting

- Now I can
  - Make a simple ‘replay()’ method that construct two event lists, one for each turn



```
[LOG-OBSERVER] Start
[LOG-OBSERVER]  -> GameEvent{eventType=TIME_PROGRESS, parameter0='', parameter1='', parameter2=0, parameter3=0, parameter4=0}
[LOG-OBSERVER]  -> GameEvent{eventType=SHIP_SPAWN, parameter0='RightUpper-0', parameter1='', parameter2=2, parameter3=4, parameter4=12}
[LOG-OBSERVER]  -> GameEvent{eventType=PLAYER_MOVE, parameter0='', parameter1='', parameter2=0, parameter3=1, parameter4=2}
[LOG-OBSERVER]  -> GameEvent{eventType=SHIP_MOVE, parameter0='RightUpper-0', parameter1='', parameter2=2, parameter3=4, parameter4=10}
```

  - And now experiment with a design for how to execute animations

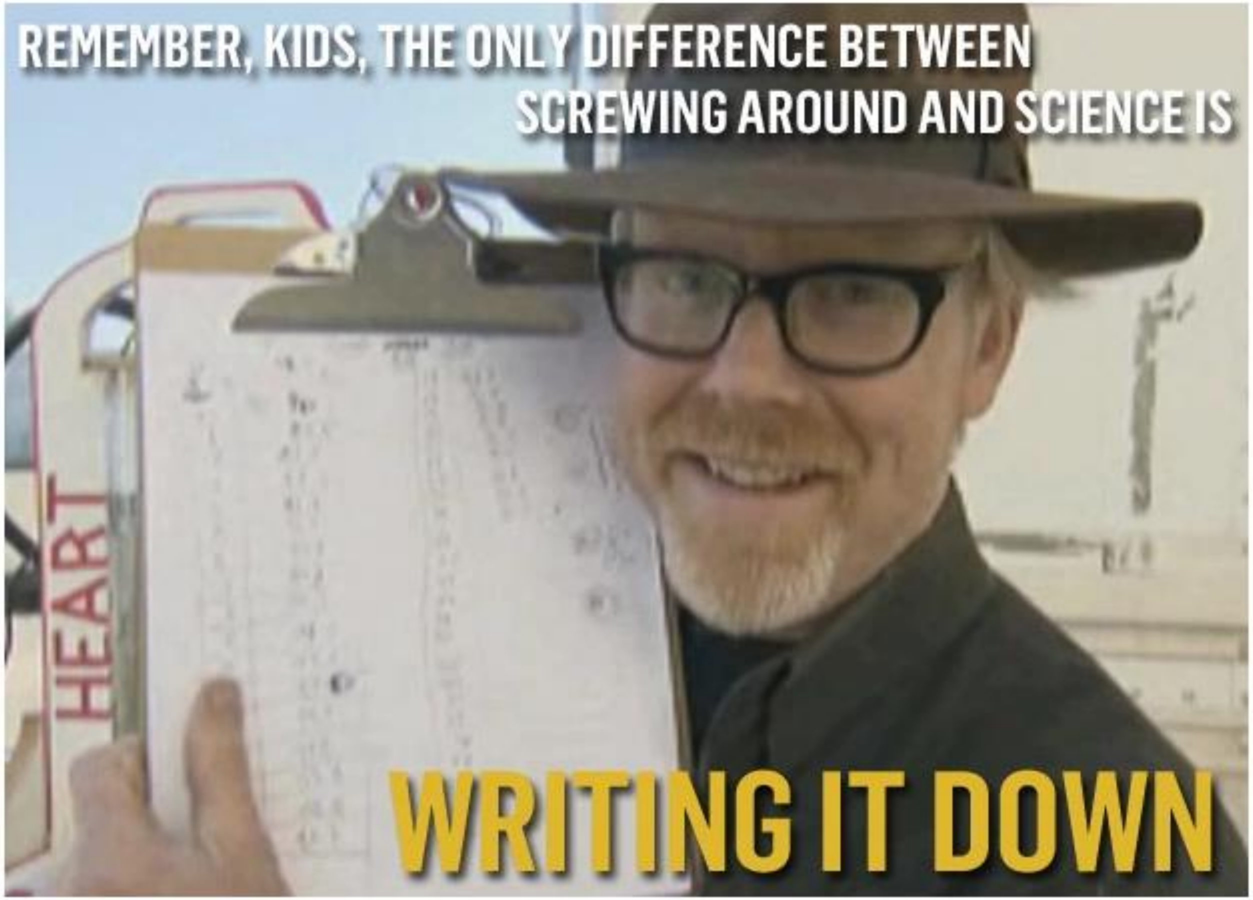
- Once ‘satisfied’ this code can then be retrofitted back…

AARHUS UNIVERSITET

Experimental Record Keeping

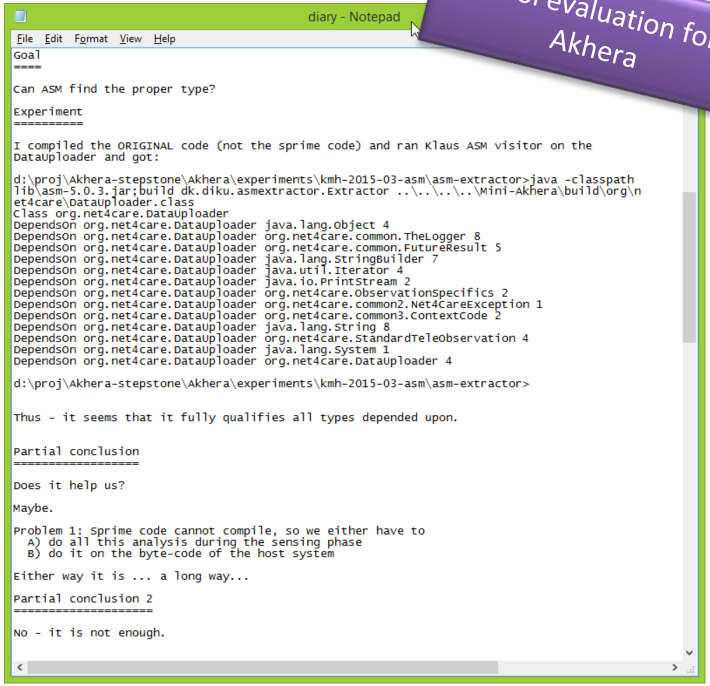
Central in Scientific Process

Andy Savage/MythBusters


REMEMBER, KIDS, THE ONLY DIFFERENCE BETWEEN
SCREWING AROUND AND SCIENCE IS
WRITING IT DOWN
HEART

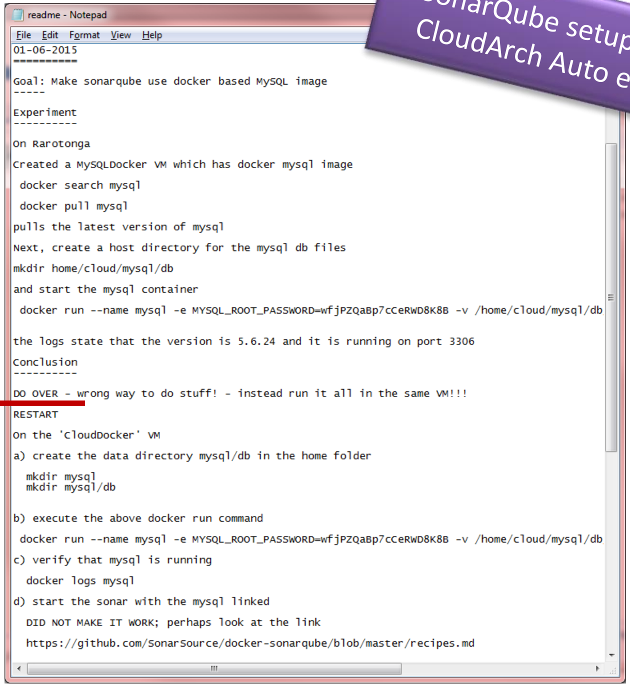
Examples: Experiments

- Keep the *diary/logbook* at hand! Make **notes!**
  - **Goal, hypothesis, experiment, conclude [loop]**



```
Goal
====

Can ASM find the proper type?

Experiment
==========

I compiled the ORIGINAL code (not the sprime code) and ran Klaus ASM visitor on the
DataUploader and got:

d:\proj\Akhera-stepstone\Akhera\experiments\kmh-2015-03-asm\asm-extractor>java -classpath
lib\asm-5.0.3.jar;build dk.diku.asmextractor.Extractor ..\..\..\..\Mini-Akhera\build\org\n
et4care\DataUploader.class
Class org.net4care.DataUploader
DependsOn org.net4care.DataUploader java.lang.Object 4
DependsOn org.net4care.DataUploader org.net4care.common.TheLogger 8
DependsOn org.net4care.DataUploader org.net4care.common.FutureResult 5
DependsOn org.net4care.DataUploader java.lang.StringBuilder 7
DependsOn org.net4care.DataUploader java.util.Iterator 4
DependsOn org.net4care.DataUploader java.io.PrintStream 2
DependsOn org.net4care.DataUploader org.net4care.ObservationSpecifics 2
DependsOn org.net4care.DataUploader org.net4care.common2.Net4CareException 1
DependsOn org.net4care.DataUploader org.net4care.common3.ContextCode 2
DependsOn org.net4care.DataUploader java.lang.String 8
DependsOn org.net4care.DataUploader org.net4care.StandardTeleObservation 4
DependsOn org.net4care.DataUploader java.lang.System 1
DependsOn org.net4care.DataUploader org.net4care.DataUploader 4

d:\proj\Akhera-stepstone\Akhera\experiments\kmh-2015-03-asm\asm-extractor>


Thus - it seems that it fully qualifies all types depended upon.


Partial conclusion
==================

Does it help us?

Maybe.

Problem 1: Sprime code cannot compile, so we either have to
  A) do all this analysis during the sensing phase
  B) do it on the byte-code of the host system

Either way it is ... a long way...

Partial conclusion 2
====================

No - it is not enough.
```

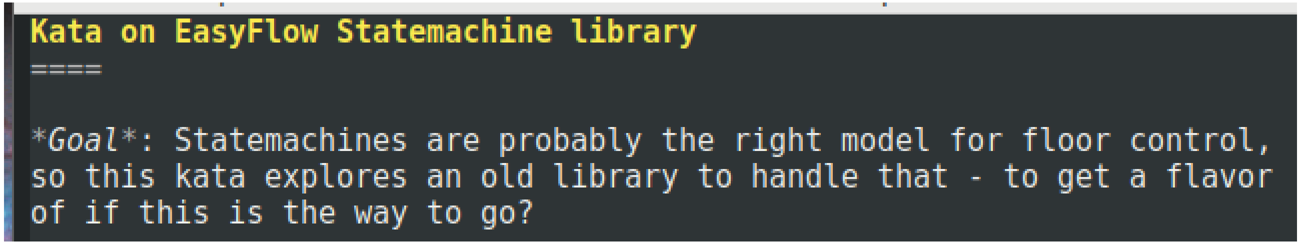
Tool evaluation for Akhera



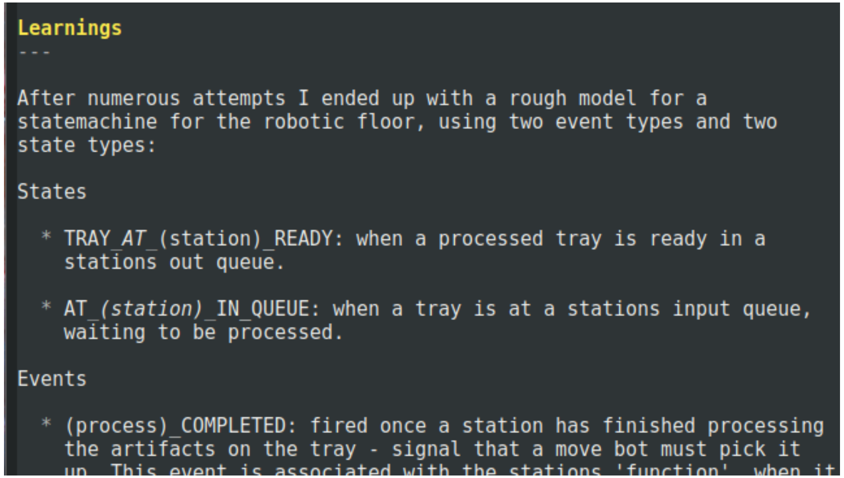
```
01-06-2015
==========

Goal: Make sonarqube use docker based MySQL image
-----

Experiment
----------

On Rarotonga

Created a MySQLDocker VM which has docker mysql image

 docker search mysql

 docker pull mysql

pulls the latest version of mysql

Next, create a host directory for the mysql db files

mkdir home/cloud/mysql/db

and start the mysql container

 docker run --name mysql -e MYSQL_ROOT_PASSWORD=wfjPZQaBp7cCeRWD8K8B -v /home/cloud/mysql/db

the logs state that the version is 5.6.24 and it is running on port 3306

Conclusion
----------

DO OVER - wrong way to do stuff! - instead run it all in the same VM!!!

RESTART

On the 'CloudDocker' VM

a) create the data directory mysql/db in the home folder

  mkdir mysql
  mkdir mysql/db


b) execute the above docker run command

 docker run --name mysql -e MYSQL_ROOT_PASSWORD=wfjPZQaBp7cCeRWD8K8B -v /home/cloud/mysql/db

c) verify that mysql is running

  docker logs mysql

d) start the sonar with the mysql linked

  DID NOT MAKE IT WORK; perhaps look at the link

  https://github.com/SonarSource/docker-sonarqube/blob/master/recipes.md
```

SonarQube setup for CloudArch Auto eval

Examples: Summarize Learnings



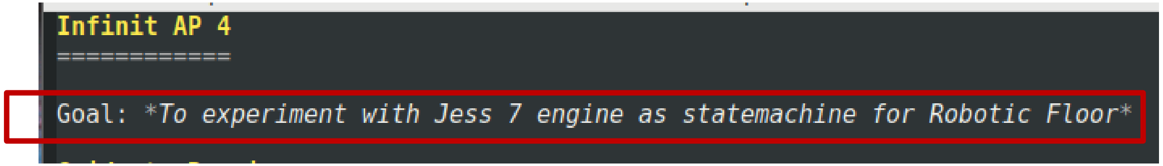
```
Kata on EasyFlow Statemachine library
====

*Goal*: Statemachines are probably the right model for floor control,
so this kata explores an old library to handle that - to get a flavor
of if this is the way to go?
```



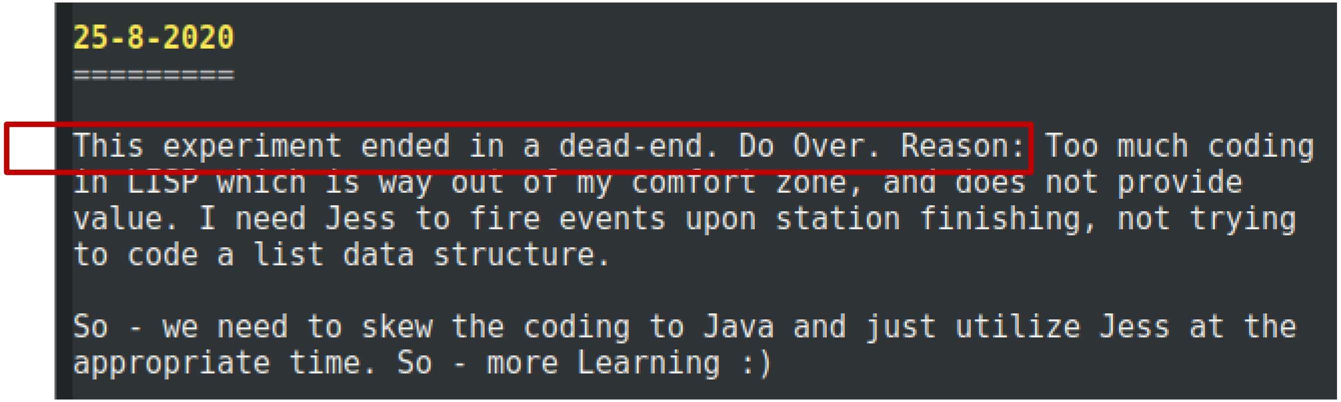
```
Learnings
---

After numerous attempts I ended up with a rough model for a
statemachine for the robotic floor, using two event types and two
state types:

States

  * TRAY_AT_(station)_READY: when a processed tray is ready in a
    stations out queue.

  * AT_(station)_IN_QUEUE: when a tray is at a stations input queue,
    waiting to be processed.

Events

  * (process)_COMPLETED: fired once a station has finished processing
    the artifacts on the tray - signal that a move bot must pick it
    up. This event is associated with the stations 'function', when it
```

The Robotic Manufacuring Case

Examples: AP Goal + Conclusion



```
Infinit AP 4
============

Goal: *To experiment with Jess 7 engine as statemachine for Robotic Floor*
```



```
25-8-2020
=========

This experiment ended in a dead-end. Do Over. Reason: Too much coding
in LISP which is way out of my comfort zone, and does not provide
value. I need Jess to fire events upon station finishing, not trying
to code a list data structure.

So - we need to skew the coding to Java and just utilize Jess at the
appropriate time. So - more Learning :)
```

Discussion

- Architectural Prototype is not invented, it is discovered ☺
  - Our job was to formalize it a bit
- Bass used to call them *Walking Skeletons*
  - *Akin the ‘evolutionary AP’*
- Klaus Marius Hansen, Architect at Microsoft
  - ”I spent quite a lot of time programming architectural prototypes…”