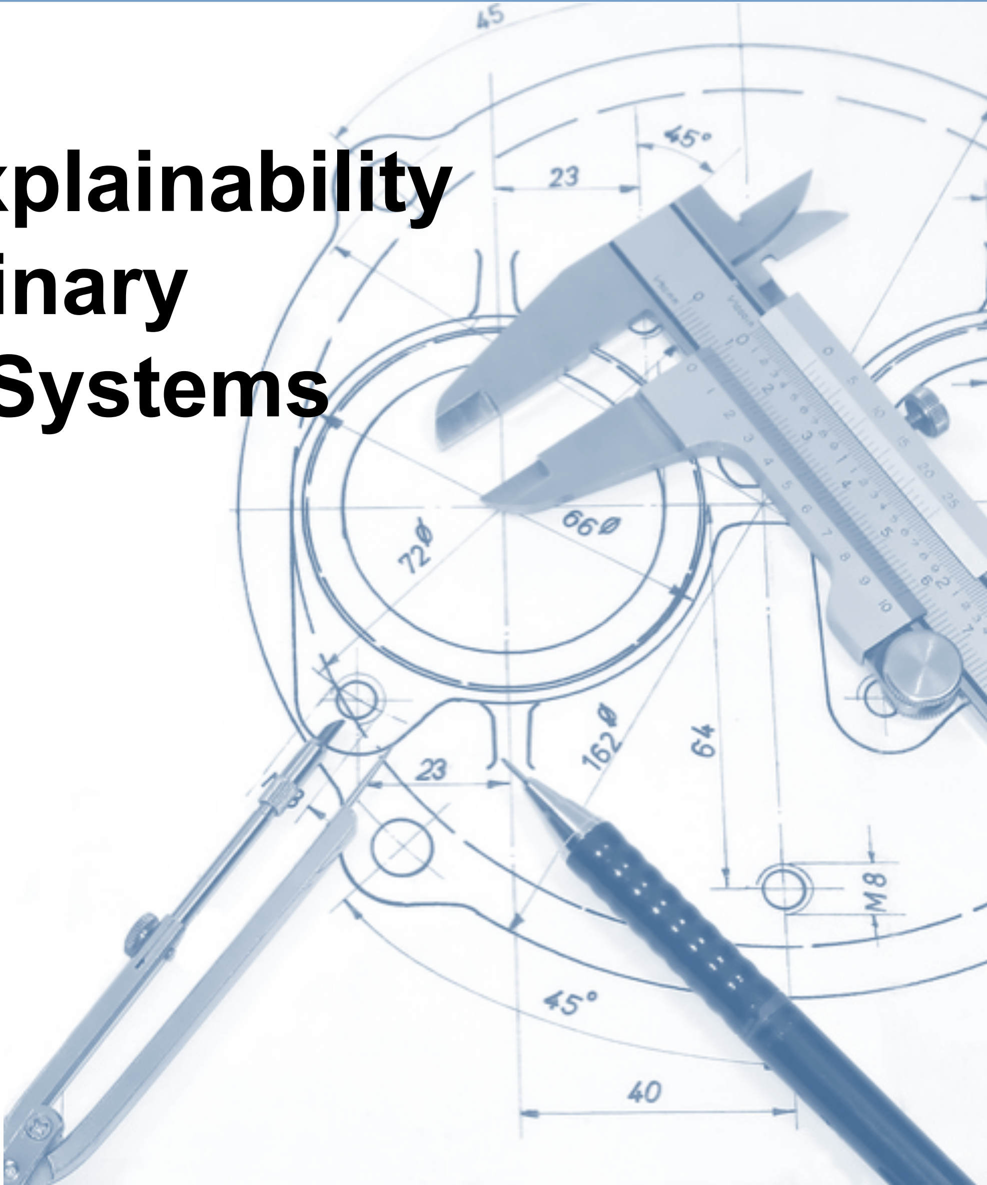


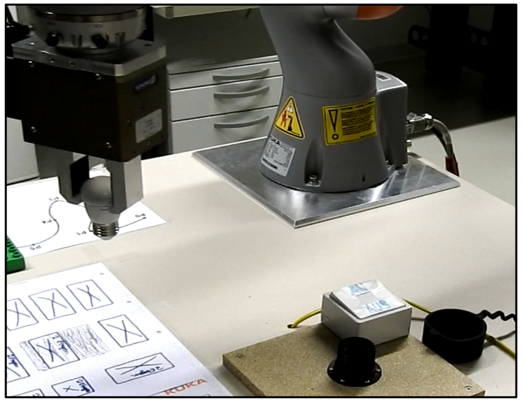
Model-Driven Explainability for Multi-Disciplinary Cyber-Physical Systems Engineering

Andreas Wortmann
Software Engineering
RWTH Aachen

http://www.se-rwth.de/
@andwor

About me

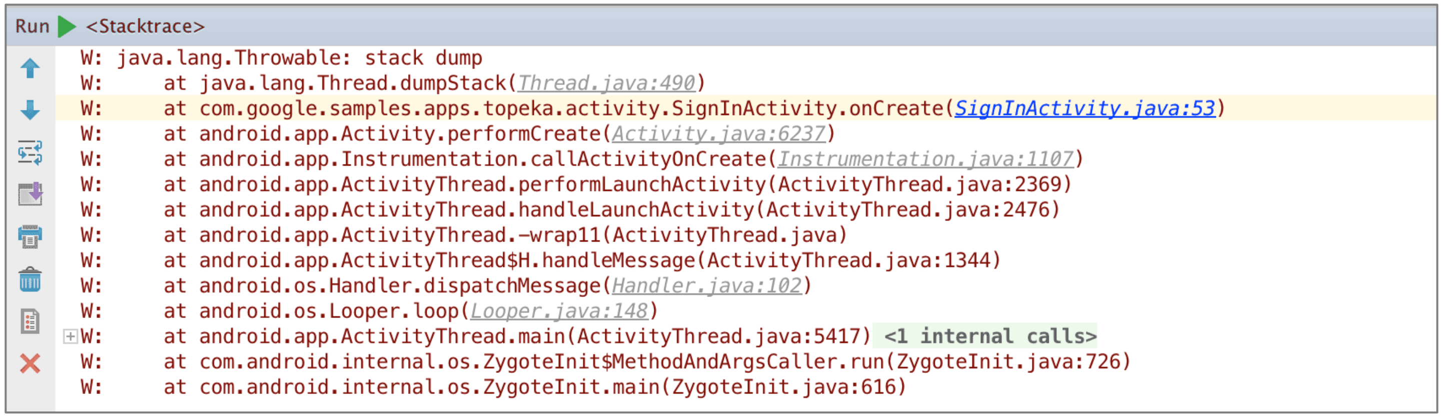
- Since 2011 working with robotics
  - knowledge-based (Golog, …)
  - imperative (ROS, SmartSoft)
  - educational & industrial
- PhD'16 on extensible ADLs for CPS
- Currently work in model-driven systems engineering
- Language-oriented systems engineering
  - build proper software languages efficiently
  - tailor, reuse, integrate existing languages
  - across different technological spaces



Andreas Wortmann
Software Engineering
RWTH Aachen

The sad state of software explainability

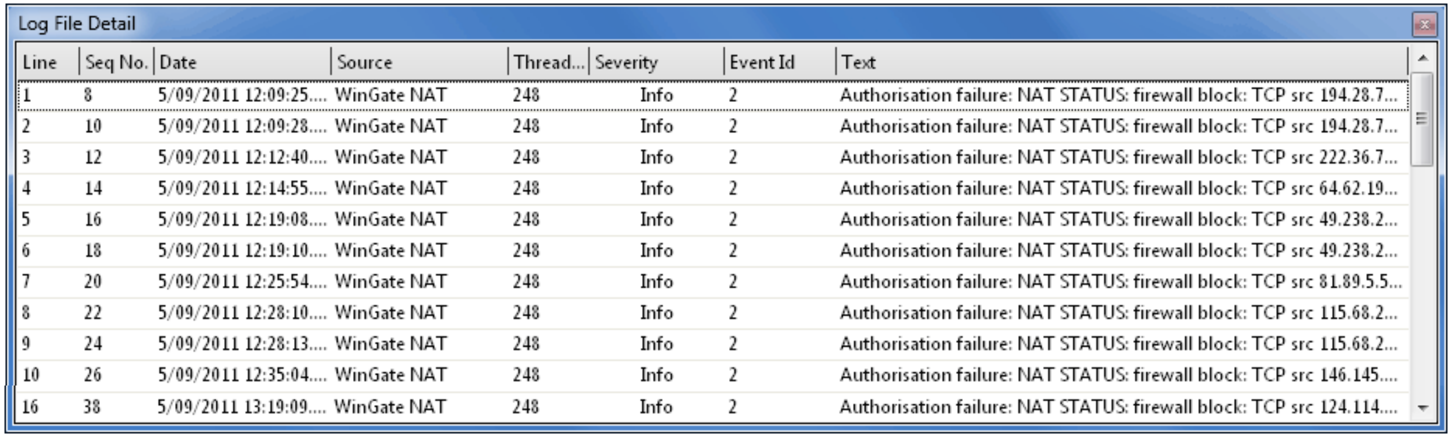
- Stack traces too technical for many purposes



Run <Stacktrace>

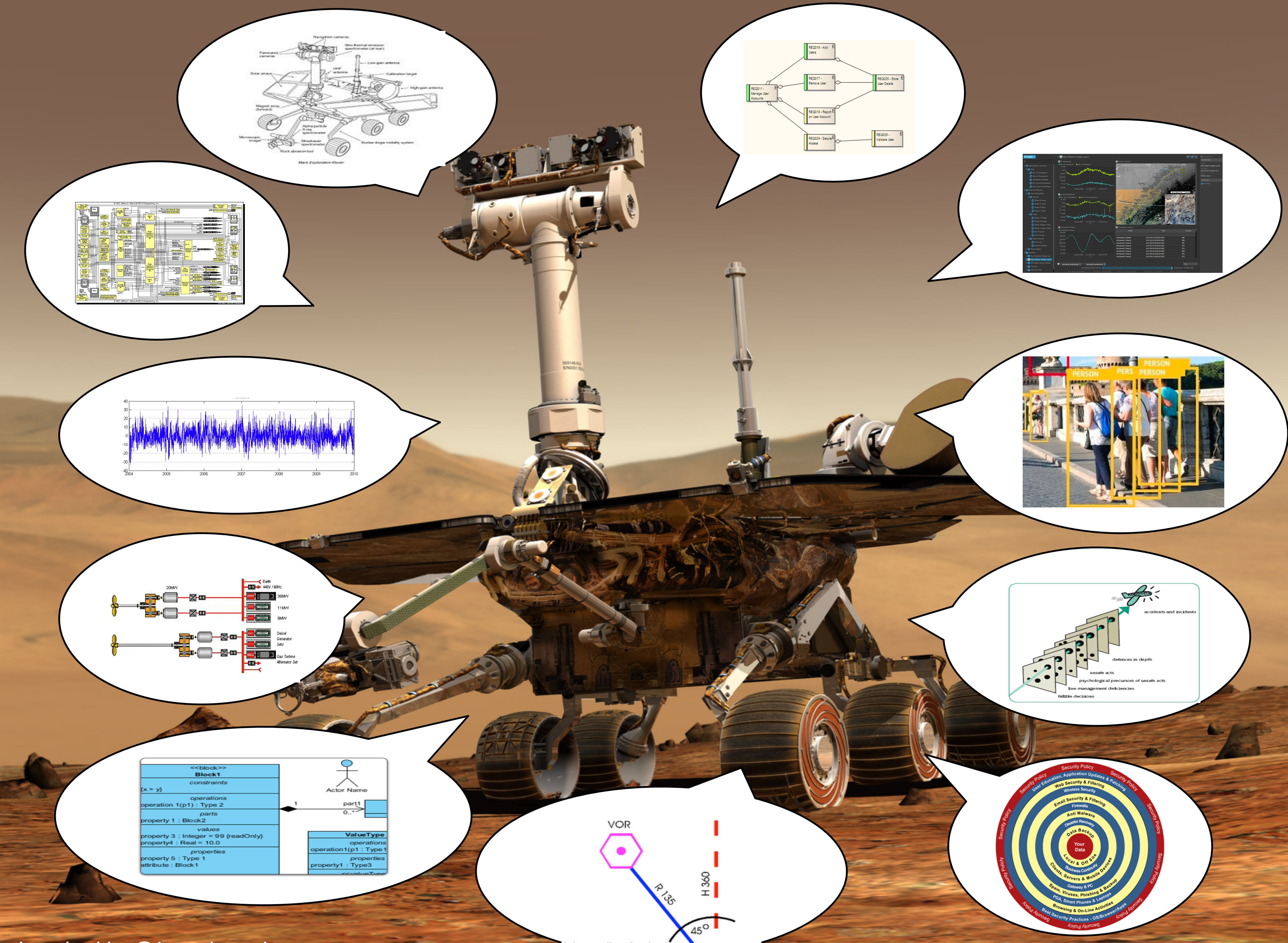
```
W: java.lang.Throwable: stack dump
W:     at java.lang.Thread.dumpStack(Thread.java:490)
W:     at com.google.samples.apps.topeka.activity.SignInActivity.onCreate(SignInActivity.java:53)
W:     at android.app.Activity.performCreate(Activity.java:6237)
W:     at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1107)
W:     at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2369)
W:     at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2476)
W:     at android.app.ActivityThread.-wrap11(ActivityThread.java)
W:     at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1344)
W:     at android.os.Handler.dispatchMessage(Handler.java:102)
W:     at android.os.Looper.loop(Looper.java:148)
W:     at android.app.ActivityThread.main(ActivityThread.java:5417) <1 internal calls>
W:     at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:726)
W:     at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:616)
```

- Log files too verbose, not abstract enough, not reader-specific



Log File Detail

Line	Seq No.	Date	Source	Thread...	Severity	Event Id	Text
1	8	5/09/2011 12:09:25....	WinGate NAT	248	Info	2	Authorisation failure: NAT STATUS: firewall block: TCP src 194.28.7...
2	10	5/09/2011 12:09:28....	WinGate NAT	248	Info	2	Authorisation failure: NAT STATUS: firewall block: TCP src 194.28.7...
3	12	5/09/2011 12:12:40....	WinGate NAT	248	Info	2	Authorisation failure: NAT STATUS: firewall block: TCP src 222.36.7...
4	14	5/09/2011 12:14:55....	WinGate NAT	248	Info	2	Authorisation failure: NAT STATUS: firewall block: TCP src 64.62.19...
5	16	5/09/2011 12:19:08....	WinGate NAT	248	Info	2	Authorisation failure: NAT STATUS: firewall block: TCP src 49.238.2...
6	18	5/09/2011 12:19:10....	WinGate NAT	248	Info	2	Authorisation failure: NAT STATUS: firewall block: TCP src 49.238.2...
7	20	5/09/2011 12:25:54....	WinGate NAT	248	Info	2	Authorisation failure: NAT STATUS: firewall block: TCP src 81.89.5.5...
8	22	5/09/2011 12:28:10....	WinGate NAT	248	Info	2	Authorisation failure: NAT STATUS: firewall block: TCP src 115.68.2...
9	24	5/09/2011 12:28:13....	WinGate NAT	248	Info	2	Authorisation failure: NAT STATUS: firewall block: TCP src 115.68.2...
10	26	5/09/2011 12:35:04....	WinGate NAT	248	Info	2	Authorisation failure: NAT STATUS: firewall block: TCP src 146.145....
16	38	5/09/2011 13:19:09....	WinGate NAT	248	Info	2	Authorisation failure: NAT STATUS: firewall block: TCP src 124.114....



inspried by @bcombemale

Software language engineering gives us better tools to explicate intent and purpose than pure code

- ”The limits of my language are the limits of my world” (Wittgenstein)
- Stakeholders of CPS speak different languages and give explanations in different languages
  - ➢ so do their software modules
- Understanding emergent system behavior requires understanding all related modules
- In a way that supports
  - reasoning about facts (what)
  - contrasting observations (why)
  - Enquiring intentions (how)
- Suitable modeling languages can support CPS explainability at run time and at design-time

Towards explanation languages for multi-disciplinary cyber-physical systems

- Modeling languages that describe explanation (parts)
  - to explain behavior based on lower level facts and explanations (F&E)

- Either general (e.g., ATL) or domain-specific explanation languages[1]
  - former better integratable, latter better accessible, demand integration

- Systems produce histories = ordered lists of F&Es in suitable languages

- F&E yield meta information (source, purpose) to reason about system behavior (*e.g., "show all crash-related info but abstract from battery"*)

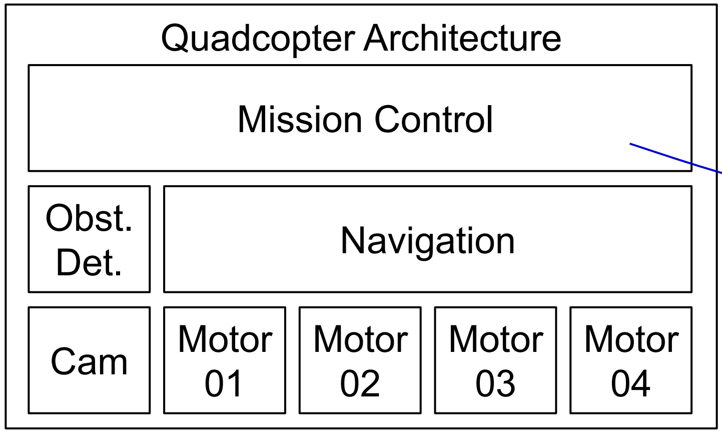
- Such explanation should be
  - receiver-specific (propulsion expert no interest in HMI explanation parts)
  - message-specific (e.g., by giving meaning stack trace segments)
  - time-specific (e.g., truncate irrelevant explanation parts)

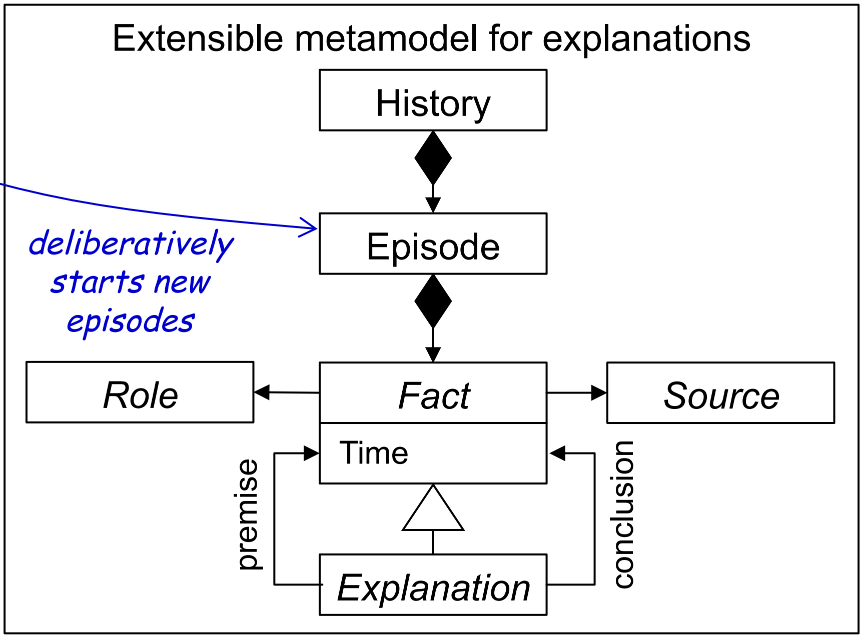
- Across models of different domains
- Throughout the complete system lifecycle

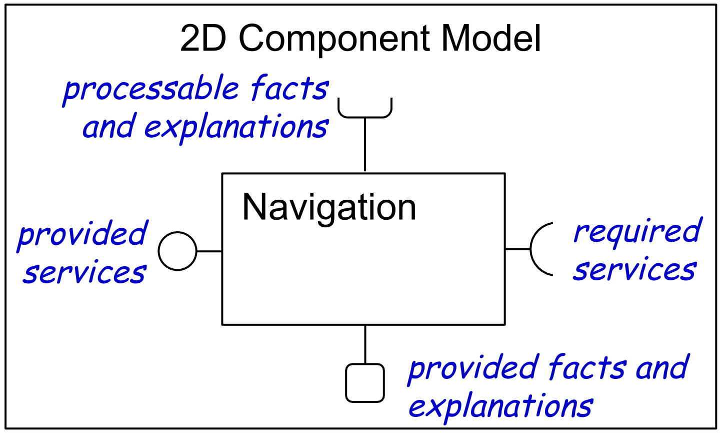
[1] K. Hölldobler, B. Rumpe, A. Wortmann. Software Language Engineering in the Large: Towards Composing and Deriving Languages. In: Computer Languages, Systems & Structures, 54, 2018.

A 2D component model to explain the behavior of a package delivery quadcopter

- Systems engineering leverages component-based notions
- Explanations as 1st level concerns in component (meta) model




Extensible metamodel for explanations
History
Episode
deliberatively starts new episodes
Role
Fact
Source
Time
premise
conclusion
Explanation

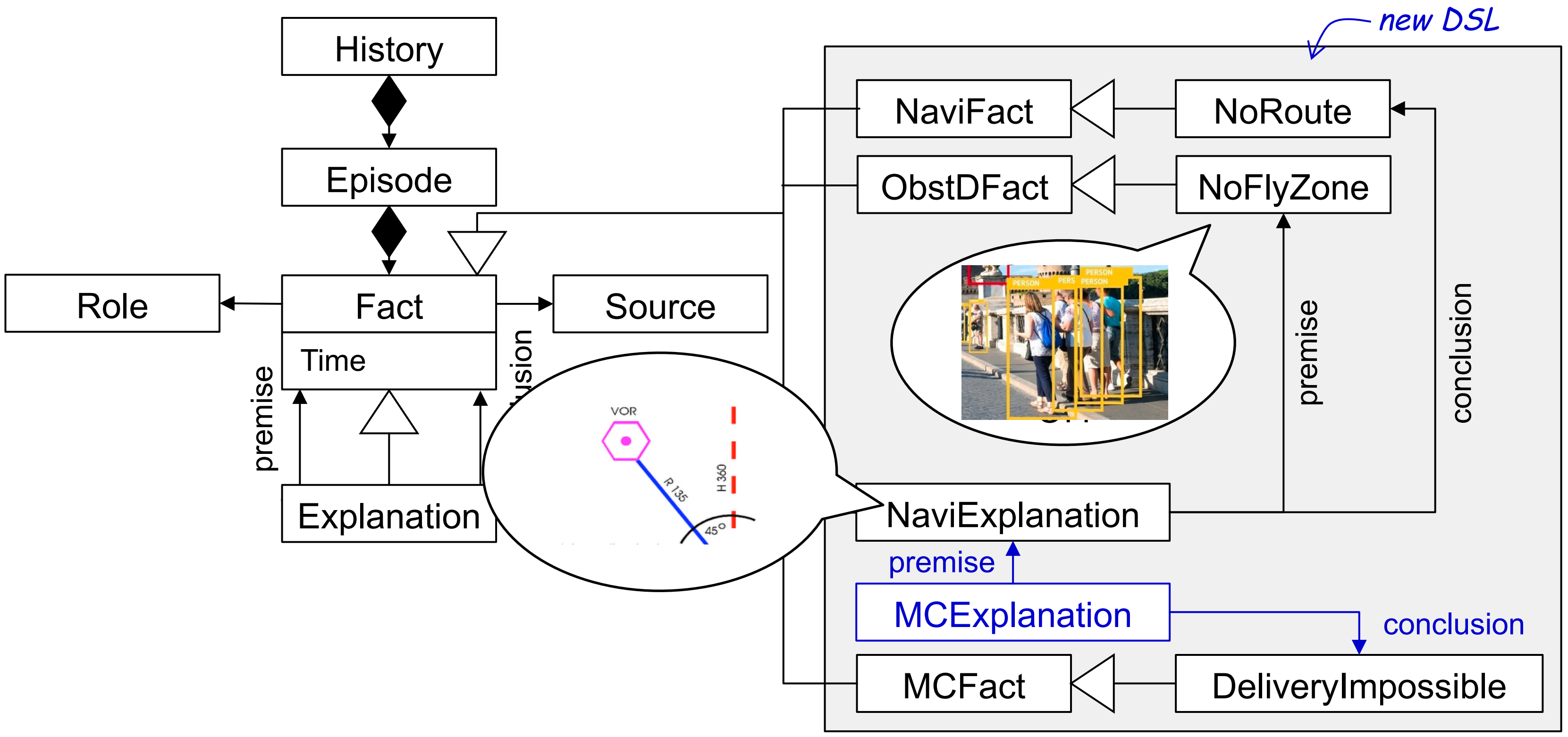

2D Component Model
processable facts and explanations
Navigation
provided services
required services
provided facts and explanations

- Architecture supports operating on F&E
- Metamodel supports tailoring to domain-specific F&E

Andreas Wortmann
Software Engineering
RWTH Aachen

A 2D component model to explain the behavior of a package delivery quadcopter

- Domain-specific instantiation of the quadcopter explanation language (e.g., language embedding[1] or merging[2])



[1] K. Hölldobler, B. Rumpe, A. Wortmann. Software Language Engineering in the Large: Towards Composing and Deriving Languages. In: Computer Languages, Systems & Structures, 54, 2018.

[2] Degueule, T., Combemale, B., Blouin, A., Barais, O., & Jézéquel, J. M. Melange: A meta-language for modular and reusable development of DSLs. In Proceedings of the 2015 SLE. 2015.

There are many challenges in explainable software for cyber-physical systems…

- Capturing and integrating facts & explanations of different domains
- Efficient adaptation between F&E of different components
  - normal system integration activity?
- Automatically deriving explanations
- A posteriori explainer integration into existing (legacy) systems
- Automated abstraction and history truncation of explanations
- Cooperative / partial explanations

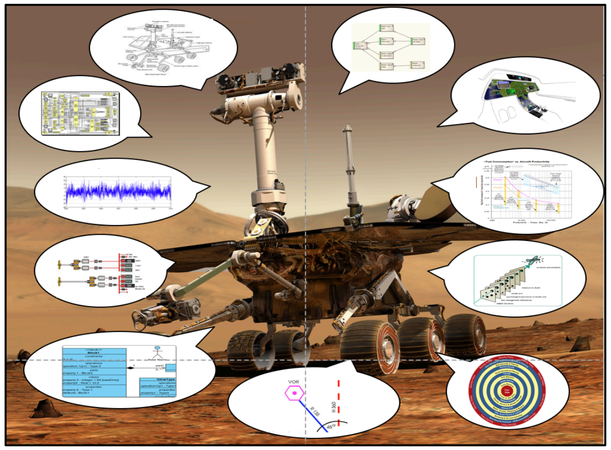
⇨ … to achieve any of these, we first need explicit explanations

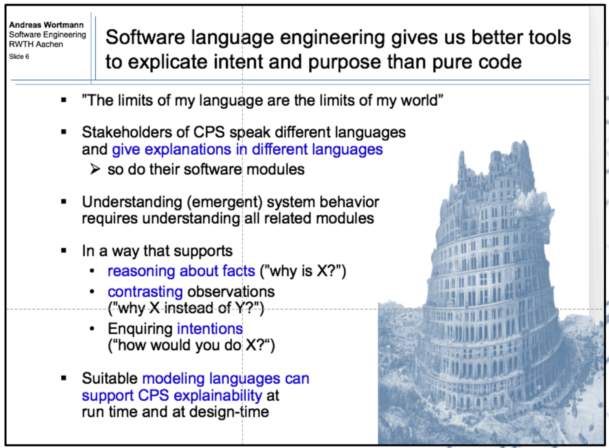
Our answers to workshop-related questions

- ES4CPS problems that we are interested in
  - making explanations explicit
  - leveraging explicit explanations at run time
  - querying explanations (facts, contrasts, …)
- ES4CPS expertise that we can contribute
  - modular software language engineering
  - smart manufacturing, automotive software testing, robotics
  - formal systems modeling (focus, mona, isabelle)
- External expertise that we need
  - domain-specific insights into explanations
  - multi-disciplinary modeling
  - reasoning about explanations

Andreas Wortmann
Software Engineering
RWTH Aachen

Thank You

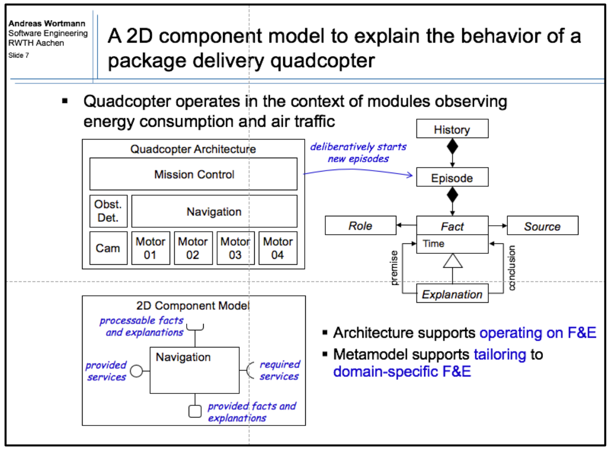




Andreas Wortmann
Software Engineering
RWTH Aachen

Software language engineering gives us better tools to explicate intent and purpose than pure code

- "The limits of my language are the limits of my world"
- Stakeholders of CPS speak different languages and give explanations in different languages
  - ➢ so do their software modules
- Understanding (emergent) system behavior requires understanding all related modules
- In a way that supports
  - reasoning about facts ("why is X?")
  - contrasting observations ("why X instead of Y?")
  - Enquiring intentions ("how would you do X?")
- Suitable modeling languages can support CPS explainability at run time and at design-time

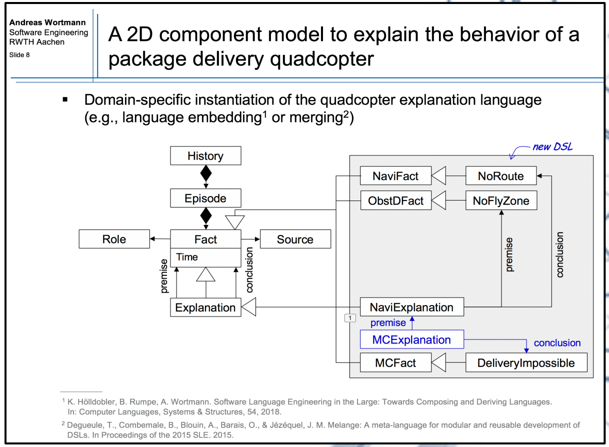


Andreas Wortmann
Software Engineering
RWTH Aachen

A 2D component model to explain the behavior of a package delivery quadcopter

- Quadcopter operates in the context of modules observing energy consumption and air traffic

- Architecture supports operating on F&E
- Metamodel supports tailoring to domain-specific F&E



Andreas Wortmann
Software Engineering
RWTH Aachen

A 2D component model to explain the behavior of a package delivery quadcopter

- Domain-specific instantiation of the quadcopter explanation language (e.g., language embedding[1] or merging[2])

[1] K. Hölldobler, B. Rumpe, A. Wortmann. Software Language Engineering in the Large: Towards Composing and Deriving Languages. In: Computer Languages, Systems & Structures, 54, 2018.
[2] Degueule, T., Combemale, B., Blouin, A., Barais, O., & Jézéquel, J. M. Melange: A meta-language for modular and reusable development of DSLs. In Proceedings of the 2015 SLE. 2015.

Publications
wortmann.ac

Videos
wortmann.ac/videos

Twitter
@andwor