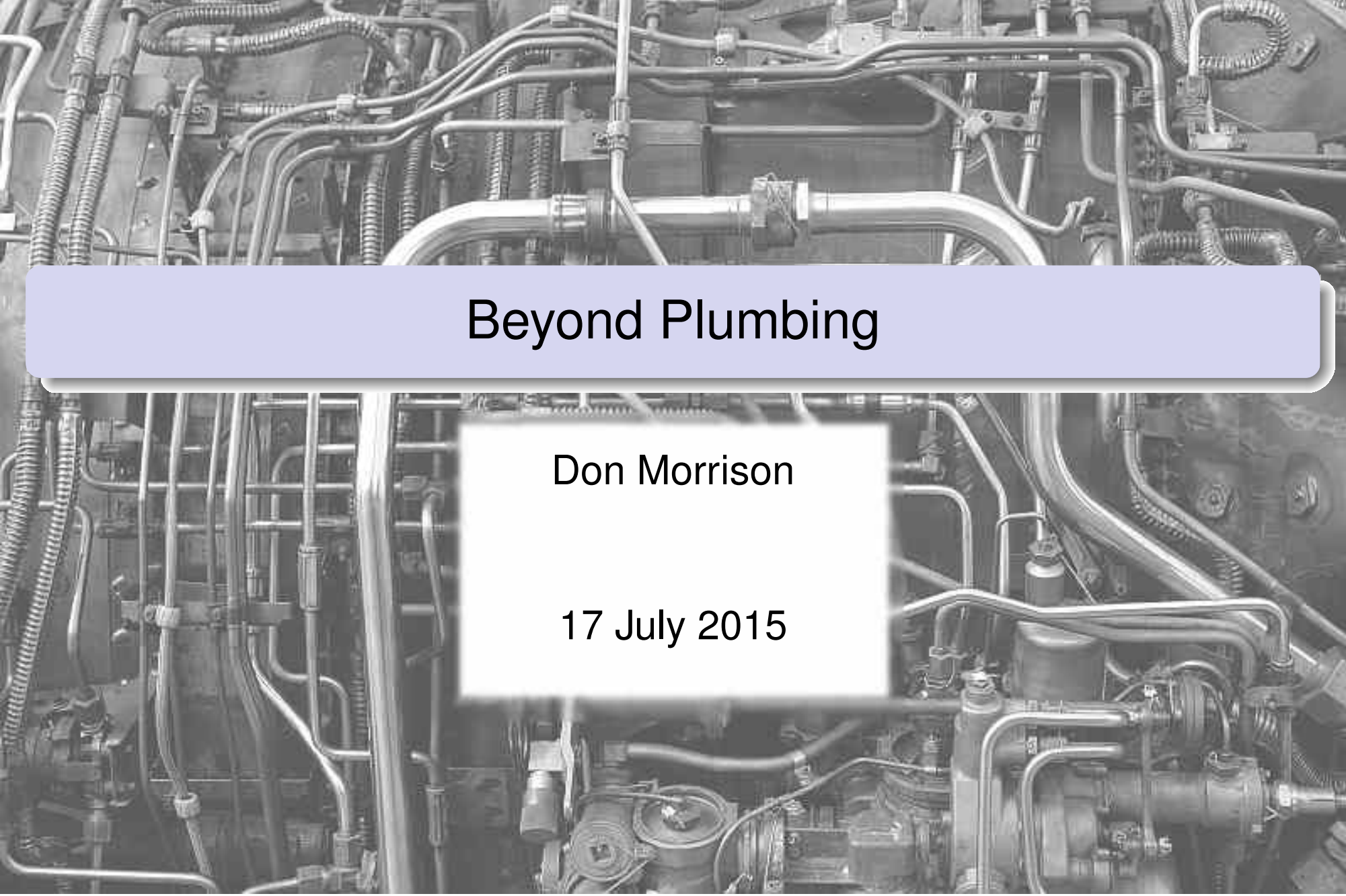

Beyond Plumbing

Don Morrison

17 July 2015

Carnegie Mellon University

What we do

Diverse projects have a similar shapes, across multiple groups

- Christian Lebiere's group
- Coty Gonzales's Dynamic Decision Making Laboratory

We're connecting a cognitive model

- written in ACT-R
- or PyIBL (a Python based Instance Based Learning toolkit)
- or ...

across a wire to something else, usually designed by someone else

- a robot
- a simulation of some kind
- another model, possibly an opponent in a game
- a human opponent in a game
- or ...

Carnegie Mellon University

How we start

- We think about how we're going to model our task
  - maybe we even prototype it with made up, local data
  - if we're lucky, maybe we even reuse parts of an existing model
- We figure out the plumbing, how we're going to get that data
  - Usually lots of low-level details, like network protocols and formats
  - If we're lucky maybe we even have a say in those details
  - Even if they're documented, subtle details are often overlooked

Syntax

- A popular syntax today is JSON. It's got lots of nice properties. There are parsers available for nearly every programming language, and it maps nicely both to Lisp S-expressions and to ACT-R chunks.
- Here's data from a robotics project sent to a model trying to deduce which walls constitute a building.


```
[[13,[{"name":"wall1314199","id":14,"lowerLeft"
5.296298,"y":-2.6026878,"z":-0.32568398},"upper
{"x":4.5158796,"y":0.18326342,"z":-2.694102},"u
"5a39457f2d3e4425aabe7f5bf8e8a599","timestamp":
1385139718489201917,"confidence":0.0,"classType
"internalState":0,"wmId":5561119446747228055,
"parentObject":13,"moveConfidence":0.0,"manipul
0.0,"quality":0.0,"hypothesized":0.0,"relPose":
...
```


Carnegie Mellon University

But it's only syntax


```
[[13,
 [{"name":"wall1314199", "id":14,
   "lowerLeft":{"x":5.296298,"y":-2.6026878,"z":-0.3
   "upperRight":{"x":4.5158796,"y":0.18326342,"z":-2
   "uuid":"5a39457f2d3e4425aabe7f5bf8e8a599",
   "timestamp":1385139718489201917,
   "classType":11, "internalState":0
   "wmId":5561119446747228055, "parentObject":13,
   ...
```


- Yes, the syntax and protocols are important, but they just scratch the surface.
- What really matters is what fields might be there? And what do they mean?

Carnegie Mellon University

More than just fields

- Grain size
  - In that same robotics project when planning how to navigate around a building we were still passed all that low-level geometry detail.
  - And then we had to pass it back again when we needed to ask for clarification of some ambiguity.
  - And got it back again....
  - It's not so much a matter of being inefficient, it's that it makes things incomprehensible.
    - A hierarchical decomposition is easier to understand,
    - and maps more naturally to the chunks in the model.
- Related is the lifetime of information and which side of the wire it's stored on.
  - That was a source of big surprises, with conflicting assumptions.

It's these semantics the modeler really needs

- They're completely different on every project.
  - Even when they look the same, they're probably different.
- More often than not we find out what many, or even all, the possibilities are by just trial and error.
- And, more often than not, the only way we find out the semantics is by asking someone working on the other end of the wire.
- It's a long, laborious process.
- And we have to do it over again next time.
- In fact, we probably have to do it over, and over again, this time:
  - Stuff changes
  - For example, in the building prediction example part-way through the project the coordinate system was changed, making many of those fields mean something completely different than they used to.

Isn't it just a matter of documentation?

- Writing good documentation is hard, and time consuming.
  - The author doesn't know everything the consumer's going to need, so they must jointly iterate.
- If we're lucky, maybe there's documentation
  - If we're really lucky, maybe it used to be accurate.
  - Even if it's accurate today, it won't be next week.
  - And it was never complete.
- In practice, even when there is documentation it tends to emphasize the syntax, and only the superficial aspects of the semantics. The producer doesn't know what the modeler's really going to need.
- We're all working in a research environment: which are we more likely to do, write good documentation, or write the next paper?

So what can we do?

- Can we create some kind of shared standards?
- Can we leverage the work of folks trying to solve similar problems?
  - Semantic Web / Ontologies
  - This problem crops up all across the software engineering universe: what else is out there we can try to use?
- It's a close cousin of the model reuse problem, too.
  - If on every project we have to reinvent how we understand the data we're pulling into our models, it makes it far less likely we're going to be able to reuse those models.
  - The not unnatural tendency to view a model as the solution to one problem, crafted specificially for the production of the next paper, to be thrown away afterwards, makes it even harder to solve these problems.

We want it all

- Maximize generality
  - Capture the broadest space of interaction between task and model.
- Minimize programmer intervention
  - Focus on the model, not the plumbing.
- Enable very general models
  - Eliminate the need to craft a new model for each task.

It's hierarchical

- It's like eating an elephant, you gotta do it one bite at a time.
- Even standards of just a high level decomposition help.
  - Recognize that nearly all tasks have contexts, actions, outcomes, etc.
  - Dan Veksler's Standard Task-Actor Protocol is one stab at this.
- The next level down gets more domain specific.
  - Are there groups of domains that we can combine, and share more standards?
  - Even if we can never get to full generality, every step closer will help.

Carnegie Mellon University