Made in MAKAO

- Re(verse)-engineering build systems -

Bram Adams

Ghent University
Department of Information Technology
Sint-Pietersnieuwstraat 41
9000 Ghent, Belgium
http://www.intec.ugent.be

bram.adams@ugent.be
http://users.ugent.be/~badams/makao

Build system

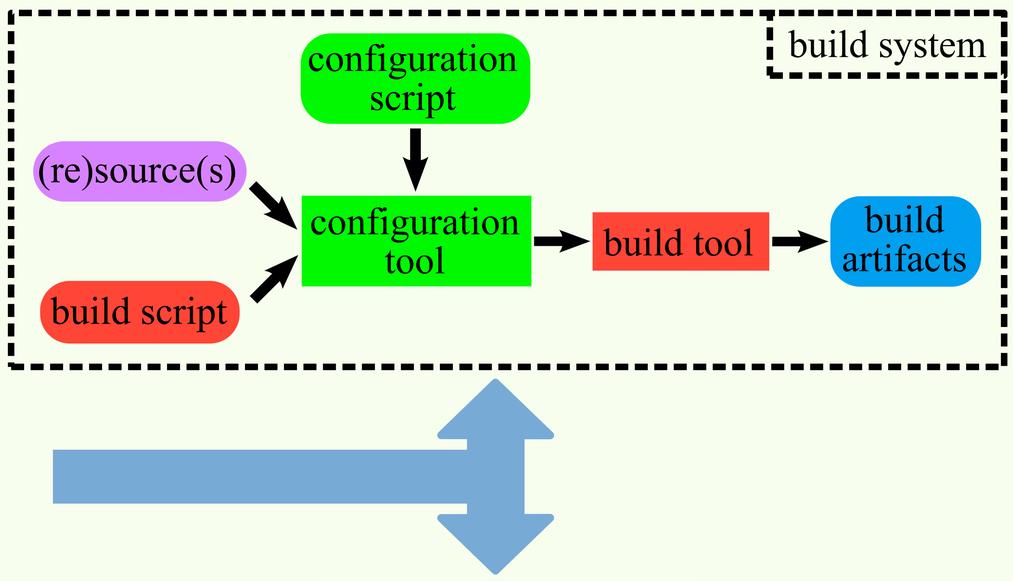
... → 1977: ad hoc build and install scripts

1977: "make" [1] ➡ great improvements in incremental compilation!

- declarative specification of dependencies between targets
- build recipe is imperative list of commands and macros
- interpreter: only (re)build target if it's older than any dependency

1977 → ...: configurability

- configuration script: high-level, platform-independent description
- build script: generated per platform



MAKAO

Build systems contain valuable data for various stakeholders about all facets of software.

Makefile Architecture Kernel for Aspect Orientation (MAKAO):

- extract and offer this knowledge to all stakeholders
- enhance data gained by source-code techniques
- aid in re(verse)-engineering of build systems

How?

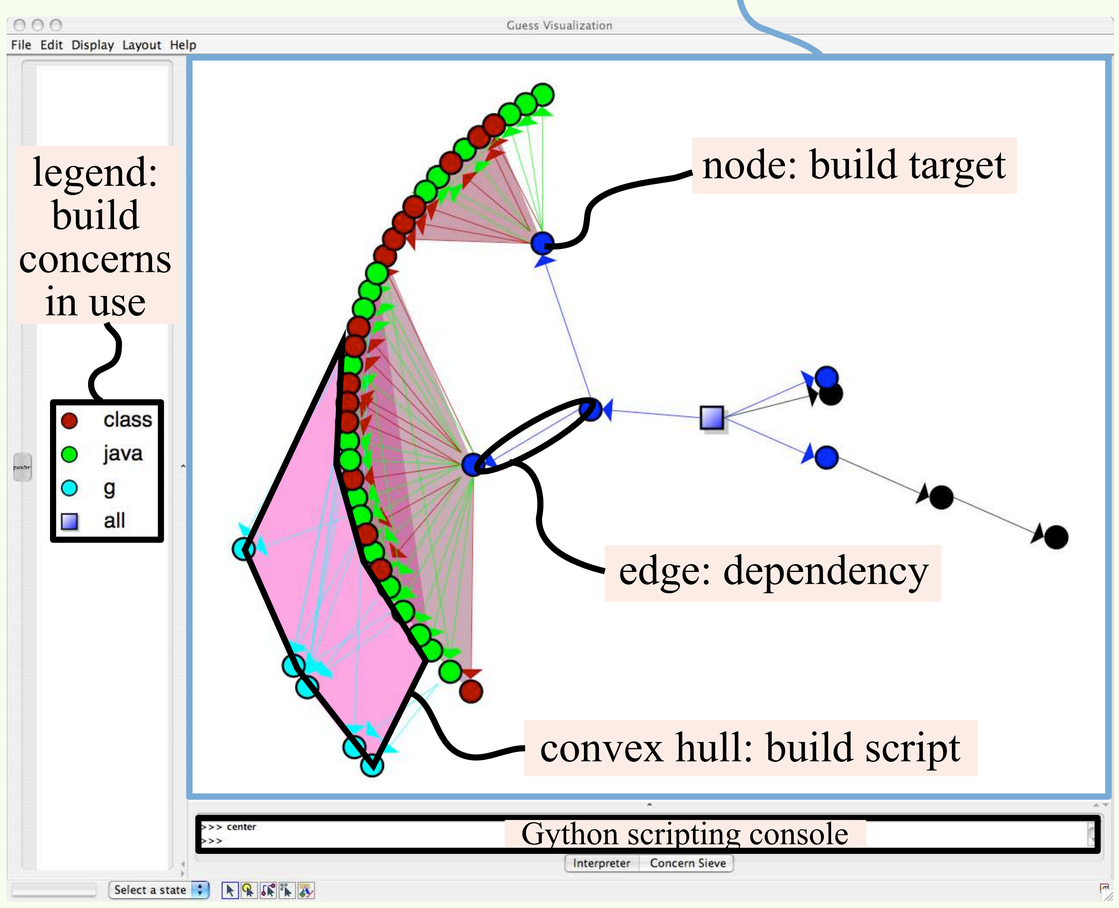
- represent dynamic traces of concrete builds

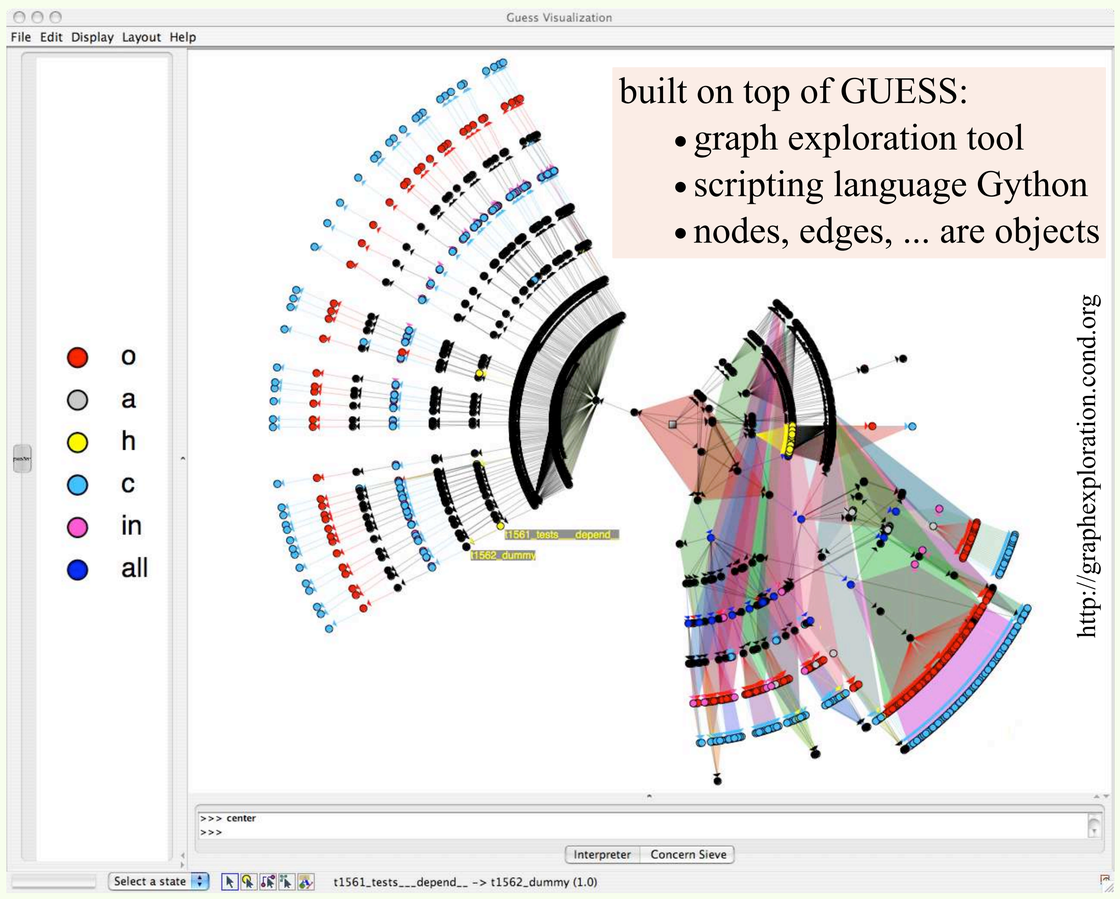
➡ Directed Acyclic Graph (DAG)

- retain links back to static build data

STAKEHOLDERS

Developer	• Assess effects of code • Try to find cause of build error • Where to change build system when adding new things?
Maintainer	• Learn inner mechanics of a new system • Check for dead code • Profiling of build
(Power) User	Find out about library dependencies
QA	Add feature and regression tests to suite
Researcher	Uninvasively integrate experimental tools





Approach

EXPLORER:

- explore and navigate through DAG
- discover available build concerns
- find spread of targets over build scripts

1

➡ example: add source code preprocessor

FINDER:

2

Query for targets and commands based on properties.

```
targets=(concern=="c").inEdges.node1.findNodes()
base=[(command,tool,T) for T in targets
                       for command in commands[T]
                       for tool in ["gcc","(CC)"]
                       if command.find(tool)!=-1]
```

ADVISER:

3

Compose modifications for dependencies and recipes.

```
advice=["\n".join(
                    [c.replace(t,t+" -E")+" -o $<",
                     "aspicere.sh $<"])
                for (c,t,T) in base                 ]
```

WEAVER:

4

Apply modifications: • logically (in-memory)
• physically (scripts)

```
weave_before([T for (c,t,T) in base],
             [c for (c,t,T) in base],
             advice)
```

Conclusion & Future Work

MAKAO helps to re(verse)-engineer build systems ↔ TODO: more surveyable visualization, other case studies, DSL on top of Gython, ...

[1] S. I. Feldman. Make-a program for maintaining computer programs. Software - Practice and Experience, 1979.