

Comparison of Behaviour-Based Architectures for a Collaborative Package Delivery Task

International Conference on Human-Agent Interaction (HAI)

Melanie Remmels

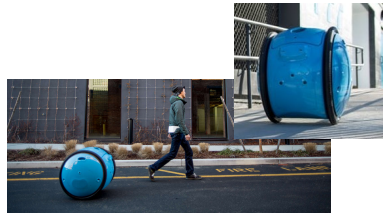
Nicolás Navarro-Guerrero and Stefan Wermter
Department of Informatics, Universität Hamburg

<https://nicolas-navarro-guerrero.github.io/>

18th October, 2017 – Bielefeld, Germany

Collaborative Package Delivery

General Motivation



The Concept



Taking advantage of complementary capabilities

- ▶ humans are dexterous manipulators and good high-level planners
- ▶ robots are strong and have endurance

Examples

- ▶ user holds the door open for the robot

The System

Part 1: The System

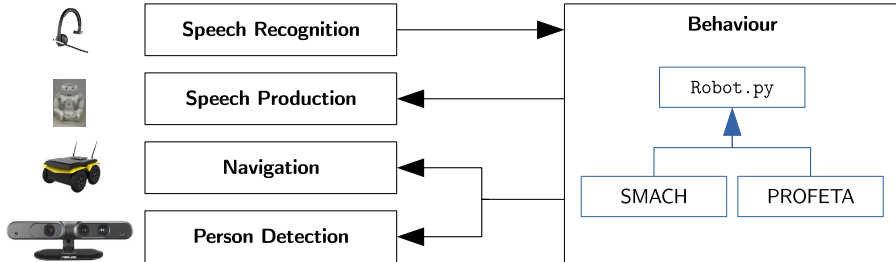


Delivery task

- ▶ Office environment
- ▶ Human-robot collaboration
- ▶ Non-specialised hardware
- ▶ Open-Source

Part 1: System Architecture

Twiefel et al., 2014



Part 1: Conclusions and Future Work

Language Recognition

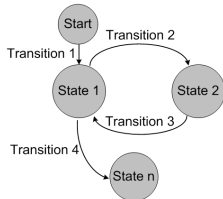
- ▶ Only one possible sentence for each command
 - ▶ Increase the number of possible sentences
 - ▶ Improvements of language understanding

Person Detection

- ▶ Person detection has problems when robot turns
 - ▶ Improve person identification capabilities

Comparison of Behavioural Architectures

Part 2: State Machine



- ▶ All possible states and transitions need to be modeled
- ▶ Sequence is repeatable

	SMACH
ROS Integration	Yes
Programming Language	Python
Documentation Status	available
Source Code available	Yes

Part 2: Belief-Desire-Intention (BDI)

- ▶ No record of previous transitions

	STRIPS	PDDL	PROFETA
ROS Integration	No	No	No
Programming Language	Lisp	PDDL	Python
Documentation Status	only publications	only publications	some examples
Source Code available	No	No	Yes

Part 2: State Machine vs BDI

- ▶ Theory for State machines
 - ▶ for small, well defined use cases ([Gongora and Irvine, 2010](#))
- ▶ Theory for BDI
 - ▶ for large use cases ([Gongora and Irvine, 2010](#))
 - ▶ Only the general concept has to be defined

Part 2: Research Question

- ▶ Why is an applied comparison needed?
 - ▶ Only theoretical, high-level comparisons exist
 - ▶ Behaviour does highly depend on the use case
- ▶ Is the package delivery a small or a large use case?
- ▶ Which Framework is easier to apply?

Part 2: Applied Comparisons

Possible Comparisons

- ▶ **Difficulty of description** (Mandes and Winker, 2016)
 - ▶ Static features e.g. Lines of code
- ▶ **Difficulty of creation** (Mandes and Winker, 2016)
 - ▶ Dynamic analyses e.g. Runtime
- ▶ **Difficulty of organization** (Mandes and Winker, 2016)
 - ▶ relevant for non-linear functionalities
- ▶ **Difficulty of appropriation** (Adam et al., 2017)
 - ▶ Check re-usability of models
- ▶ **Subjective Comparison**
 - ▶ e.g. Tutorials, Community

Part 2: Difficulty of Description

Implementation-Based Comparison

- ▶ Preprocessing
 - ▶ Put all code of one framework into one file
 - ▶ Removing comments and blank lines

	SMACH	PROFETA
Structure		
Number of characters	8465	5794
Classes		
Distribution of Class types	7 states	6 Beliefs, 10 Goals, 11 Actions, 1 Sensor
Lines of Code		
Total Number of Lines	252	188
Indentation		
Average nested block depth level	3	1
Maximum nested block depth level	11	6

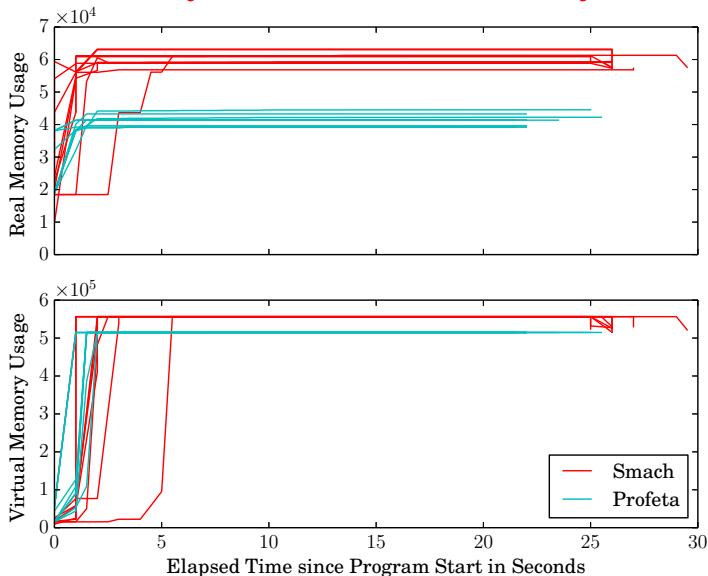
Part 2: Difficulty of Creation

Dynamic Comparison: Runtime

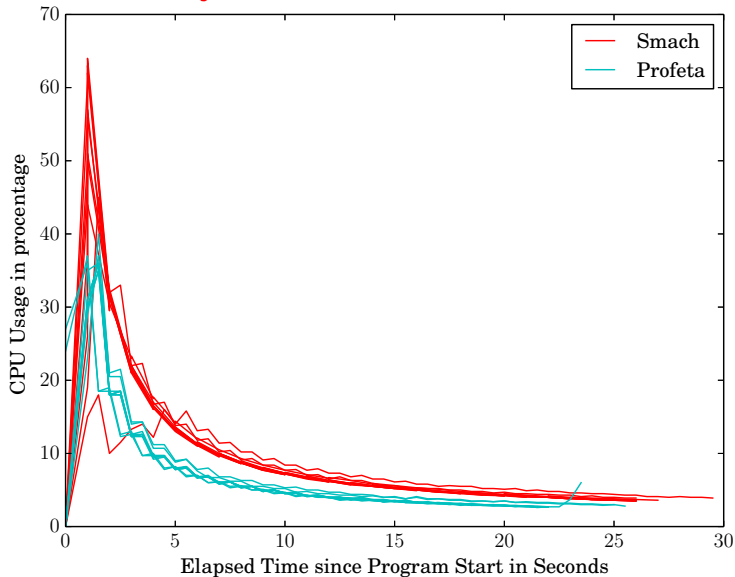
- ▶ Preprocessing
 - ▶ Simulating the transitions of the system
 - ▶ Predefined protocol for the interaction

	Average Time
SMACH	3.72 min
PROFETA	3.71 min

Part 2: Difficulty of Creation: Memory



Part 2: Difficulty of Creation: CPU



Part 2: Subjective Comparison

	SMACH	PROFETA
Repository		
Developers/Contributors	12	1
Number of Commits	148 (2010 - 2016)	38 (2012 - 2016)
Adapting the Code		
First entry	easy	medium
Modify an existing Use Case	hard	easy
Adapt an Use Case slightly	easy	hard
Number of Publications		
Publications describing the Framework	2	2
Authors which described the Framework	10	4
Publications using the Framework	5	1
Other		
Visualisation	available	None
Documentation	available	brief overview
Tutorials	available	only some examples
Community	connected to the ROS community	None

Part 2: Conclusions and Future Work

Which framework should be used?

- ▶ SMACH has an advantage in usability
- ▶ BDI is better for complex use cases, but need better syntax

Future Work

- ▶ Developer user study to quantify subjective parameters
- ▶ Identify small and large use cases
- ▶ Usability study
- ▶ Improvement to software modules

Reference List



Adam, C., Taillandier, P., and Dugdale, J. (2017). « Comparing Agent Architectures in Social Simulation: BDI Agents Versus Finite-State Machines ». In: *Hawaii International Conference on System Sciences (HICSS)*.
<http://hdl.handle.net/10125/41181> <http://hicss.hawaii.edu/program/>. Hilton Waikoloa Village, HI, USA: ScholarSpace, pp. 267–273 (cit. on p. 14).



Gongora, M. and Irvine, D. (2010). « Adaptive Intelligent Agents Based on Efficient Behaviour Differentiation Models ». In: *IEEE ANDESCON*. Bogota, Colombia: IEEE, pp. 1–6 (cit. on p. 12).



Mandes, A. and Winker, P. (2016). « Complexity and Model Comparison in Agent Based Modeling of Financial Markets ». *Journal of Economic Interaction and Coordination*, pp. 1–38 (cit. on p. 14).



Twiefel, J., Baumann, T., Heinrich, S., and Wermter, S. (2014). « Improving Domain-Independent Cloud-Based Speech Recognition with Domain-Dependent Phonetic Post-Processing ». In: *AAAI Conference on Artificial Intelligence*. Vol. Twenty-Eighth. Québec City, Québec, Canada: AAAI Press, pp. 1529–1535 (cit. on p. 7).