TU Clausthal
Clausthal University of Technology

LightJason

A BDI Framework inspired by Jason

M. Aschermann, Ph. Kraus, J. P. Müller
Clausthal University of Technology
15. Dec. 2016

Motivation – Goal

- large and complex application domains e.g. (multimodal) traffic, shared spaces, product lifecycle management, . . .
- millions of agents with complex behaviour
- a domain-independent scripting language

Meskel Square (Addis Ababa, Ethiopia)
http://www.youtube.com/watch?v=UEIn8GJIg0E

Motivation – About us

- graduation in Business Information Technology (eBusiness, ERP, decentralised product models, data/web mining)
- research field: multi-agent-based simulations, microscopic traffic manoeuvres and traffic coordination mechanisms

- apprenticeship as software-developer
- software-developer (freelance) 15 years
- graduation in theoretical computer-science (high-scalability, machine learning)
- research field: high-scalability, distributed multi-agent systems and big data

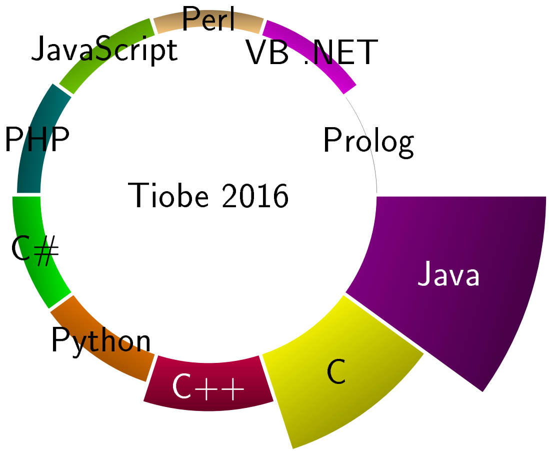
Motivation – Requirements

- state-of-the-art technologies, concurrency support, established software design-pattern
- Clean-Code[1] development and continuous integration workflow
- well documented software (not just “documented by research papers”)

- portability to existing platforms and frameworks
- cloud platform support for high-scalability

[1] by Robert C. Martin

Motivation – Logical Languages Rarely Used



- [TIOBE, 2016]: Only listed logical language (Prolog) ranked 33rd.
- [PopularitY, 2016, RedMonk, 2016] similar; logical languages ranked out.

Methods – Identification of Related Academic Work

- 2APL
- CArtAgO
- GOAL
- Jade
- Jadex
- Jason
- Mason
- Moise

(Java-based)

Methods – Analysis of Related Academic Work

FindBugs (http://findbugs.sourceforge.net/) developed by University of Maryland, supported by Google and Oracle, detects following errors:

- malicious code vulnerability, correctness, security
- bad practice, internationalisation, dodgy code
- performance, multithreaded correctness, experimental code

JDepend (http://clarkware.com/software/JDepend.html) measures code quality through metrics. Measurement of quality for each package of

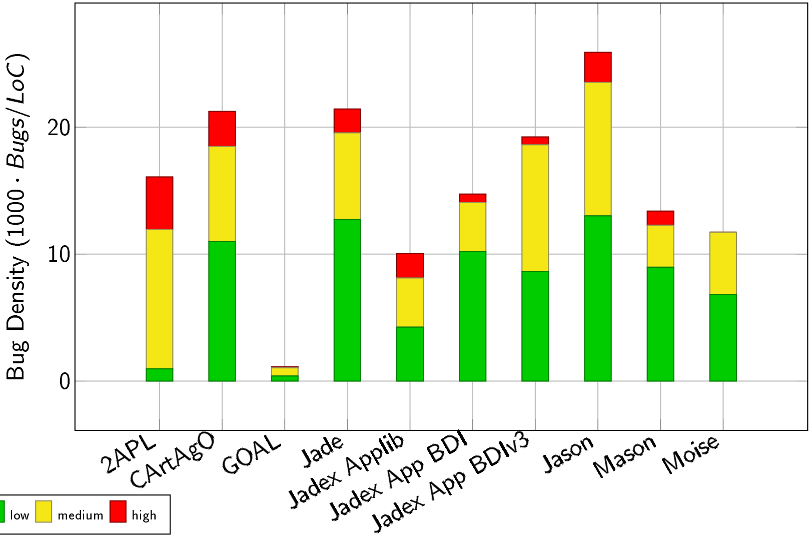
- extensibility efficiency
- reusability efficiency
- maintainability efficiency

TU Clausthal
Clausthal University of Technology

Methods – FindBugs: Code Quality Example

```
List<Belief> l = new ArrayList();
for( int i=0; i < 1000; i++ )
{
    Belief x = this.generate_belief();
    l.add(x);
}
```

Methods – FindBugs: Results



Methods – JDepend: Definitions

JDepend (http://clarkware.com/software/JDepend.html) measures code quality through the following metrics:

Abstractness (A): Defines the ratio of abstractness

$$A := \frac{\sum \text{interfaces} + \sum \text{abstract classes}}{\sum \text{all items}}$$

Instability (I): Indicator of the resilience to change

$$I := \frac{\sum \text{classes which referenced by other packages}}{\sum \text{classes which are references outside and inside the package}}$$

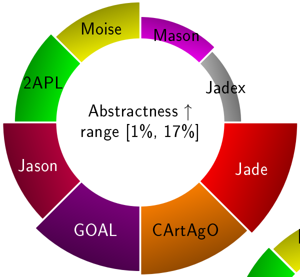
Distance (D): Indicator of balance between abstractness and stability

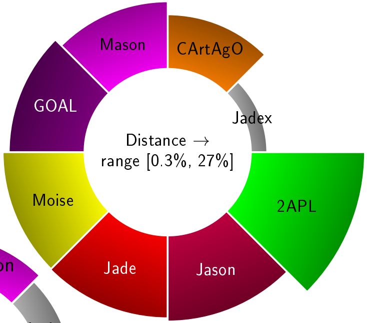
$D := A + I \Rightarrow 1$ (for ideal packages)

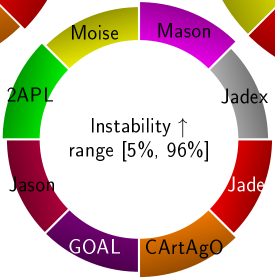
- completely abstract and stable ($A = 1 \wedge I = 0$)
- completely concrete and instable ($A = 0 \wedge I = 1$)

TU Clausthal
Clausthal University of Technology

Methods – JDepend: Results







Methods – Summary

Analysed MAS platforms do not satisfy our requirements

- no easy integration into existing software because of built-in runtimes
- no high-scalability for cloud platform support
- no well-written source code with clean architecture

- poor quality and lack of state-of-the-art developing technologies
- mostly poor code ⇒ expandable mainly by trial and error

99 little bugs in the code.
99 little bugs in the code.
Take one down, patch it around.
127 little bugs in the code...

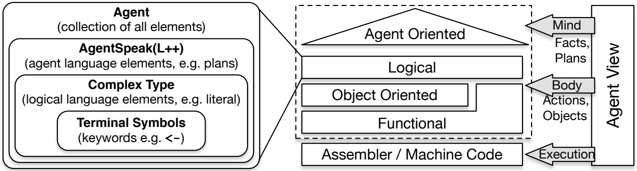
http://www.sjcnet.co.uk/2014/06/08/image-99-little-bugs/

LightJason – Contribution

- AgentSpeak(L++) based on AgentSpeak(L) [Rao, 1996, Bordini et al., 2007], but
  - has a modularised grammar written with AntLR
  - redesigned for concurrent execution
  - written in Java 1.8 with state-of-the-art techniques

LightJason – Contribution

- AgentSpeak(L++) based on AgentSpeak(L) [Rao, 1996, Bordini et al., 2007], but
  - has a modularised grammar written with AntLR
  - redesigned for concurrent execution
  - written in Java 1.8 with state-of-the-art techniques
- *Hybrid programming language* (logical, functional & imperative components)



- for more details, see technical report [Kraus et al., 2016]

LightJason – Contribution (work in progress)

- well-documented source code ✓
- state-of-the-art developing process and techniques ✓
- clean and well-structured software architecture (based on metrics) ✓
- benchmarks show fairly and evenly distributed workload for 15.000 agents with $>$ 10.000 beliefs on regular desktop PCs ✓

$\Rightarrow$ fulfilled requirements stated in motivation

LightJason – Contribution (work in progress)

- well-documented source code ✓
- state-of-the-art developing process and techniques ✓
- clean and well-structured software architecture (based on metrics) ✓
- benchmarks show fairly and evenly distributed workload for 15.000 agents with $>$ 10.000 beliefs on regular desktop PCs ✓

$\Rightarrow$ fulfilled requirements stated in motivation

- fuzziness
- explicit repair-planning
- built-in concurrency and supporting components e.g. BLAS, crypto, ...
- optimisation with scoring function

LightJason – Contribution (work in progress)

- well-documented source code ✓
- state-of-the-art developing process and techniques ✓
- clean and well-structured software architecture (based on metrics) ✓
- benchmarks show fairly and evenly distributed workload for 15.000 agents with $>$ 10.000 beliefs on regular desktop PCs ✓

$\Rightarrow$ fulfilled requirements stated in motivation

- fuzziness
- explicit repair-planning
- built-in concurrency and supporting components e.g. BLAS, crypto, ...
- optimisation with scoring function

- ReSTful API component to control agent with browser
- high-scalability support for cloud systems as optional component

TU Clausthal
Clausthal University of Technology

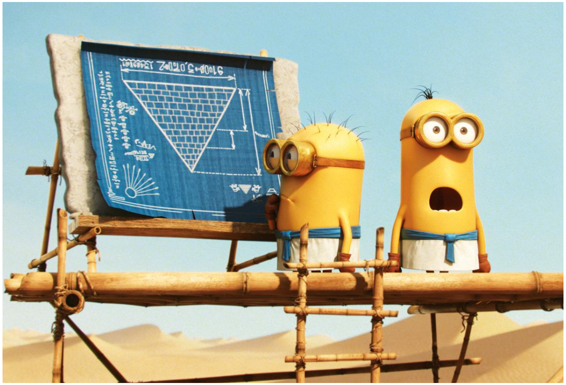
Thank You For Your Attention

Any questions?

Talk to us
or write an email

info@lightjason.org

Downloads & Publications on http://lightjason.org



http://www.mifus.de/out/pictures/master/product/2/27928.pt01.jpg

References

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007).
Programming multi-agent systems in AgentSpeak using Jason.
Wiley & Sons.

Kraus, P., Aschermann, M., and Müller, J. P. (2016).
LightJason: A BDI Framework Inspired by Jason.
IfI Technical Report Series IfI-16-04, Department of Informatics, Clausthal University of Technology.

PopularitY (2016).
http://pypl.github.io/, accessed: 2016-06-27 (archived by WebCite® at http://www.webcitation.org/6iZxjsbBs).

Rao, A. S. (1996).
AgentSpeak(L): BDI agents speak out in a logical computable language.
In Proc. of MAAMAW '96, pages 42–55, Secaucus, NJ, USA. Springer-Verlag New York, Inc.

RedMonk (2016).
http://redmonk.com/sogrady/2016/02/19/language-rankings, accessed: 2016-06-27 (archived by WebCite® at http://www.webcitation.org/6iZxPEb9K).

TIOBE (2016).
http://www.tiobe.com/tiobe_index, accessed: 2016-06-27 (archived by WebCite® at http://www.webcitation.org/6iZwpVq0y).