30 Years of **E**ngineering **M**ulti-**A**gent **S**ystems: What and Why **(EMAS)**

Michael Winikoff
AAMAS 2024
Auckland, New Zealand

These slides:
tinyurl.com/
AAMAS2024Thu9am

Note: bibliography at end

Agenda

- An overview of EMAS (Engineering Multi-Agent Systems)
    1. What
    2. Why
    3. Challenges and future directions
- Some EMAS research

These slides:
tinyurl.com/
AAMAS2024Thu9am

Agenda

- An overview of EMAS (Engineering Multi-Agent Systems)
    1. What
    2. Why
    3. Challenges and future directions
- Some EMAS research

Disclaimer …

These slides:
tinyurl.com/
AAMAS2024Thu9am

Agenda

- An overview of EMAS (Engineering Multi-Agent Systems)
    1. What
    2. Why
    3. Challenges and future directions
- Some EMAS research

Disclaimer …

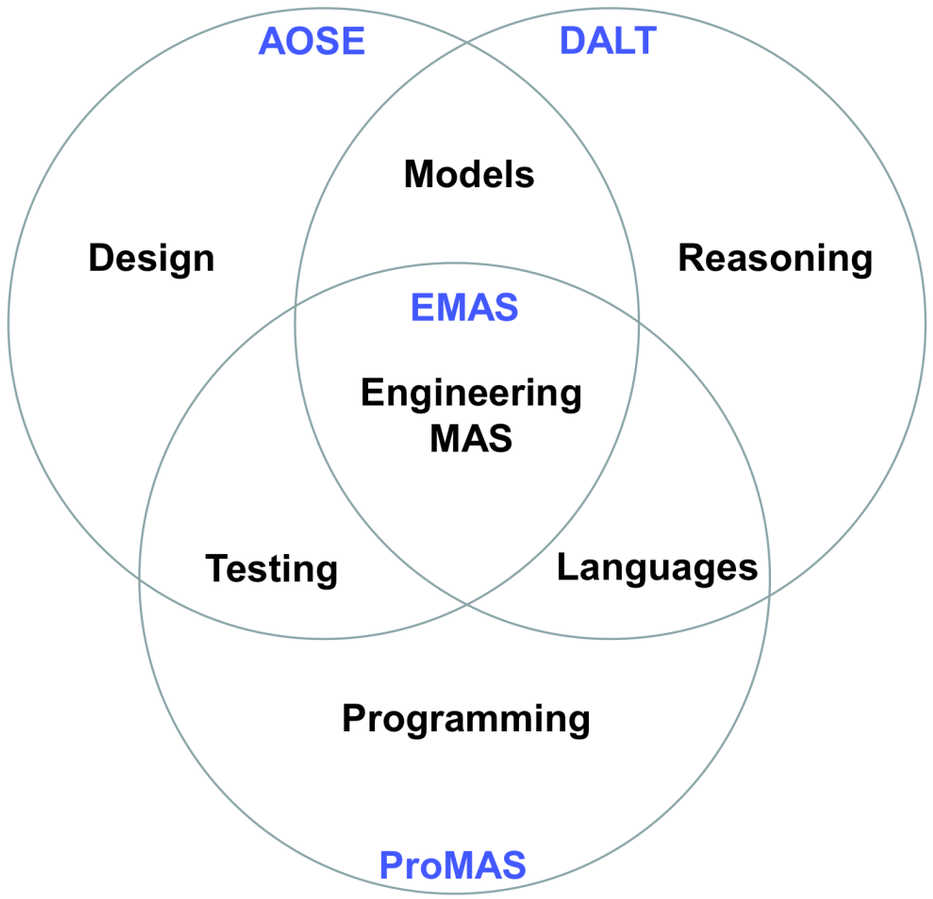
Thanks:
Lin Padgham

What is EMAS?

- "*Despite the substantial body of knowledge and expertise developed in the design and development of Multi-Agent Systems (MAS), the* ***systematic development of large-scale and open MAS*** *still poses many challenges*" (EMAS 2024 CFP, emphasis added)

  https://emas.in.tu-clausthal.de/2024/cfp/

- Research on Engineering Multi-Agent Systems is concerned with a range of topics that aim to provide software engineers with the **concepts**, **processes**, **notations**, **techniques**, **languages**, and **tools** to be able to effectively develop multi-agent systems.



EMAS 2014 keynote:
Twenty Years of Engineering MAS
(Hindriks, 2014)

EMAS Work …

- Less about algorithms/theories/theorems …
- … more about (human-oriented) models/processes/tools
- Importance (and challenge!) of artefacts (effort, sharing, incentives)

Why?

- **Why (multi-)agents?**

- Wide range of applications involving distribution, adaptability, flexibility, robustness, autonomy, complexity

- Domains include production scheduling, energy, transport, disaster relief, manufacturing, simulation, health, UAVs …

Benfield et al, 2006
Belecheanu et al, 2006
Munroe et al, 2006
Dignum & Dignum, 2010
Müller & Fischer, 2014
Singh & Padgham, 2015
Briola et al, 2023

- **Why Engineering?**

- Why not just write it Java?

Source: Singh & Padgham, 2015

Benefits?

"Based on this analysis, agent-oriented methodology, architecture, and development delivered a 368% improvement on overall project productivity. (2.11 FP Actual/0.45 FP Expected) …

In a **wide range of complex business applications**, we show that the use of **BDI** [belief-desire-intention] **technology** incorporated within an enterprise-level architecture can **improve** overall developer **productivity** by an average [of] 350%. For java coding alone, the increase in productivity was over" (Benfield et al., 2006; emphasis added)

- Paper by Agentis Software staff — company commercialising agent technology, industries include finance, insurance, logistics, energy, IT infrastructure.

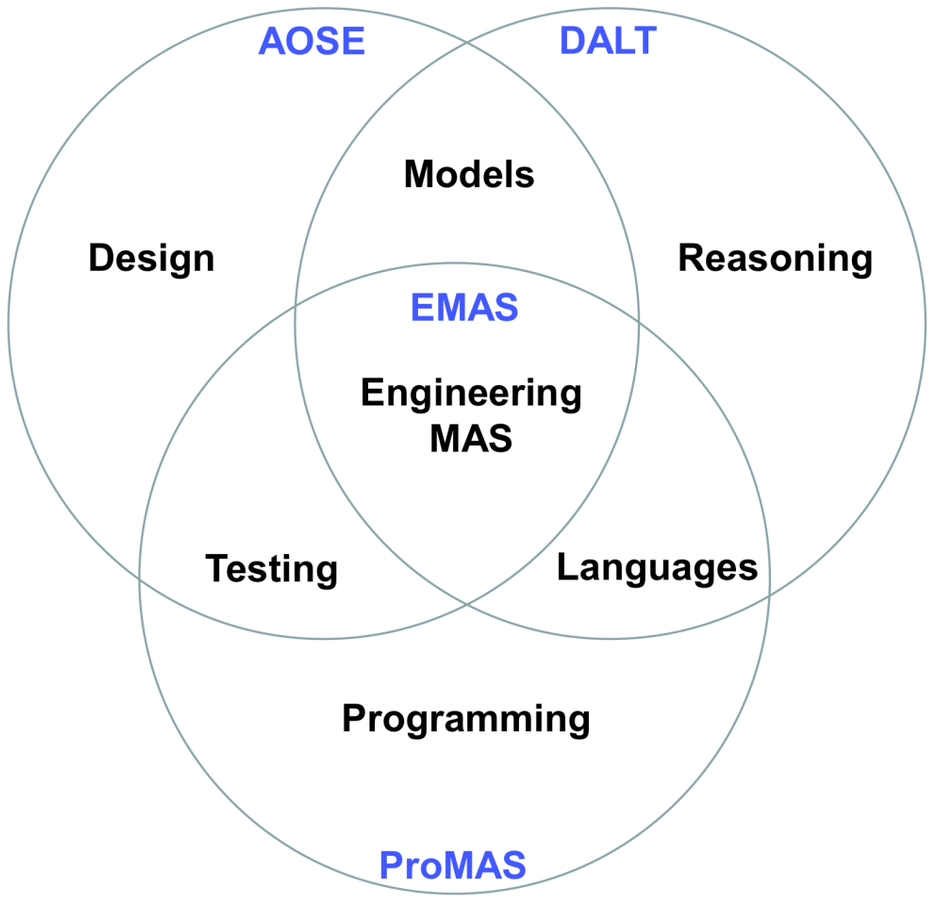
But: unverified claims by industry …

What is EMAS?

- "*Despite the substantial body of knowledge and expertise developed in the design and development of Multi-Agent Systems (MAS), the* ***systematic development of large-scale and open MAS*** *still poses many challenges*" (EMAS 2024 CFP, emphasis added)

  https://emas.in.tu-clausthal.de/2024/cfp/

- Research on Engineering Multi-Agent Systems is concerned with a range of topics that aim to provide software engineers with the **concepts**, **processes**, **notations**, **techniques**, **languages**, and **tools** to be able to effectively develop multi-agent systems.



EMAS 2014 keynote:
Twenty Years of Engineering MAS
(Hindriks, 2014)

Concepts

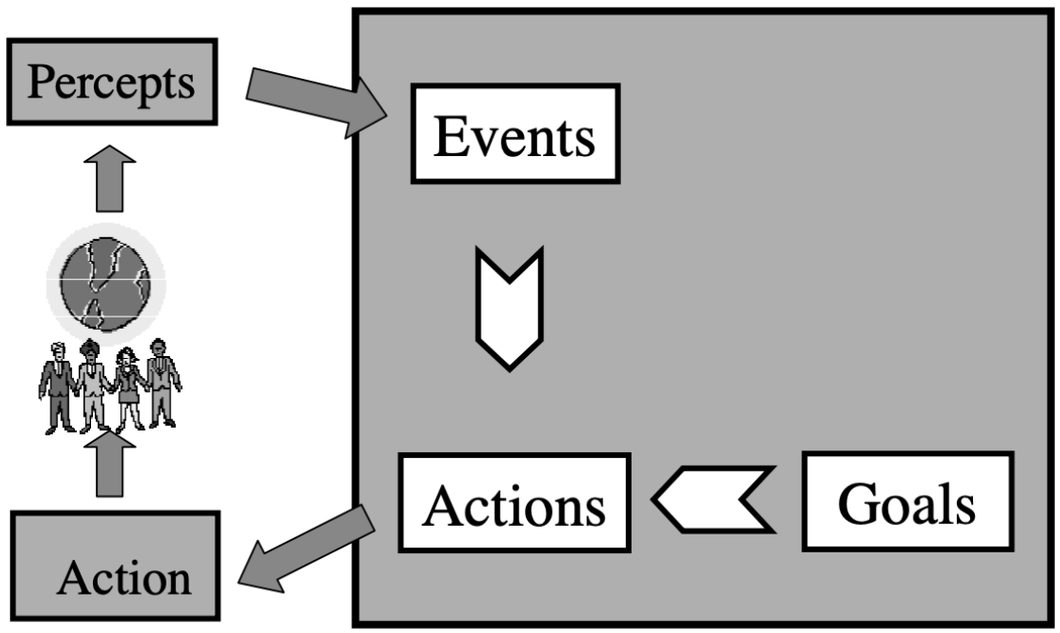
Agents are:

- *situated* - hence **actions** and **percepts**

- *reactive* - hence **events**

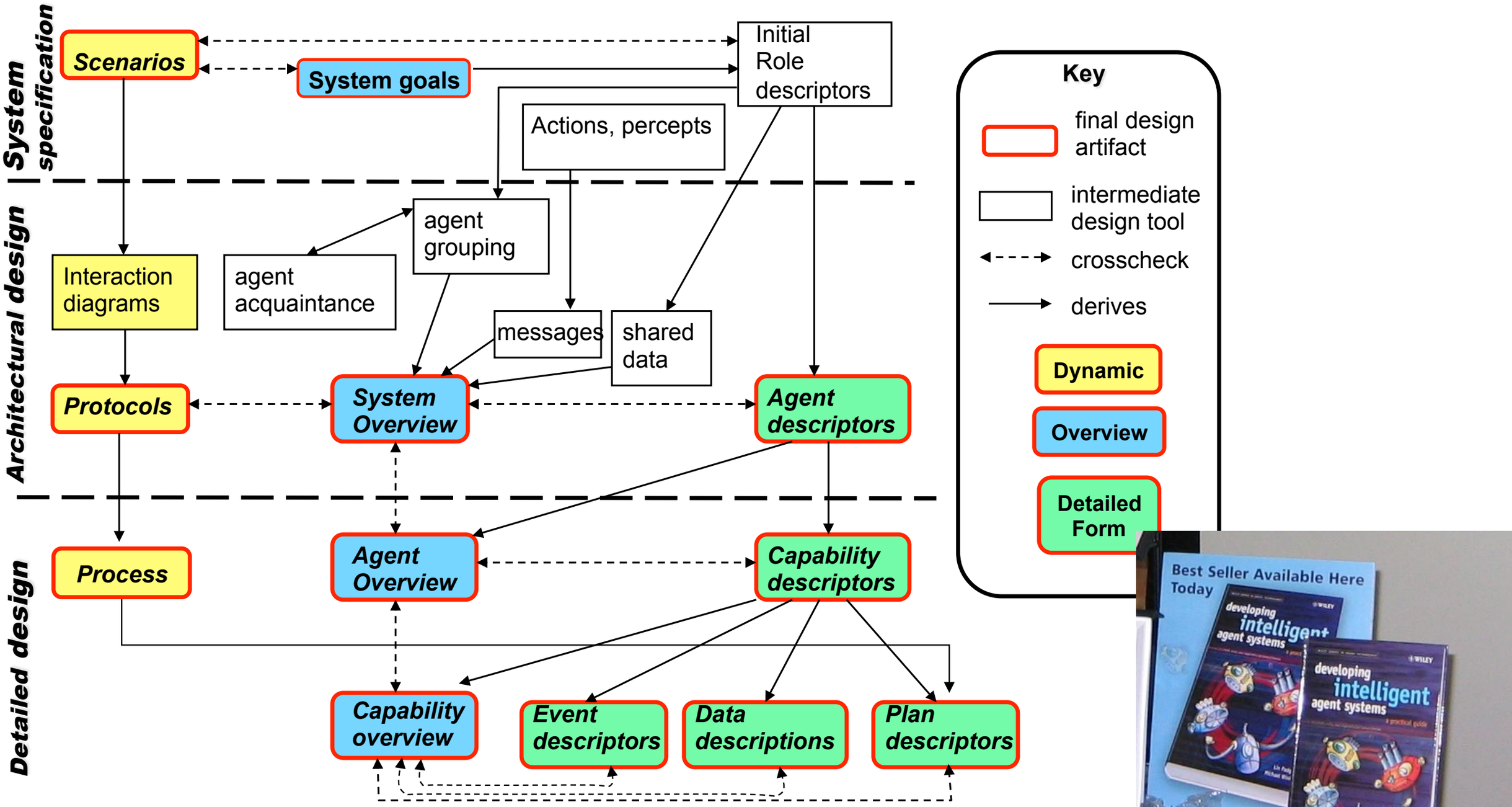
- *proactive* - hence **goals** (key concept)!

- *social* - hence **messages**, **commitments**, …



Source: Winikoff et al, 2001

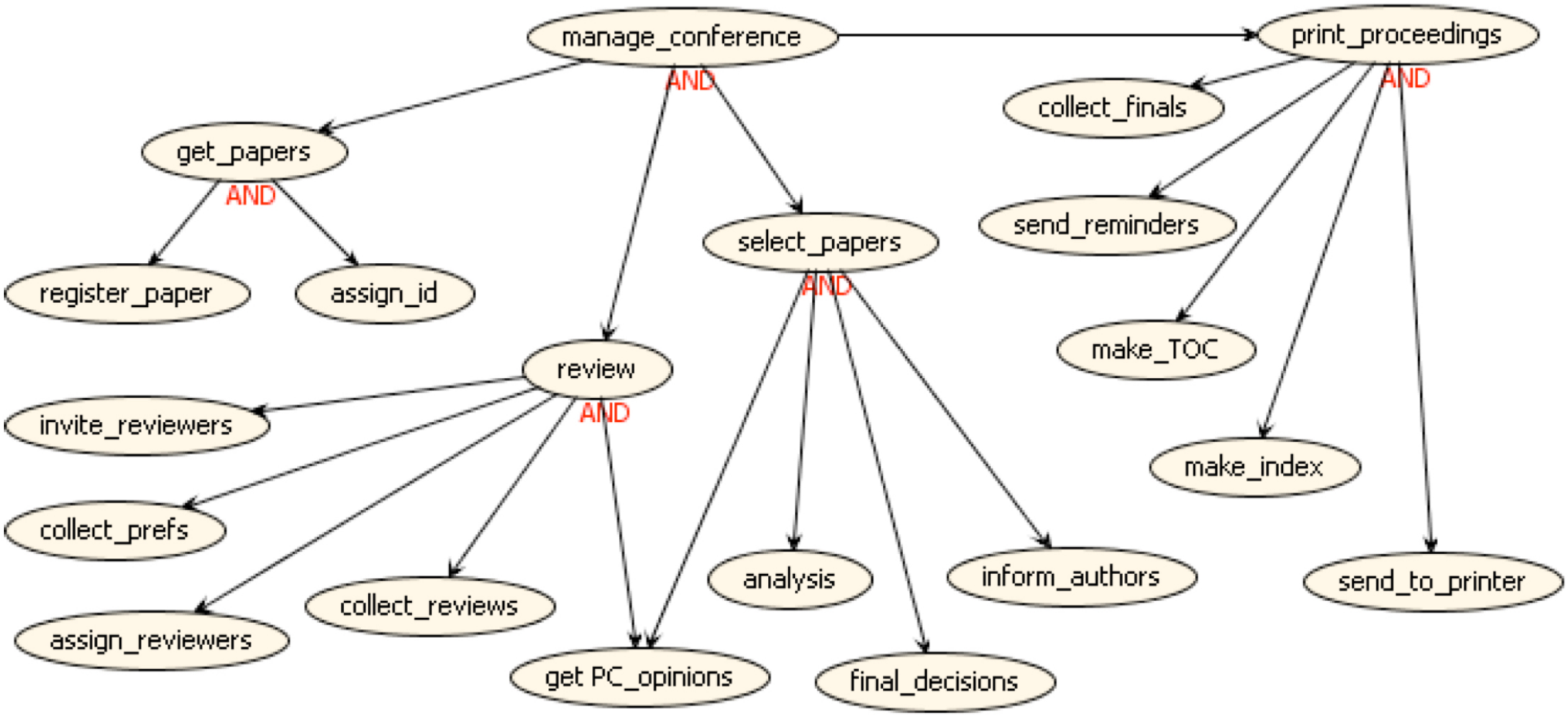
Process



Source: Padgham & Winikoff, 2004

Process (cont)

- …
- Implementation
- Assurance (testing, debugging, verification)
- Maintenance
- *Note: not sequential - iterative … (and Agile …)*



Source: Padgham et al, 2008

Edit Scenario - review scenario

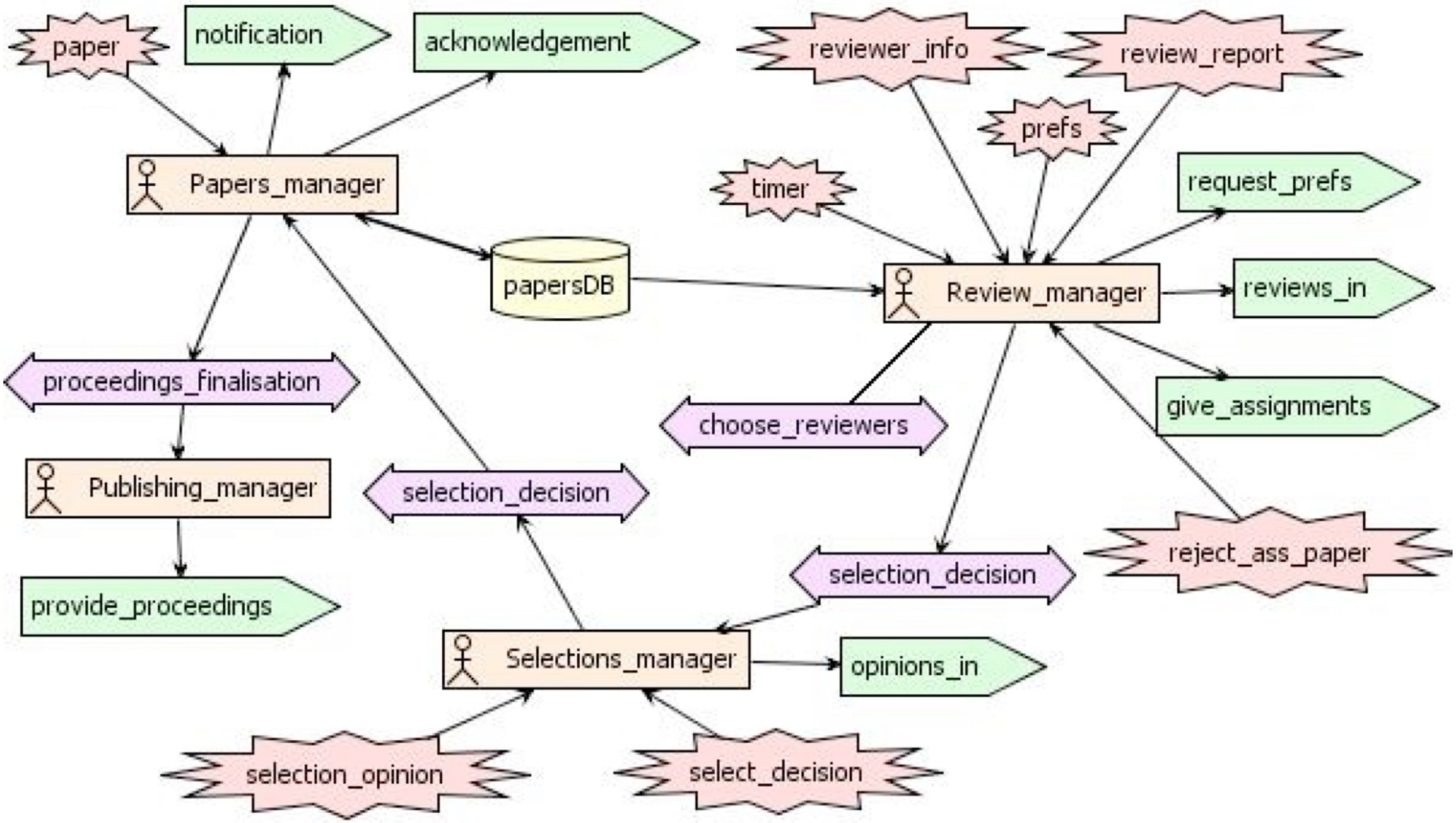
	Type	Name	Role	Data	Description
1	G	invite_reviewers	Review_management	ReviewerDB	Invite candidates to join the review panel
2	G	collect_prefs	Assignment	ReviewerDB	Collect the preference of the reviewers
3	G	assign_reviewers	Assignment	ReviewerDB	Assign papers to reviewers based on their prefe...
4	A	give_assignments	Assignment		Send the papers to the allocated reviewers
5	P	review_report	Review_management		Receive the review from the reviewers
6	G	collect_reviews	Review_management	ReviewDB	Collect all the reviews from the reviewers

Insert Step Edit Remove

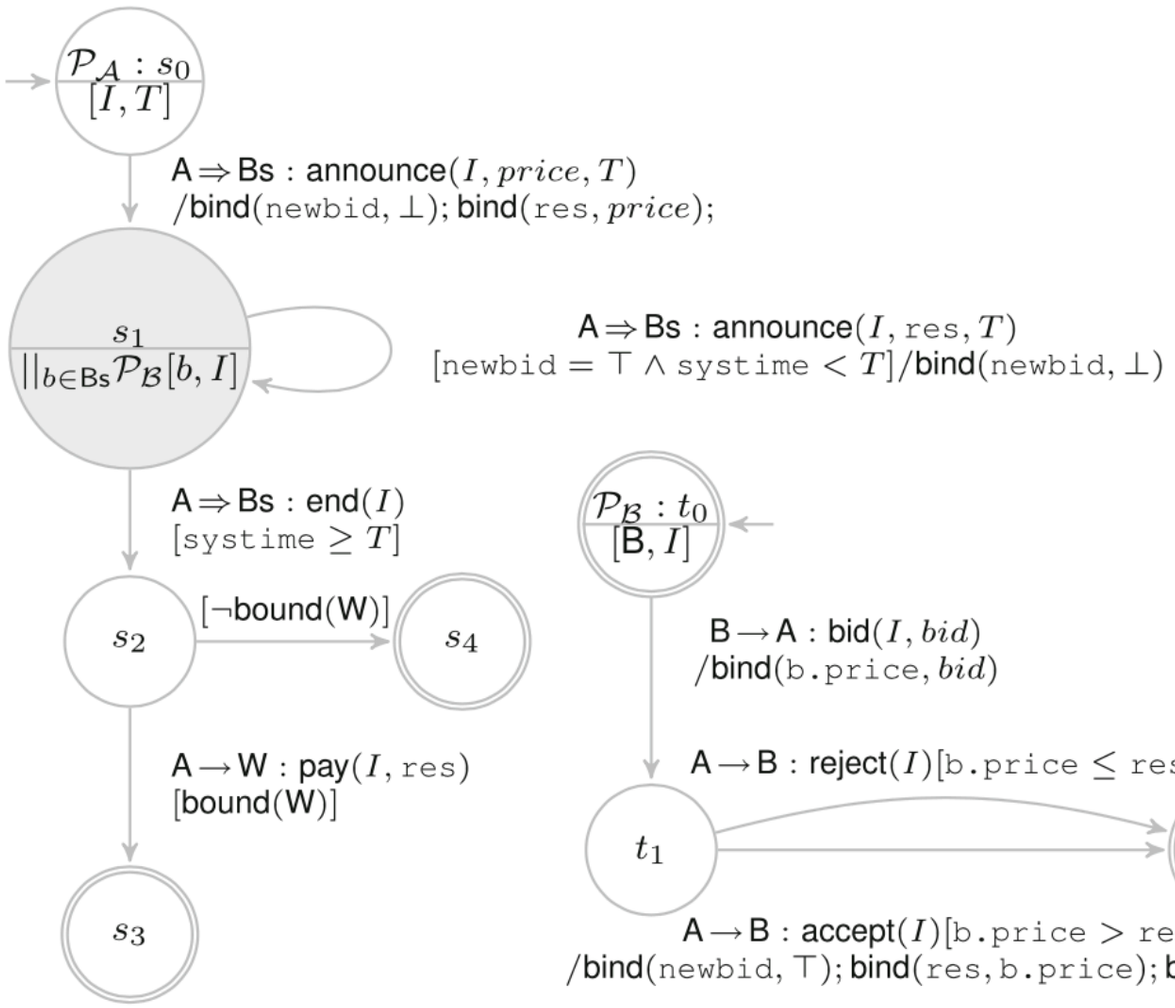
Close

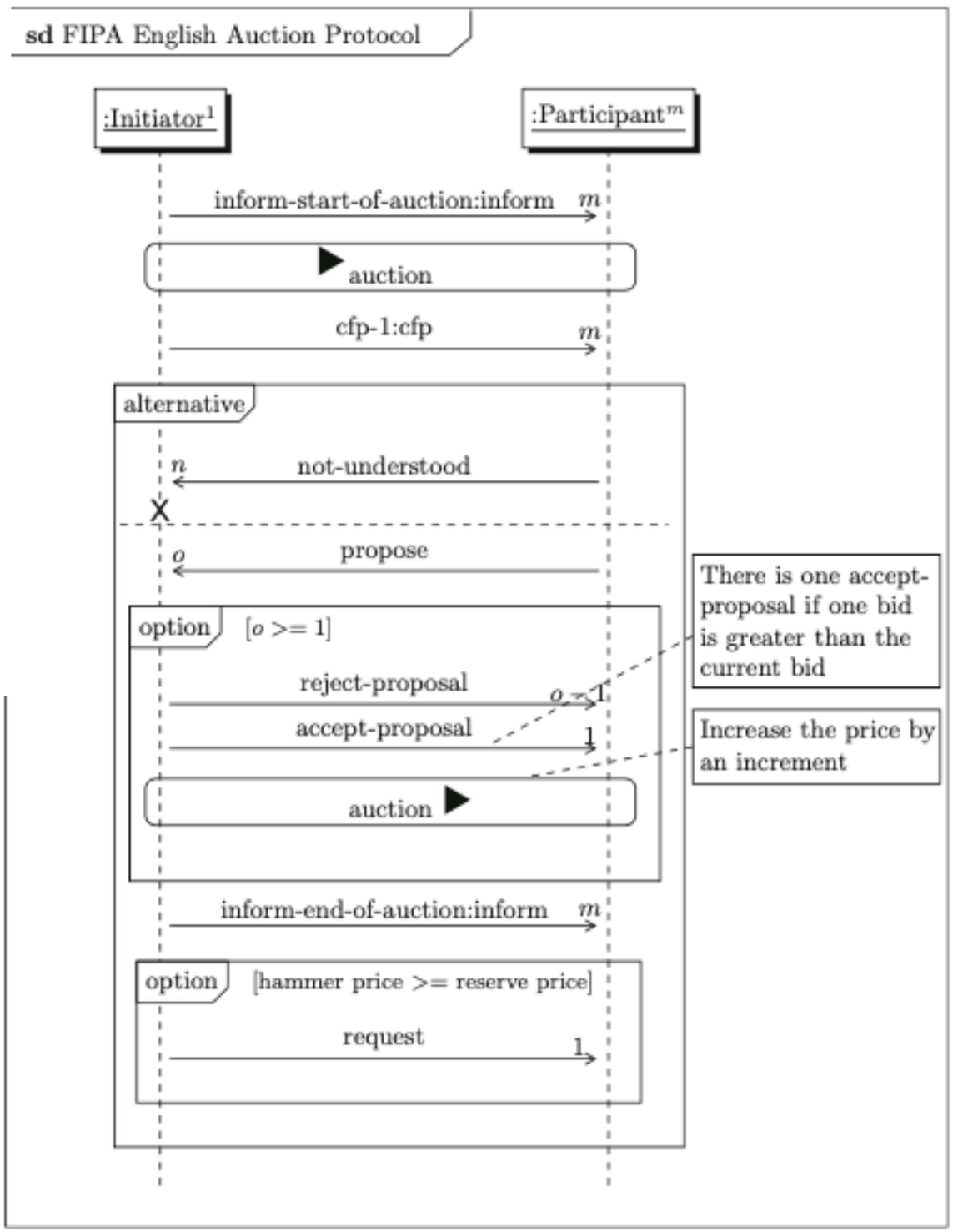
A -> Action G -> Goal O -> Others P -> Percept S -> Scenario

Source: Padgham et al, 2008



Source: Padgham et al, 2008



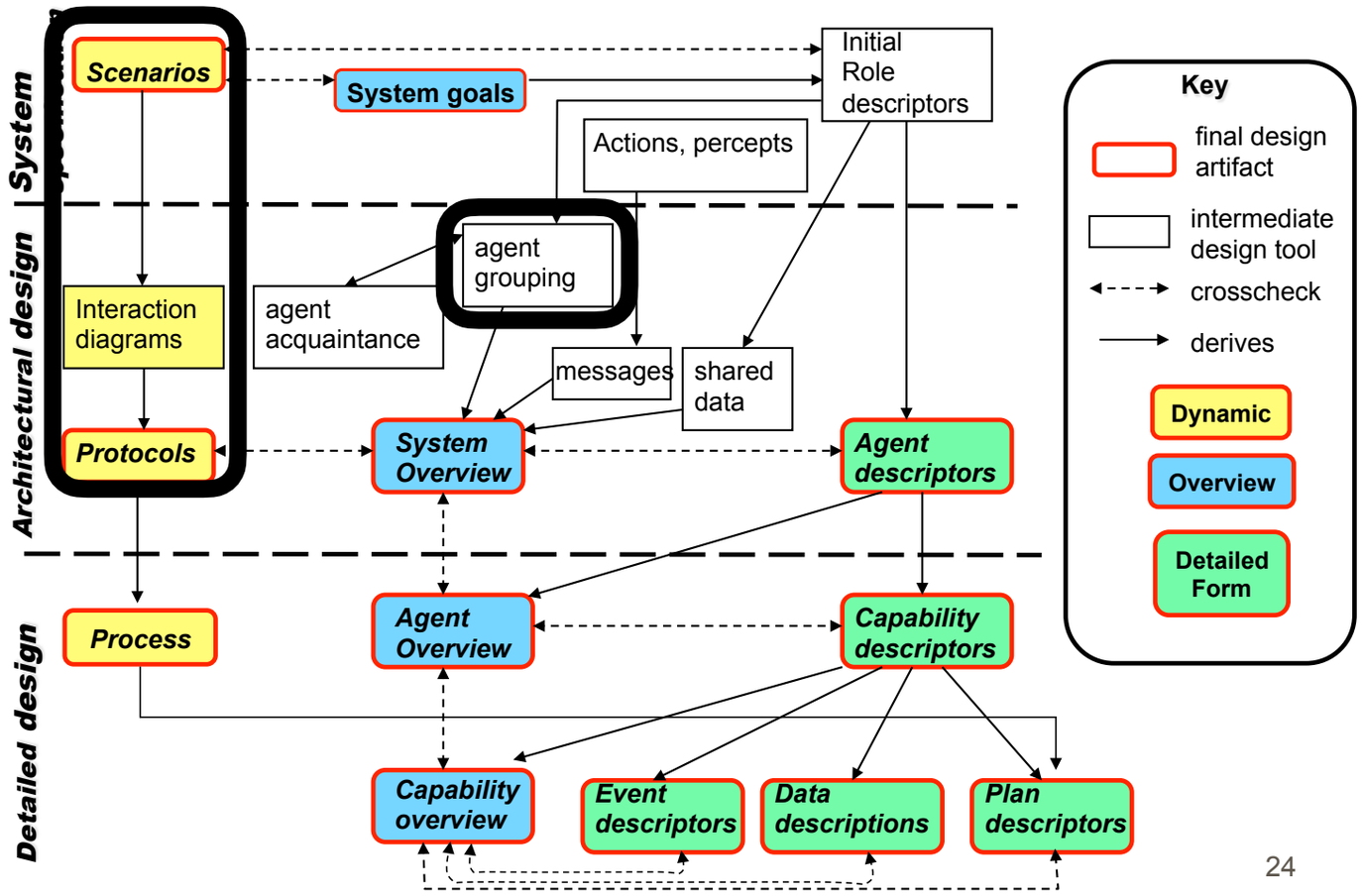


Source: Winikoff et al, 2018

Techniques

- Example: Define agent types by grouping smaller chunks, considering coupling and cohesion

- Example: Scenarios → Interaction Diagrams → Protocols



Languages

- ***Could have a whole talk just on this!***

- Many languages (e.g. 2APL, 3APL, AgentSpeak, Jason, JaCaMo [Jason+CArtAgO+Moise], Astra, Brahms, GOAL, GOLOG, ConGolog, IndiGolog, Gwendolen, JACK, JADE, Jadescript, Jadex, JIAC, MetateM, SARL …)

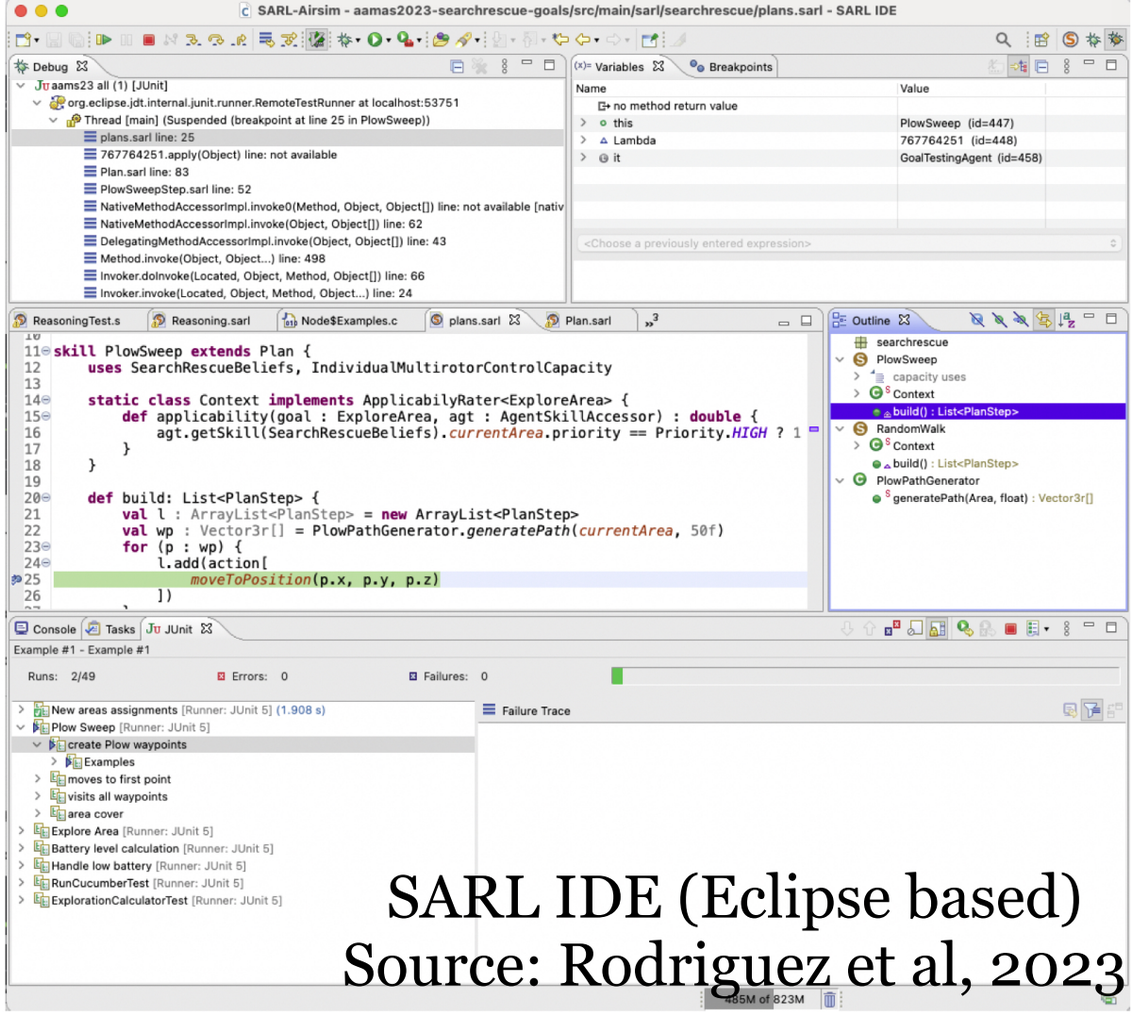
Bordini et al, 2005 & 2009
Bordini & Dix, 2016

- Will introduce BDI (Belief-Desire-Intention) and show an example AgentSpeak program later …

Bratman, 1987
Ingrand et al, 1992

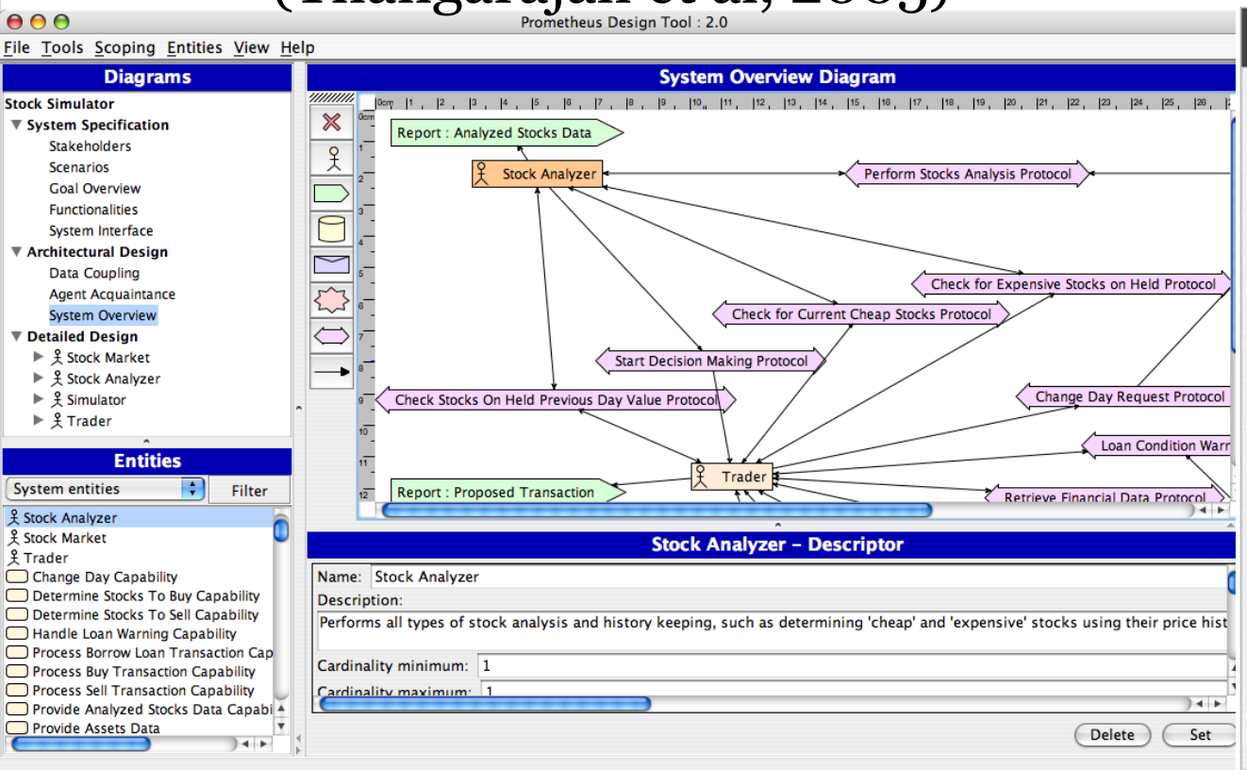
Tools

- Support design, coding, testing, debugging, …

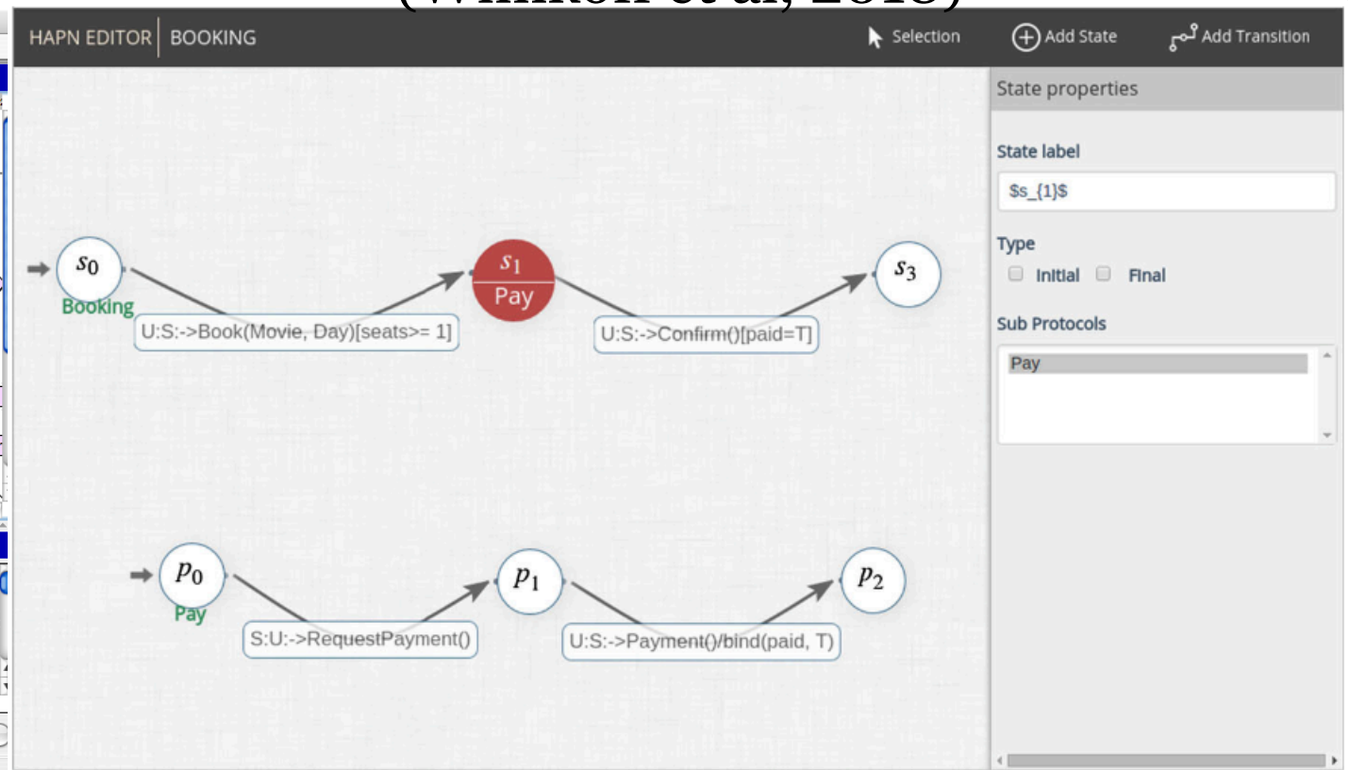


SARL IDE (Eclipse based)
Source: Rodriguez et al, 2023

Prometheus Design Tool (Thangarajah et al, 2005)



HAPN Protocol Editor (Winikoff et al, 2018)



Some EMAS Challenges

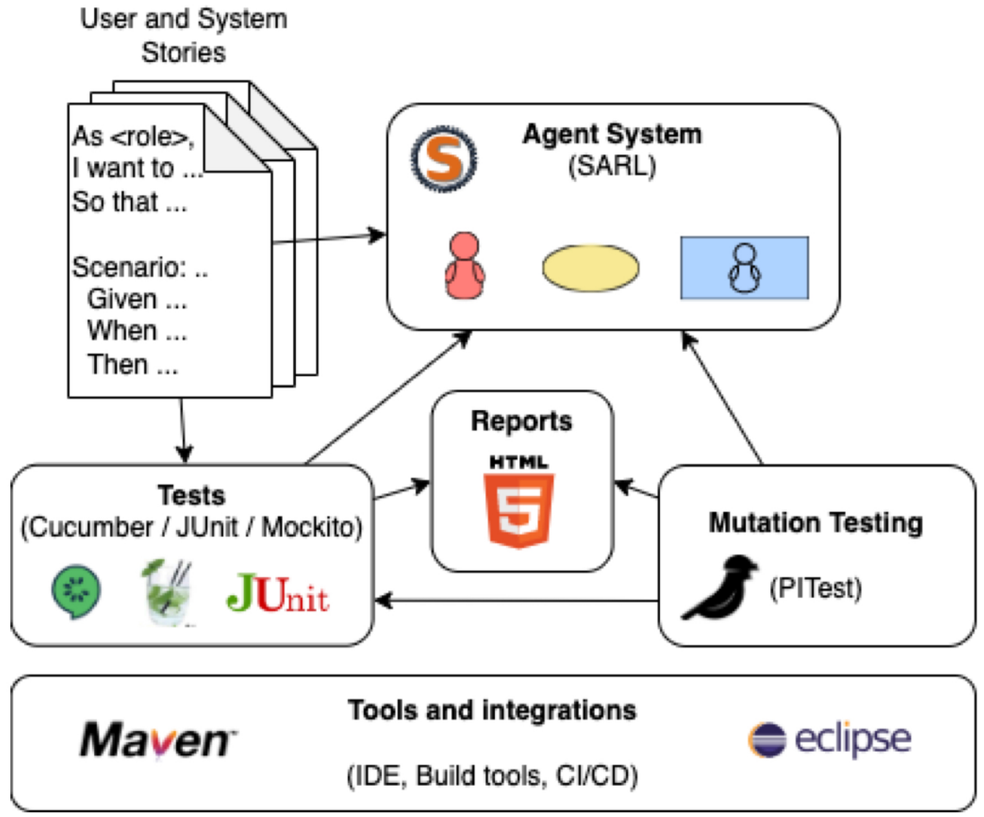
- Practicalities
- Community
- Research

Practicalities

- **Modernise** - the world has changed! (Agile, Python, DevOps, hypermedia…)

- **Integrate** (Eclipse, etc.)

- Why? Ease-of-use (familiar tools), easier adoption, performance, robustness, scalability …



Source: Rodriguez et al, 2023

Rodriguez EMAS 2024 keynote

Sebastian Rodriguez: Agile Approach for Agent Oriented Software Engineering

(Chair: Brian Logan)

Abstract: The agile software development life cycle is widely used in industry today due to its highly flexible and iterative processes that facilitate rapid prototyping. In this talk, we advocate to close the gap between mainstream software engineering and agent technology by adopting and adapting well-known and accepted techniques. We introduce an agile approach to capturing requirements in agent systems via user and system stories and how to translate these requirements into goal-oriented agent models. We cover how to define test cases to verify the expected system behaviour following a Behaviour-Driven Development (BDD) approach. We leverage a range of state-of-the-art development tools, inheriting the rich set of features they provide. Finally, we discuss future directions and opportunities.

Community

- **Link** with other relevant research and practice communities (including industry!)

- E.g. software engineering, internet/web of things, (social) AI, service, grid, autonomic … and AAMAS beyond EMAS!

- Quantify (and market) benefits and applicability (test case, competitions …)

(some) Research

- **Beyond BDI:** more powerful cognition (e.g. adding values, planning, …) - want simplicity, transparency and efficiency - difficult! (Logan, 2015)

- **What about Learning?** "Agent programming in the cognitive era" (Ricci, 2022; Bordini et al, 2020)

  - *The "Cognitive Hourglass": Agent Abstractions in the Large Models Era,* Ricci *et al.*, Blue Sky 2, Friday 2-4pm

- **Flexible interaction:** Commitment Machines (Yolum & Singh, 2002), BSPL (Singh, 2011; Chopra & Christie V, 2023)

- **Assurance** …

Assurance is important

"The verification of such complex software is **critical to its acceptance** by science mission managers." (Havelund et al., 2000)

"… validation through extensive tests was mandatory … However, the task proved challenging for several reasons. First, agent-based systems explore realms of **behaviour outside people's expectations and often yield surprises** …" (Munroe et al., 2006, section 3.7.2)

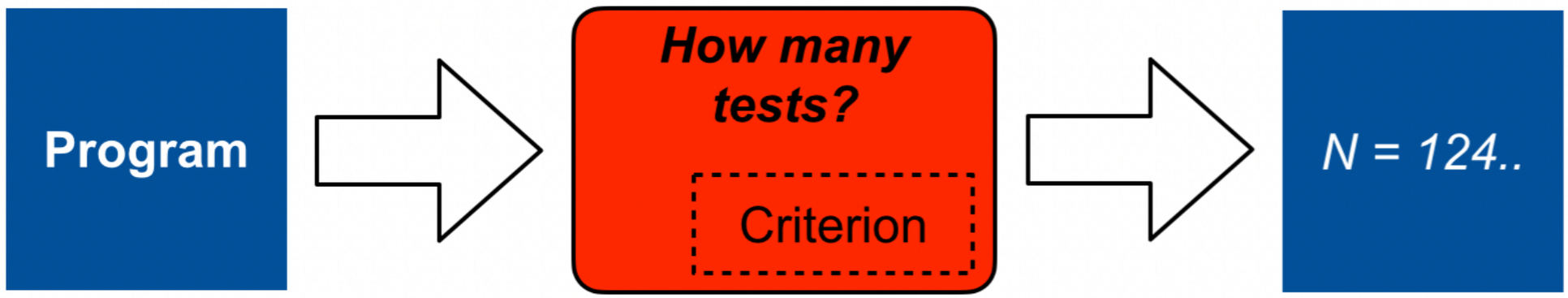
Assurance …

- How testable are BDI agents?
- How can I *verify* my MAS?
- How can I *debug* my MAS?
- *Explaining* my agent

How testable are agents?

Winikoff & Cranefield, 2014

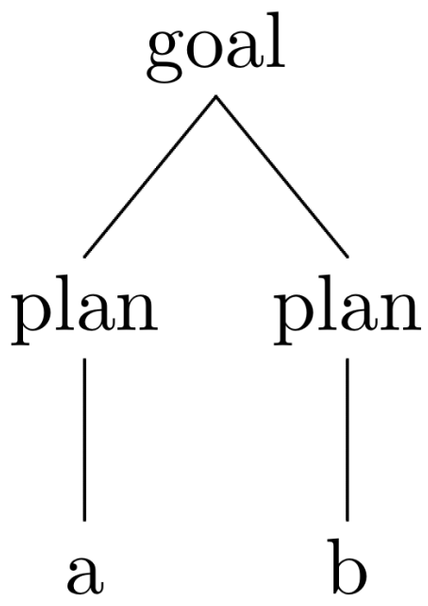
- Assurance important, traditionally done through testing
- But intuitively agents harder to test - is this true?
- Consider all paths testing criterion
- How many paths are there for a BDI* program?

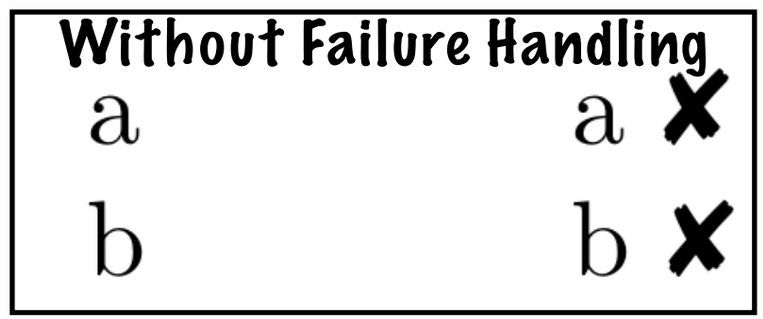


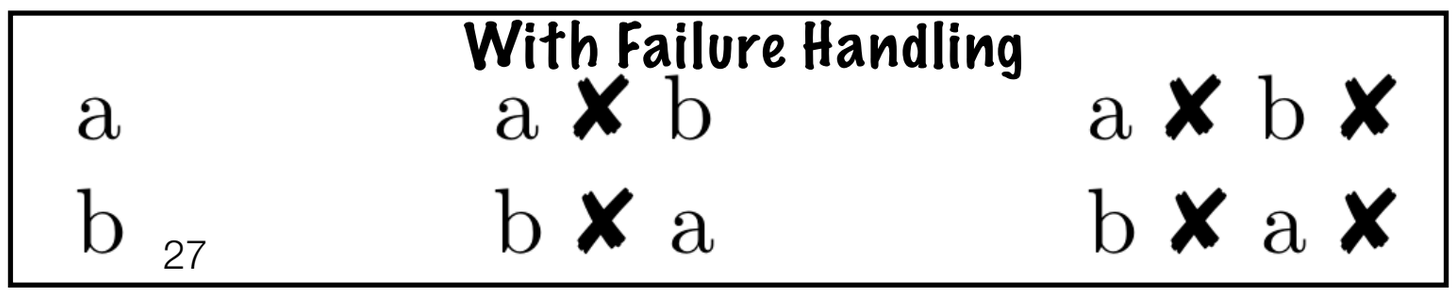
(*) BDI = Belief Desire Intention

BDI Goal-Plan Trees

- Goal: select one of its applicable plan instances
- Plan: execute all sub-goals (in order)
- Failure handling mechanism varies between BDI platforms. Common mechanism: when plan fails, re-post the goal, and select an unused plan instance







Parameters			Number of			No Failure Handling		With Failure Handling	
j	k	d	goals	plans	actions	$n^{✔}(g)$	$n^{✘}(g)$	$n^{✔}(g)$	$n^{✘}(g)$
2	2	3	21	42	62 (13)	128	614	6.33×10^{12}	1.82×10^{13}
3	3	3	91	273	363 (25)	1,594,323	6,337,425	1.02×10^{107}	2.56×10^{107}
2	3	4	259	518	776 (79)	1,099,511,627,776	6,523,509,472,174	1.82×10^{157}	7.23×10^{157}
3	4	3	157	471	627 (41)	10,460,353,203	41,754,963,603	3.13×10^{184}	7.82×10^{184}

- Astronomical numbers
- Failure handling makes a huge difference

With Failure Handling	
$n^{✔}(g)$	$n^{✘}(g)$
6.33×10^{12}	1.82×10^{13}
1.02×10^{107}	2.56×10^{107}
1.82×10^{157}	7.23×10^{157}
3.13×10^{184}	7.82×10^{184}

Reality check

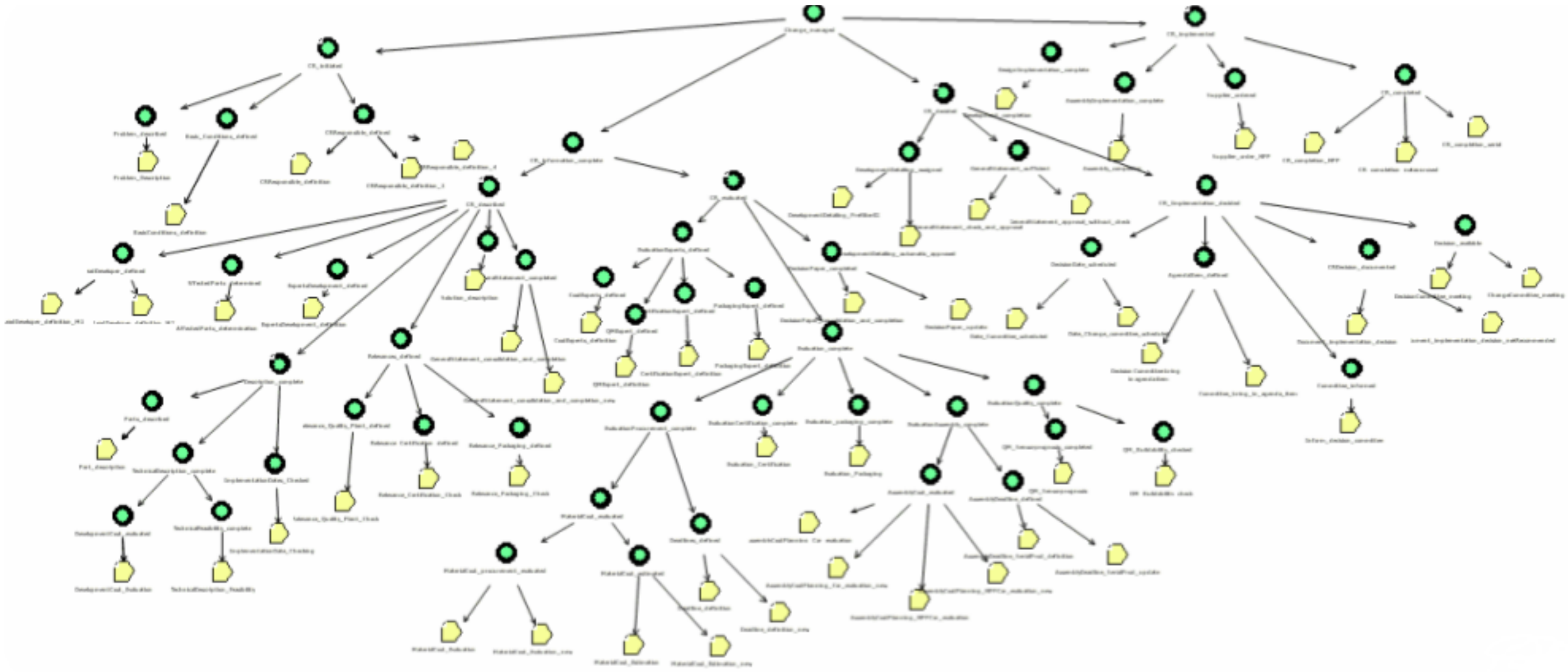


Figure 11: Goal-plan tree from the work of Burmeister et al. (2008, Figure 6) (reproduced with permission from IFAAMAS)

Parameters			Number of			No Failure Handling		With Failure Handling		
j	k	d	goals	plans	actions	$n^{✔}(g)$	$n^{✘}(g)$	$n^{✔}(g)$	$n^{✘}(g)$	$n(m)$
2	2	3	21	42	62 (13)	128	614	6.33×10^{12}	1.82×10^{13}	6,973,568,802
3	3	3	91	273	363 (25)	1,594,323	6,337,425	1.02×10^{107}	2.56×10^{107}	5.39×10^{57}
2	3	4	259	518	776 (79)	1,099,511,627,776	6,523,509,472,174	1.82×10^{157}	7.23×10^{157}	2.5×10^{123}
3	4	3	157	471	627 (41)	10,460,353,203	41,754,963,603	3.13×10^{184}	7.82×10^{184}	5.23×10^{99}
Workflow with 57 goals(*)						294,912	3,250,604	2.98×10^{20}	9.69×10^{20}	$(\ell = 4)$
(*) The paper says 60 goals, but their						294,912	1,625,302	6.28×10^{15}	8.96×10^{15}	$(\ell = 2)$
figure 6 actually has 57 goals.						294,912	812,651	9.66×10^{11}	6.27×10^{11}	$(\ell = 1)$

Abstract tree (j,k=2-3, d=3)

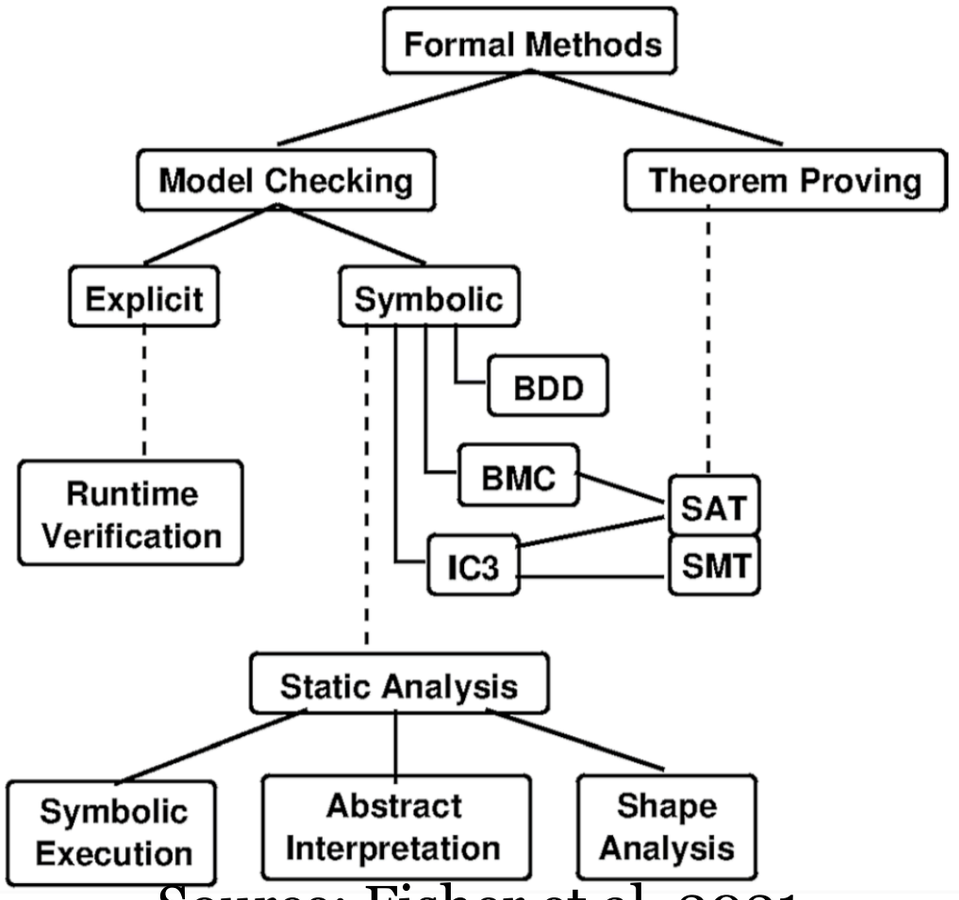
Real industry application

With Failure Handling

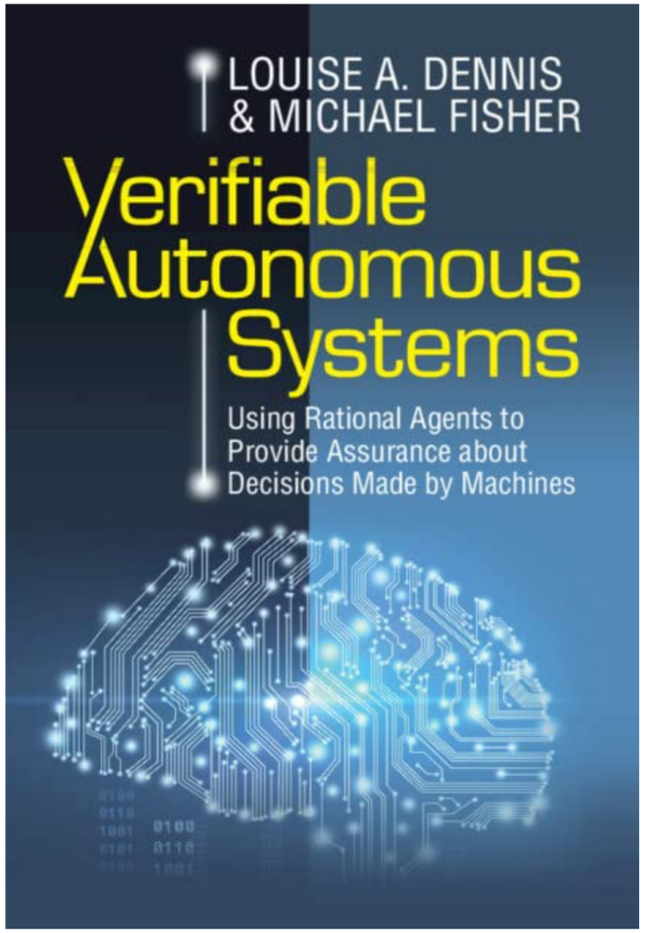
$n^{✔}(g)$	$n^{✘}(g)$	$n(m)$
6.33×10^{12}	1.82×10^{13}	6,973,568,802
1.02×10^{107}	2.56×10^{107}	5.39×10^{57}
1.82×10^{157}	7.23×10^{157}	2.5×10^{123}
3.13×10^{184}	7.82×10^{184}	5.23×10^{99}
2.98×10^{20}	9.69×10^{20}	$(\ell = 4)$
6.28×10^{15}	8.96×10^{15}	$(\ell = 2)$
9.66×10^{11}	6.27×10^{11}	$(\ell = 1)$

How can I *verify* my MAS?

- Range of techniques …
- Formal verification of agent programs (Dennis & Fisher, 2023)
- One key challenge: identifying & capturing requirements! (Next slide)



Source: Fisher et al, 2021



Initial process

Fisher et al, 2021

- Consider the **human context** (licensing, assumed human capabilities, laws/regulation, the interface) to **identify requirements**

    - Licensing: physical capabilities, domain knowledge, regulatory knowledge, ethical normalisation

        - Capabilities: general knowledge, common sense, physical capabilities, maturity/life-experience, ethics/values (e.g. self-preservation)

- Consider **how to assess requirements for an autonomous system**; e.g. assess behaviour in scenarios, rather than exam-based assessment of domain knowledge

How can I *debug* my MAS?

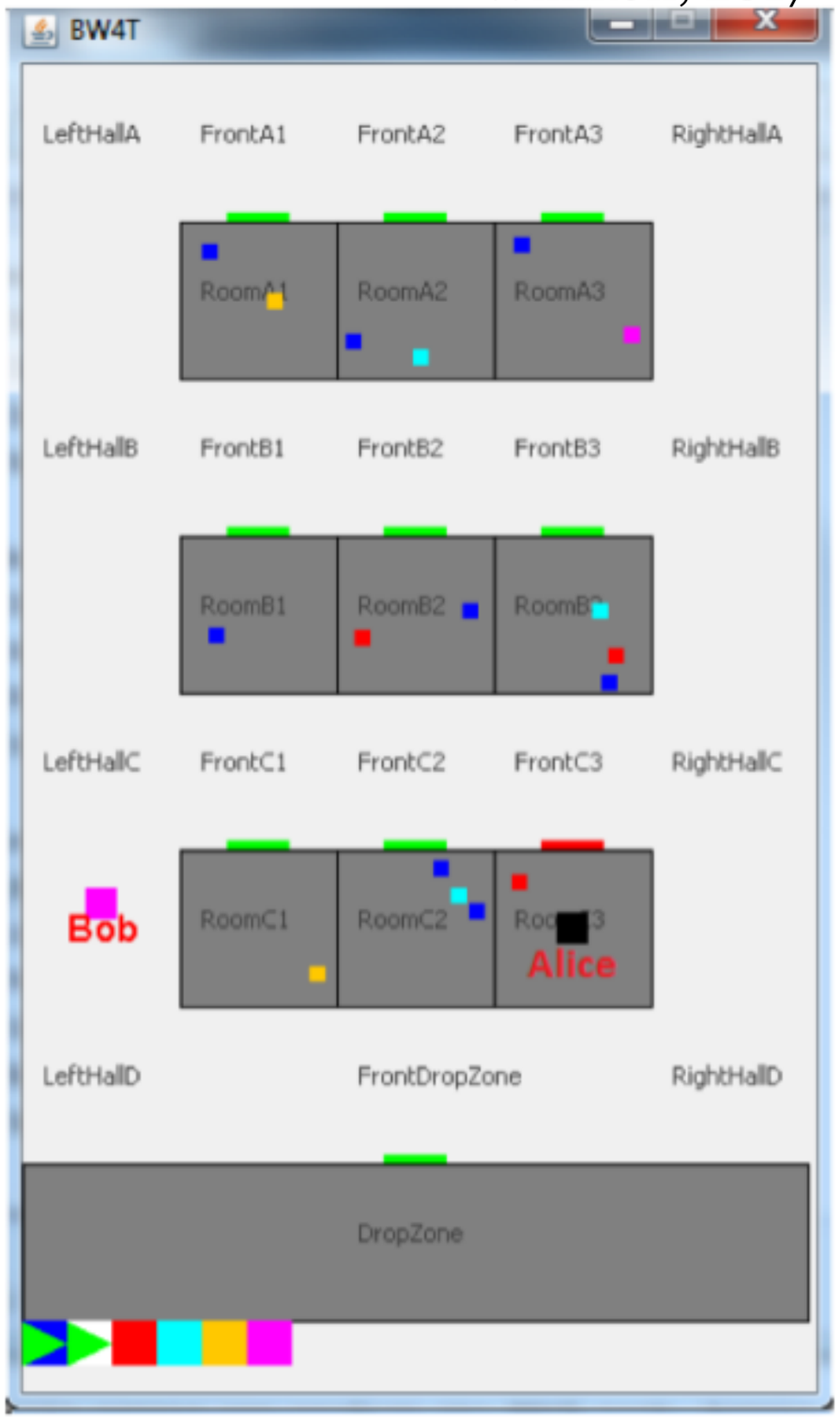
Winikoff, 2017

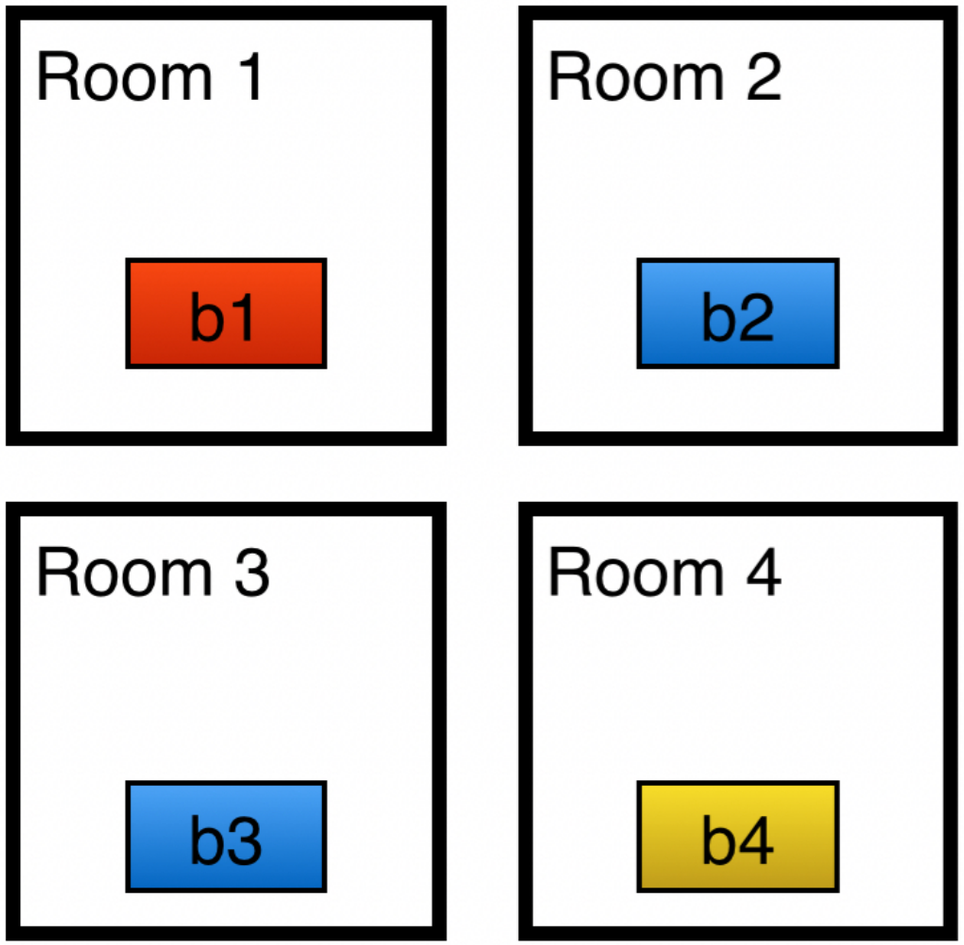
- Agent programs hard to debug: parallelism, complex control flow

- Key idea: use interactive questioning (“Why?”) (Hindriks 2012, Ko & Myers 2008)

- Example: Blocks World for Teams (BW4T)

- Complementary to algorithmic debugging (Ahlbrecht, 2023)





Goal: 1 2 3

```
 1  goto(room1)
 2  gotoBlock(b1) % red
 3  pickUp
 4  goto(dropzone)
 5  putDown % now need yellow ...
 6  goto(room2) % exploring ...
 7  goto(room3)
 8  goto(room4)
 9  gotoBlock(b4) % yellow
10  pickUp
11  goto(dropzone)
12  putDown % now need blue ...
13  gotoBlock(b3) % blue
14  pickUp
15  call(deliver)
16  gotoBlock(b2) % blue
17  pickUp % FAILED (because already holding b3)
18  goto(dropzone)
19  putDown % now done
```

+!deliver : nextColour(done) ← +done. *% If done then stop*
% select a block of the right colour and go get and deliver it
+!deliver : colour(B, C) ∧ nextColour(C) ∧ ¬holding(B) ← gotoBlock(B) ; pickUp ; !deliver.
+!deliver : holding(B) ∧ colour(B,C) ∧ nextColour(C) ← goto(dropzone) ; putDown ; !deliver.
% if holding a block that is not the next colour required then put it down (this may occur if e.g. someone else delivers a block, so the next colour changes)
+!deliver : holding(B) ∧ colour(B,C) ∧ ¬nextColour(C) ← putDown ; !deliver.
% if I know of a place that I've not yet visited then go there (explore)
+!deliver : place(P) ∧ ¬beenthere(P) ← goto(P) ; !deliver.

Debugging with Questions

Q1: why did pickUp at 17?

A1: because (i) !deliver at 15, (ii) plan π_2 has true context condition of "***colour(b2,blue) & nextColour(blue) & ~holding(b2)***", and (iii) the step gotoBlock(b2) succeeded

Q2: why believe colour(b2,blue) & nextColour(blue) & ~holding(b2) at 15?

A2: because putDown done at 12, and colour(b2,blue) & ***~holding(b2)*** was true

Q3: why believe ~holding(b2) at 12?

A3: that's been true since the beginning of execution ...

11 goto(dropzone)

Q3 12 putDown

13 gotoBlock(b3) *% blue*

14 pickUp

Q2 15 call(deliver)

16 gotoBlock(b2) *% blue*

Q1 17 pickUp *% FAILED*

...

π_2 = +!deliver :
colour(B, C)
& nextColour(C)
& ~holding(B)
← gotoBlock(B) ;
pickUp ; !deliver.

Questions and Answers

- Why (not) do S?
- Why (not) believe C?
- Why did step S succeed/ fail?
- Why was plan P selected?
- Prior step S_n
- Prior belief C_n
- No Applicable Plan
- “Inevitable”
- “Actually, no”
- AND, OR

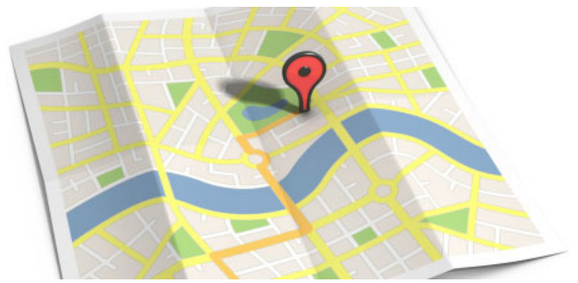
How to answer “Why?”

Principle: answer with prior point in execution where a different option would have led to a different outcome.

- Begin with closest explanation point
- Repeated "Why?" finds earlier explanations …

Explaining my agent

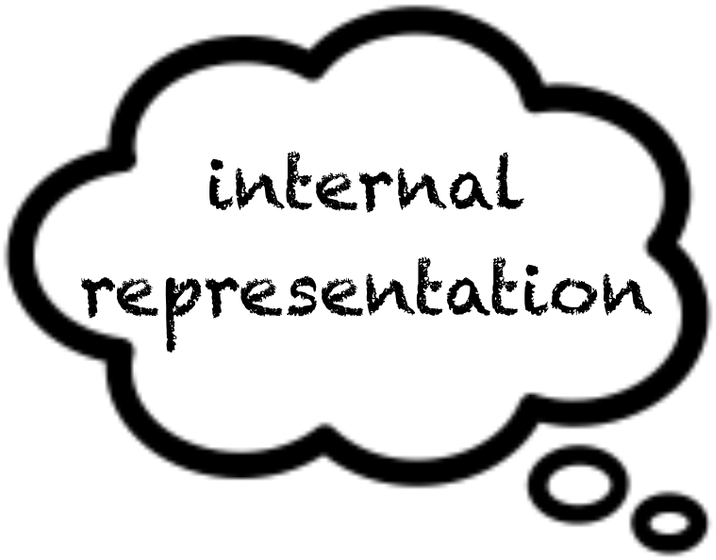
- Explain why a decision was made (or course of action taken)

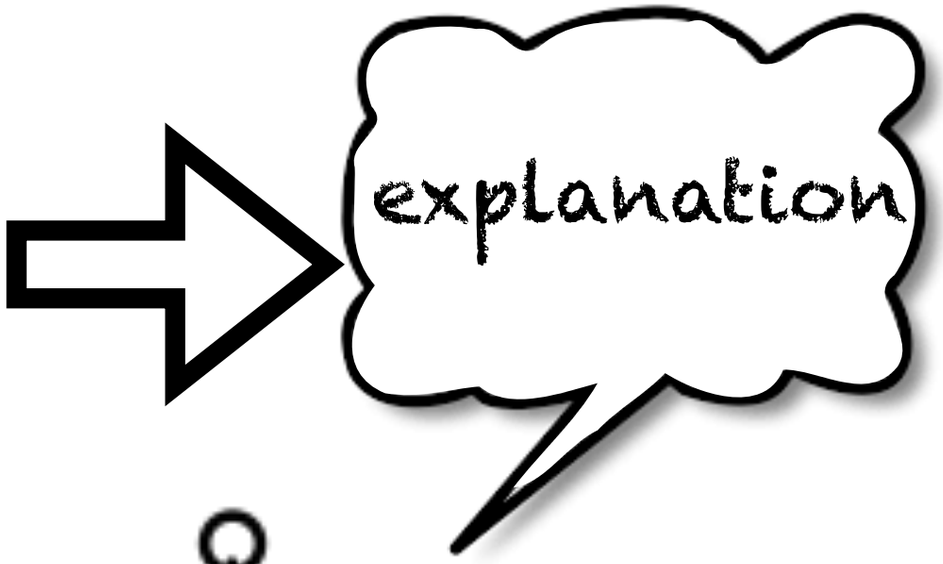


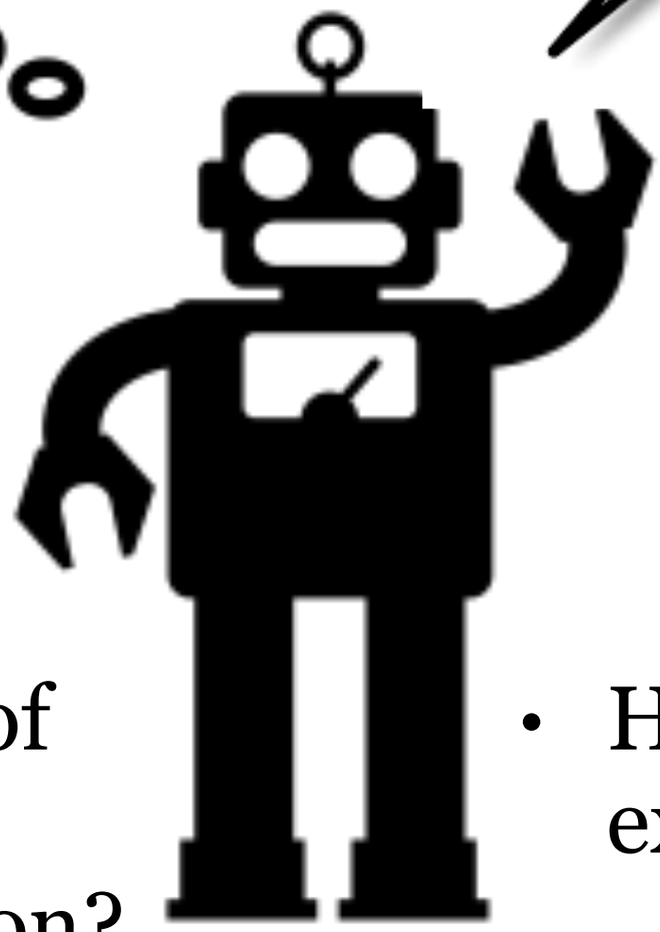
- Important for trust, especially when system finds a non-obvious solution, and for operating in a social world

***"An ability of a system to <u>explain</u> its advice was thought to be its most important attribute**. Second in importance was the ability of a system to understand and update its own knowledge base. …" (Teach & Shortliffe, 1981, emphasis added)*

"… for instance … for users of care or domestic robots a why-did-you-do-that button which, when pressed, causes the robot to explain the action it just took" (IEEE, Ethically Aligned Design v1, p20)





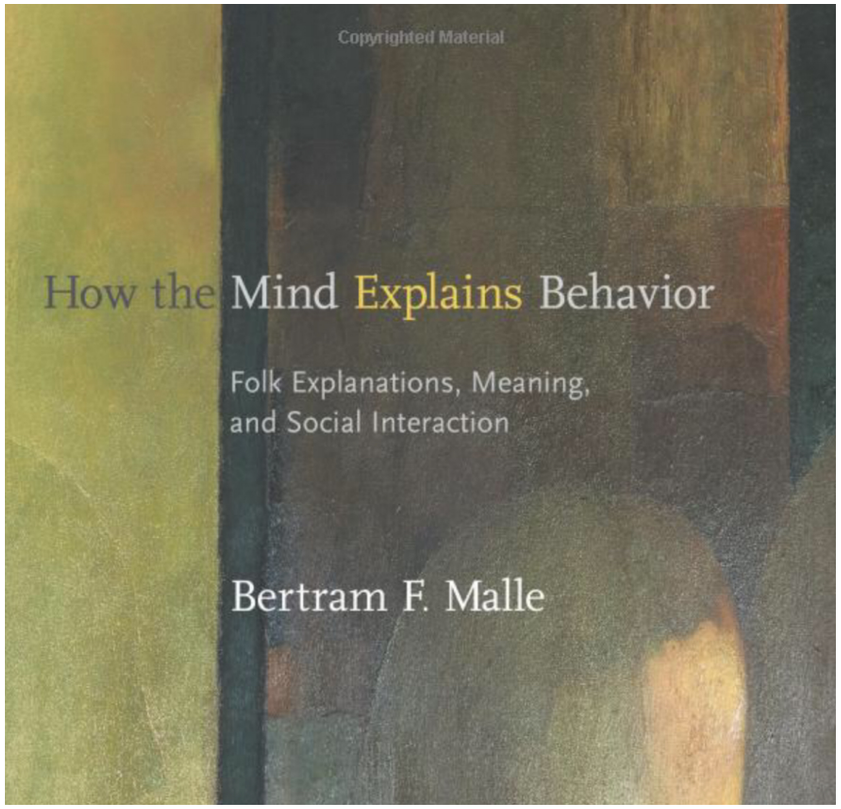


- What form of question?
  … explanation?
- What form of internal representation?

- How generate explanations?
- What difference do they make?

- Human explanations use *folk psychological constructs*, specifically Beliefs, Desires, Valuings (Malle, 2004)

- BDI agents use beliefs, desires …



Key claim

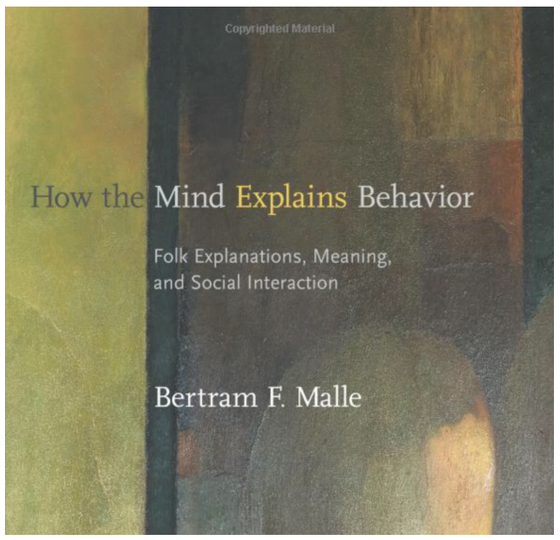
Winikoff et al, 2021

Cognitive Agents are **explainable**, since they use similar concepts to those humans use in explanation.

Desires,
Beliefs,
Valuings

explanation

Why?



ELSEVIER

Artificial Intelligence

Volume 267, February 2019, Pages 1-38

Explanation in artificial intelligence: Insights from the social sciences

Tim Miller ✉

Show more

https://doi.org/10.1016/j.artint.2018.07.007

Get rights and content

"I had money, Ann was not in her office, I preferred shop coffee to the kitchen (because it's better quality coffee), I needed to go to the shop in order to do getCoffee(shop), and I desired to getShopCoffee"

Summary

- EMAS - What? Concepts, notations, processes, techniques, tools, and languages to effectively develop MAS

- EMAS - Why?

- Some challenges: practicalities, motivation, community

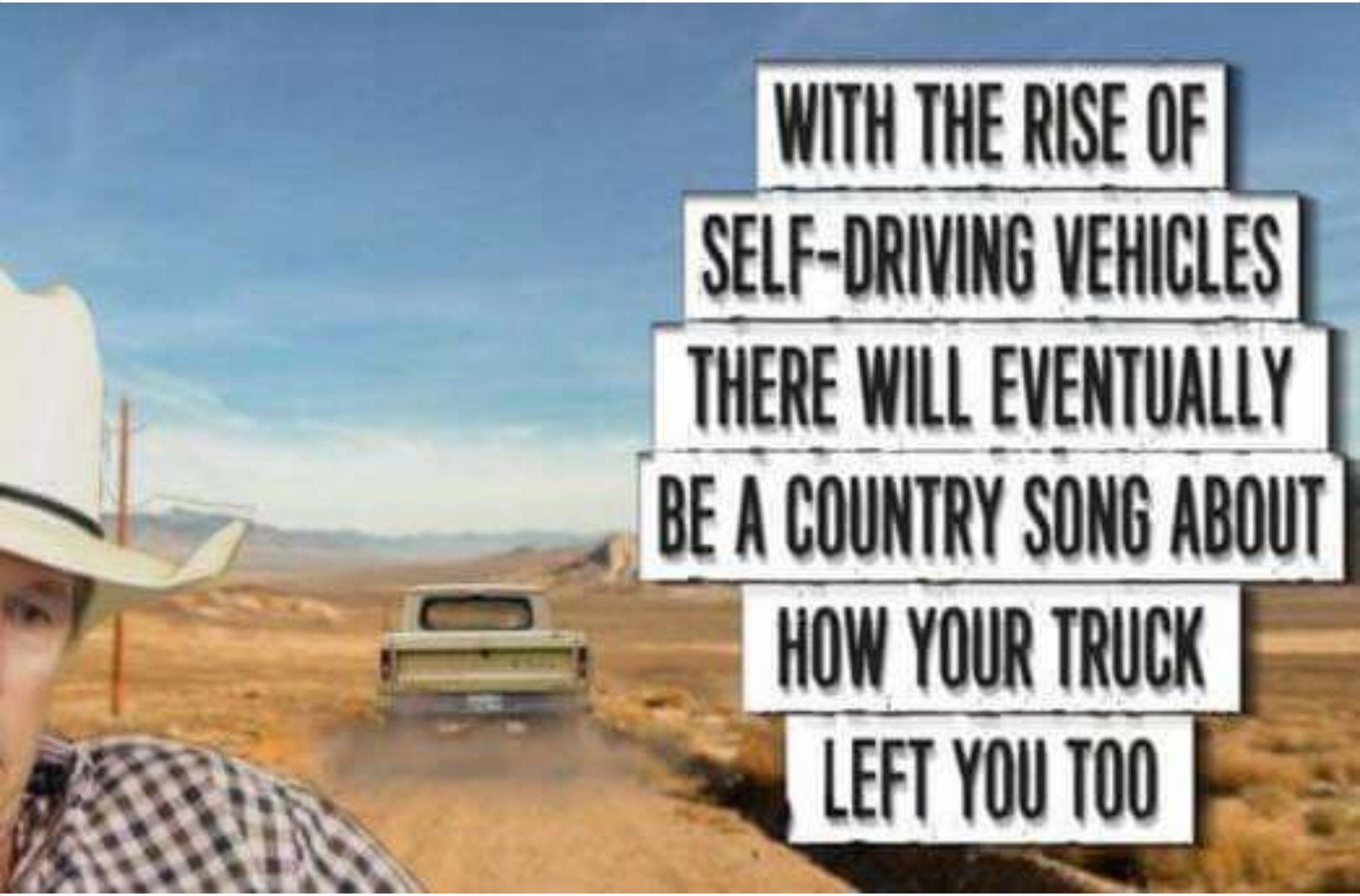
- Research: beyond BDI; what about learning?; flexible interactions; assurance (testing, debugging, verification) … and explanation

Thank you!

- Abdullah Al-Amin (s)
- Aditya Ghose (AU)
- Akın Günay (s*)
- Alex Richter (NZ)
- Alexander Egyed (AT)
- Alexander Reder (AT)
- Aloys Mbala (AU)
- Amir Aryani (s)
- Andrew Hodgson (AOS, AU)
- Andrew Lucas (AOS, AU)
- Angelo Ferrando (s*,IT)
- Ayelet Cohen (NZ)
- Bastin Tony Roy Savarimuthu (NZ)
- Beatriz López (ES)
- Bernd-Holger Schlingloff (DE)
- Brent Martin (NZ)
- Carles Sierra (ES)
- Cathal Doyle (NZ)
- Christoph Matthaei (NZ)
- Christopher Cheong (s,AU)
- David Poutakidis (s)
- David Pym (UK)
- Dhirendra Singh (AU)
- Duc Q. Pham (s)
- Ed Kazmierczak (AU)
- Frank Dignum (~~NL~~SE)
- Galina Sidorenko (~~NZ~~SE)
- Gaya Buddhinath Jayatilleke (s)
- Guannan (“Peter”) Li (s)
- Hanno-Felix Wagner (s*)
- Harald Søndergaard (AU)
- Heinz Schmidt (AU)
- Hywel Lloyd (BPAC, NZ)
- Ian D. Peake (AU)
- Ian Mathieson (AU)
- Islam Elgadawy (s)
- James Harland (AU)
- James Harland (AU)
- Jamie Curmi (AOS, AU)
- Jane Li (NZ)
- Jason Khallouf (s)
- Jenny McDonald (NZ)
- Jocelyn Cranefield (NZ)
- John Thangarajah (s,AU)
- Joshua Hutchison (s)
- Joy R. Rudland (NZ)
- Judith Swan (NZ)
- Julija Sardelić (NZ)
- June Tordoff (NZ)
- Khanh Hoa Dam (s,AU)
- Klaus Fischer (DE)
- Kristin Yvonne Rozier (US)
- Lavindra Priyalal de Silva (s,UK)
- Leon Sterling (AU)
- Lin Padgham (AU)
- Louise Dennis (UK)
- M. Birna van Riemsdijk (NL)
- Mal Gorman (BoM, AU)
- Manjula Devananda (s)
- Margaret Hamilton (AU)
- Mariusz Nowostawski (~~NZ~~ NO)
- Maryam Purvis (NZ)
- Massimo Cossentino (FR/IT)
- Mehdi Dastani (NL)
- Michael Fisher (UK)
- Minjie Hu (s)
- Myrthe Tielman (NL)
- Neil Yorke-Smith (NL)
- Nitin Yadav (AU)
- Omer Rana (UK)
- Paul Maisano (AOS, AU)
- Peter Vlugter (NZ)
- Phil Blyth (NZ)
- Philip Dart (AU)
- Pınar Yolum (TRNL)
- Rafael Bordini (UK, BR)
- Rainer Unland (DE)
- Ralph Rönnquist (AOS, AU)
- Richard Zeng (NZ)
- Rob Wass (NZ)
- Roberto E. Lopez-Herrejon (AT)
- Roger Jarquin (Jade, NZ)
- Sandy Dance (BoM, AU)
- Sarah Rennie (NZ)
- Scott DeLoach (US)
- Sebastian Rodriguez (AU)
- Sharmila Savarimuthu (NZ)
- Stephen Cranefield (NZ)
- Stephen Duffull (NZ)
- Swee Kin Loke (NZ)
- Tatjana Lutovac (s*)
- Thomas Juan (s*)
- Thomas Young (s*)
- Tim Miller (AU)
- Toan Phung (s)
- Tobias Ahlbrecht (s*)
- Virginia Dignum (~~NL~~SE)
- Viviana Mascardi (IT)
- Wamberto Vasconcelos (UK)
- Wei Liu (AU)
- Yoosef Abushark (s*)
- Zahir Tari (AU)
- Zhiyong Zhang (s*)

s = student
s* = not my student
XX (e.g. AU) = country

AOS/BoM/BPAC/JADE = industry


WITH THE RISE OF
SELF-DRIVING VEHICLES
THERE WILL EVENTUALLY
BE A COUNTRY SONG ABOUT
HOW YOUR TRUCK
LEFT YOU TOO

https://youtu.be/ESNDtH6suUo

https://youtu.be/mxhcQgvBD2Y

Bibliography (1/2)

Tobias Ahlbrecht. 2023. An algorithmic debugging approach for belief-desire-intention agents. Ann Math Artif Intell. https://doi.org/ms96

Roxana A. Belecheanu, Steve Munroe, Michael Luck, Terry Payne, Tim Miller, Peter McBurney, and Michal Pěchouček. 2006. Commercial applications of agents: lessons, experiences and challenges. AAMAS. https://doi.org/c3t6x5

Steve S. Benfield, Jim Hendrickson, and Daniel Galanti. 2006. Making a strong business case for multiagent technology. AAMAS. https://doi.org/cspvtz

Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni. 2005. Multi-Agent Programming: Languages, Platforms and Applications. Multiagent Systems, Artificial Societies, and Simulated Organizations 15, Springer, ISBN 0-387-24568-5

Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni. 2009. Multi-Agent Programming, Languages, Tools and Applications. Springer, ISBN 978-0-387-89298-6

Rafael H. Bordini and Jürgen Dix. 2016. Programming Multiagent Systems, chapter 13 of Multiagent Systems, Second Edition Edited by Gerhard Weiss, 2016. ISBN: 9780262533874. MIT Press. Pages 587-639

Rafael H. Bordini, Amal El Fallah Seghrouchni, Koen Hindriks, and Brian Logan, Alessandro Ricci. 2020. Agent programming in the cognitive era. Auton Agent Multi-Agent Syst. https://doi.org/ggwqv2

Michael E. Bratman. 1987. Intentions, Plans, and Practical Reason. Harvard University Press, Cambridge, MA.

Daniela Briola, Angelo Ferrando, and Viviana Mascardi. 2023. Fantastic MASs and Where to Find Them: First Results and Lesson Learned. EMAS. https://doi.org/ms97

Amit K. Chopra and Samuel H. Christie V. 2023. Communication Meaning: Foundations and Directions for Systems Research. AAMAS.

Louise Dennis and Michael Fisher. 2023. Verifiable Autonomous Systems: Using Rational Agents to Provide Assurance about Decisions Made by Machines. Cambridge University Press, 2023. ISBN 978-1108484992

Virginia Dignum and Frank Dignum. 2010. Designing agent systems: state of the practice. https://doi.org/fnqs7n

Michael Fisher, Viviana Mascardi, Kristin Yvonne Rozier, Bernd-Holger Schlingloff, Michael Winikoff, and Neil Yorke-Smith. 2021. Towards a framework for certification of reliable autonomous systems. Auton Agent Multi-Agent Syst. https://doi.org/ms98

Klaus Havelund, Michael Lowry, SeungJoon Park, Charles Pecheur, John Penix, Willem Visser, and Jonathan White. 2000. Formal Analysis of the Remote Agent Before and After Flight. Proc. 5th NASA Langley Formal Methods Workshop, Williamsburg, VA, Jun. 13-15 2000.

Koen V. Hindriks. 2012. Debugging Is Explaining. PRIMA https://doi.org/ms99

Koen V. Hindriks. 2014. The Shaping of the Agent-Oriented Mindset. EMAS. https://doi.org/mtbb Slides: https://emas14.wordpress.com/wp-content/uploads/2013/11/koen-emas-14.pdf

The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems. Ethically Aligned Design: A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems, Version 2. IEEE, 2017. http://standards. ieee.org/develop/indconn/ec/autonomous_systems.html

F. F. Ingrand, M. P. Georgeff and A. S. Rao. 1993. An architecture for real-time reasoning and system control. IEEE Expert. http://doi.org/dz574s

Amy J. Ko and Brad A. Myers. 2008. Debugging reinvented: asking and answering why and why not questions about program behavior. ICSE. https://doi.org/dgvqm8

Brian Logan. 2015. A Future for Agent Programming. EMAS. https://doi.org/mtbc

Bertram F. Malle. 2004. How the Mind Explains Behavior: Folk Explanations, Meaning, and Social Interaction. The MIT Press. https://doi.org/mtbd

Bibliography (2/2)

Aditya Mathur. 2008. Foundations of Software Testing. Pearson.

Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. AIJ. https://doi.org/gfwcxw

Jörg P. Müller and Klaus Fischer. 2014. Application Impact of Multi-agent Systems and Technologies: A Survey. In: Shehory, O., Sturm, A. (eds) Agent-Oriented Software Engineering. Springer, Berlin, Heidelberg. https://doi.org/mtbg

Steve Munroe, Tim Miller, Roxana Belecheanu, Michal Pěchouček, Peter McBurney, and Michael Luck. 2017. Crossing the agent technology chasm: Lessons, experiences and challenges in commercial applications of agents. The Knowledge Engineering Review. http://doi.org/b6xbv3

Lin Padgham, John Thangarajah, and Michael Winikoff. 2008. The Prometheus Design Tool – A Conference Management System Case Study. AOSE. https://doi.org/fn7r5q

Lin Padgham and Michael Winikoff. Developing Intelligent Agent Systems: A Practical Guide. June 2004, ISBN 0-470-86120-7, John Wiley and Sons.

Alessandro Ricci. 2022. "Go to the Children": Rethinking Intelligent Agent Design and Programming in a Developmental Learning Perspective. AAMAS.

Sebastian Rodriguez, John Thangarajah, and Michael Winikoff. 2023. A Behaviour-Driven Approach for Testing Requirements via User and System Stories in Agent Systems. AAMAS.

Dhirendra Singh and Lin Padgham. 2015. Community Evacuation Planning for Bushfires Using Agent-Based Simulation: Demonstration. AAMAS.

Munindar P. Singh. 2011. Information-driven interaction-oriented programming: BSPL, the blindingly simple protocol language. AAMAS.

R. L. Teach and E. H. Shortliffe. 1981. An Analysis of Physician Attitudes Regarding Computer-Based Clinical Consultation Systems. Computers and Biomedical Research, 14:542–558.

John Thangarajah, Lin Padgham, and Michael Winikoff. 2005. Prometheus design tool. AAMAS. https://doi.org/bhchsb

Michael Winikoff. 2017. Debugging Agent Programs with Why? Questions. AAMAS.

Michael Winikoff and Stephen Cranefield. 2014. On the testability of BDI agent systems. JAIR. http://doi.org/mtbj

Michael Winikoff, Lin Padgham, and James Harland. 2001. Simplifying the Development of Intelligent Agents. AI. https://doi.org/dvbqgb

Michael Winikoff, Galina Sidorenko, Virginia Dignum, and Frank Dignum. 2021. Why bad coffee? Explaining BDI agent behaviour with valuings. AIJ. https://doi.org/gm6wss

Michael Winikoff, Nitin Yadav, and Lin Padgham. 2018. A new Hierarchical Agent Protocol Notation. Auton Agent Multi-Agent Syst. https://doi.org/gct7gn

Pınar Yolum and Munindar P. Singh. 2002. Commitment Machines. ATAL. https://doi.org/c8d6cx