Jheronimus Academy of Data Science

TOSCA *Intent* Models: Goal-Modelling for Infrastructure-as-Code

Damian A. Tamburri, Willem-Jan Van den Heuvel (TU/e, UvT - JADS)

Chris Lauwers (VNomic) Paul Lipton (CA Tech.)

Derek Palma (UDemy) Matt Rutkowski (IBM)

What are we up to here today?

- DevOps in a Nutshell
- TOSCA in a Nutshell
- Research Problem & Scope
- TOSCA Intent-Modelling Explained
- Outlook and Future Work
- Conclusions

DevOps

- **What is it:** “Practices or tools that bridge the gap between development and operations”
- **Goal:** Creates a collaborative mindset where a single team performs Dev and Ops
  →the team **must** contain differentiated competences, background, etc.
- **Requires:**
  - Culture management;
  - Automation tools;
  - Organisational as much as technical metrics
  - Continuous sharing artifacts, procedures, languages, approaches…

DevOps

- **What is it:** "Practices or tools that bridge the gap between development and operations"

- **Goal:** Creates a collaborative mindset where a single team performs Dev and Ops
  →the team **must** contain differentiated competences, background, etc.

- **Requires:**
  - Culture management;
  - Automation tools;
  - Organisational as much as technical metrics
  - Continuous sharing artifacts, procedures, languages, approaches…

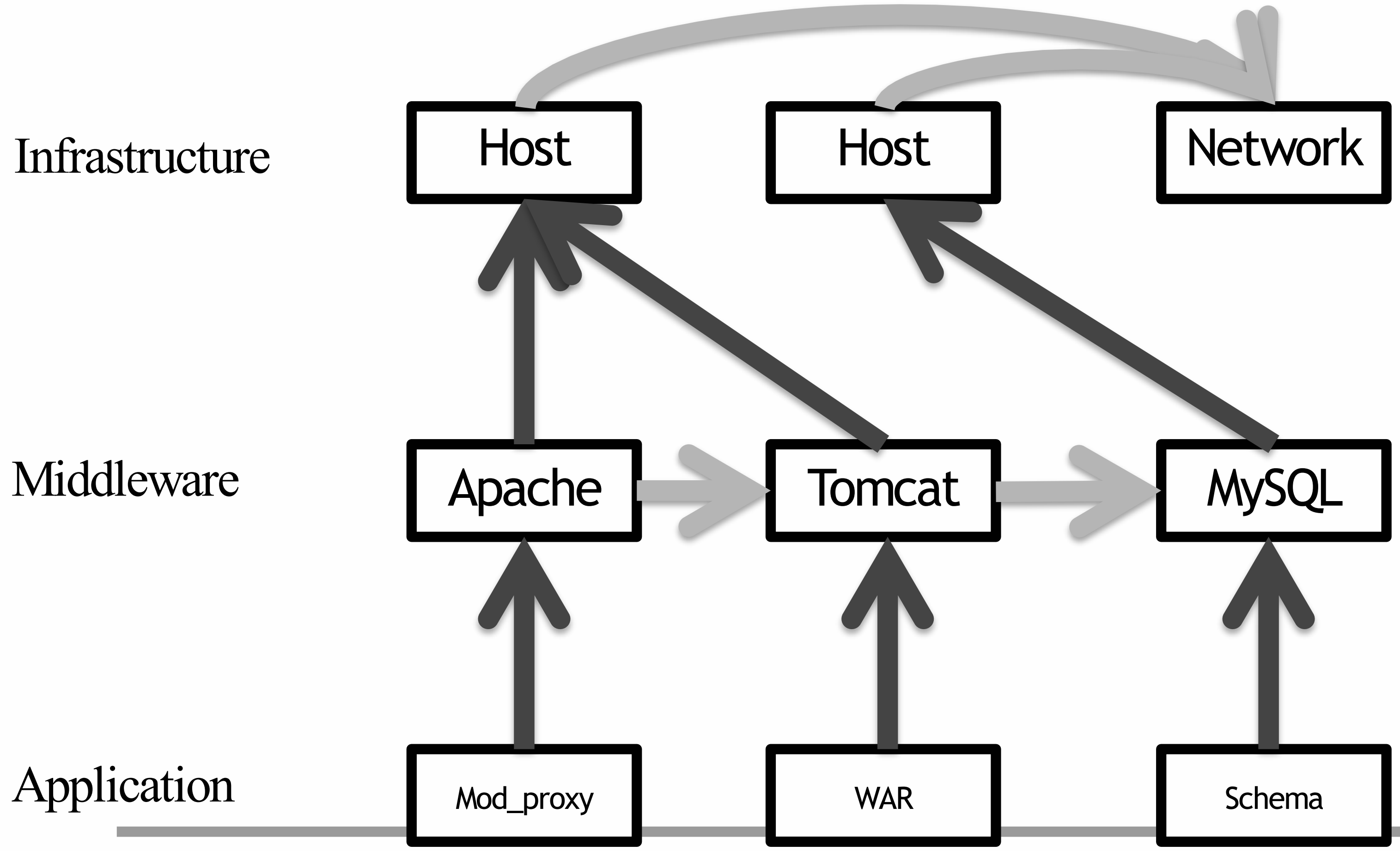
Infrastructure-as-code!

TOSCA in a Nutshell

- OASIS Standard for infrastructure-as-code

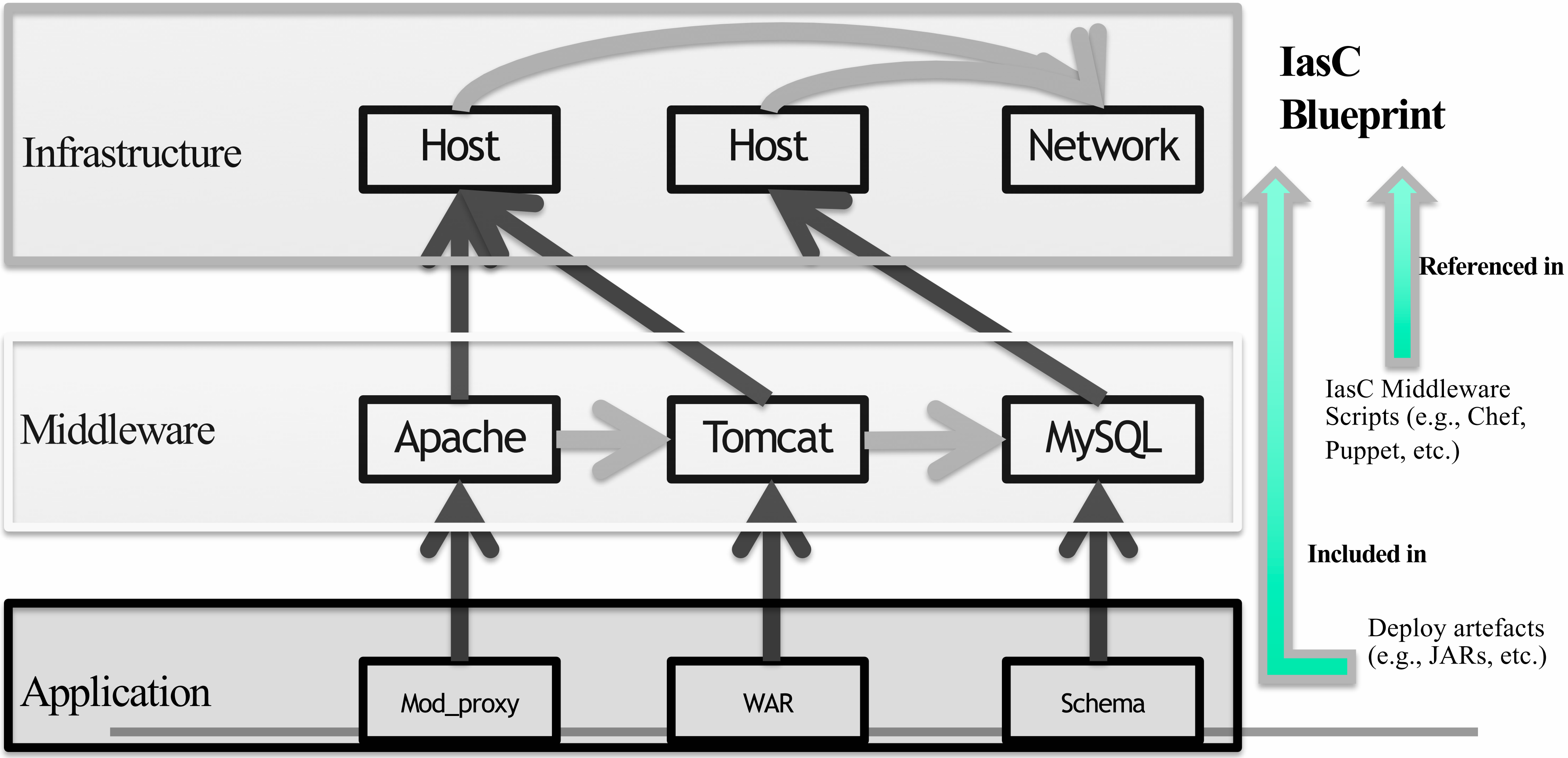
Towards standard Infrastructure Code

➔ **An Application Deployment Topology**, i.e., “a graph of physical artefacts that need support for several lifecycle phases (e.g., procurement, installation, configuration, deployment, undeployment, teardown, etc.)” [6]



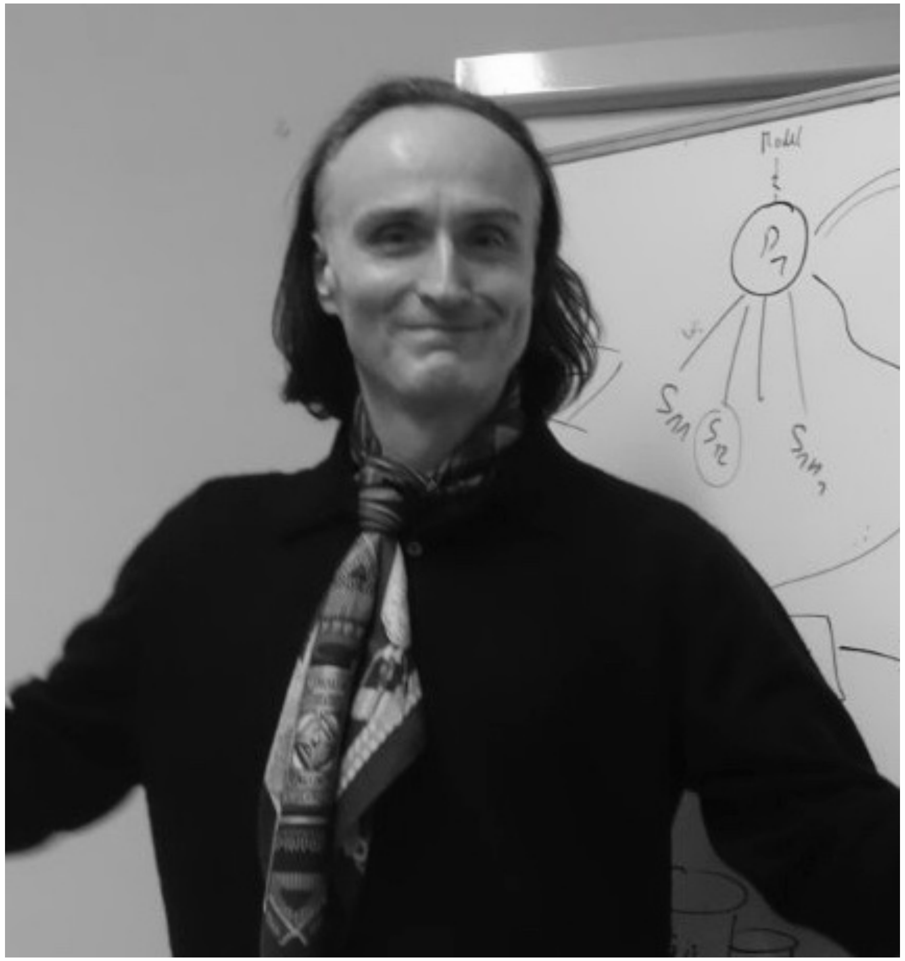
Towards standard Infrastructure Code

➔ **Infrastructure-as-code**, i.e., "a blueprint detailing physical artefacts, all scripts for all lifecycle phases and all artefacts needed for deployment" [6]



TOSCA in a Nutshell

- OASIS Standard for infrastructure-as-code
  - (btw, thanks Frank!)



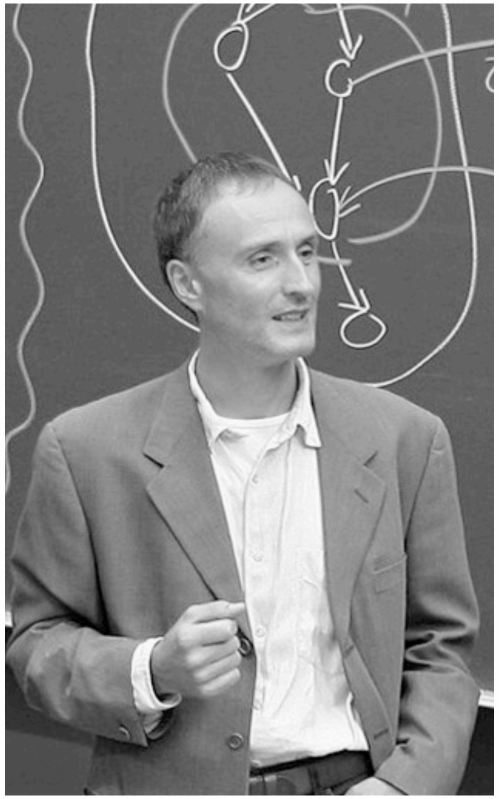
Research Problem & Scope

- **Problem statement**
  - Suppose you want to recommend TOSCA to your friends or foes…

Research Problem & Scope

- **Problem statement**
  - Suppose you want to recommend TOSCA to your friends or foes…

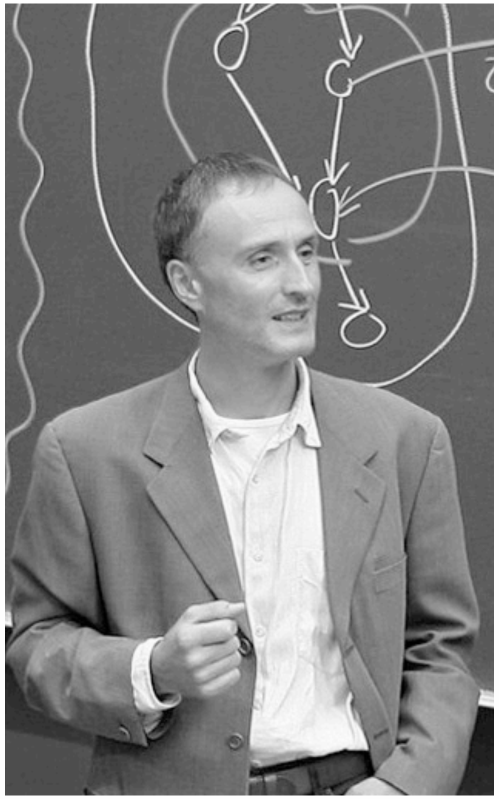
Our Frank



Research Problem & Scope

- **Problem statement**
  - Suppose you want to recommend TOSCA to your friends or foes…

Our Frank



➡

another Frank (not as friendly as ours)

Research Problem & Scope

- **Problem statement**
  - Suppose you want to recommend TOSCA to your friends or foes…

myself

→

Average EU FP7, H2020 reviewer

Research Problem & Scope

- **Problem statement**
  - Suppose you want to recommend TOSCA to your friends or foes…

    RQ: What is its intended *design* and *programming* model?

Research Problem & Scope

- **Problem statement**
  - Suppose you want to recommend TOSCA to your friends or foes…

    RQ: What is its intended *design* and *programming* model?

Why is this important?

1. *If you know the *design model* you can automate it, prepare process models for it…*

2. *If you know the *programming model*, you can extend it, play around with it, design tools for it…*

TOSCA Intent-Modelling Explained

- So… how is TOSCA design model different than those we know already?

TOSCA Intent-Modelling Explained

- So… how is TOSCA design model different than those we know already?

TOSCA Intent-Modelling Explained

- So… how is TOSCA design model different than those we know already?
  - Imperative Design (e.g., BPMN)?

<Frank is involved so this is your first go-to thought…>

TOSCA Intent-Modelling Explained

- So… how is TOSCA design model different than those we know already?
  - Imperative Design (e.g., BPMN)?

<Frank is involved so this is your first go-to thought…>

<BUT… you would be wrong!>

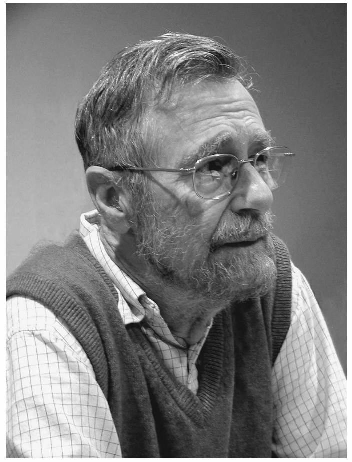
TOSCA Intent-Modelling Explained

- So… how is TOSCA design model different than those we know already?
  - Imperative Design (e.g., BPMN)
  - Declarative Design (e.g., Alloy, SMV, Goal-Modelling)?

TOSCA Intent-Modelling Explained

- So… how is TOSCA design model different than those we know already?
  - Imperative Design (e.g., BPMN)
  - Declarative Design (e.g., Alloy, SMV, Goal-Modelling)?

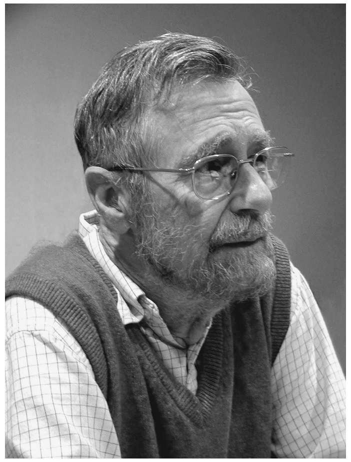
<the formalists would be making this guess, I think ☺>



TOSCA Intent-Modelling Explained

- So… how is TOSCA design model different than those we know already?
  - Imperative Design (e.g., BPMN)
  - Declarative Design (e.g., Alloy, SMV, Goal-Modelling)?

<Wrong again! [but almost right, let's say 50%]>



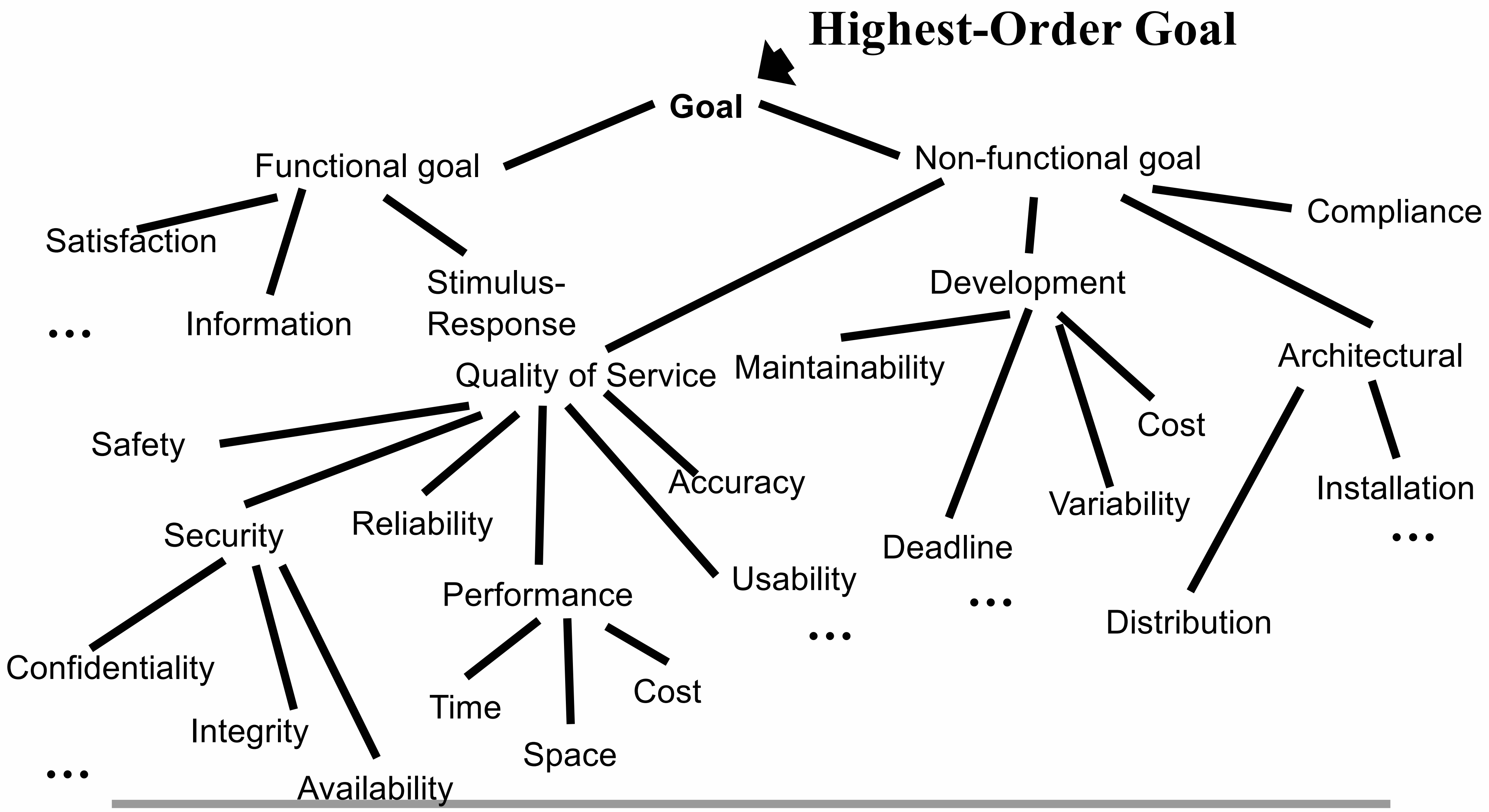
TOSCA Intent-Modelling Explained

- So… how is TOSCA design model different than those we know already?
  - Imperative Design (e.g., BPMN)
  - Declarative Design (e.g., Alloy, SMV, Goal-Modelling)
  - **Intent Design!**

TOSCA Intent-Modelling Explained

- So… how is TOSCA design model different than those we know already?
  - Imperative Design (e.g., BPMN)
  - Declarative Design (e.g., Alloy, SMV, **Goal-Modelling**)
  - **Intent Design!**

TOSCA vs. Goal-Modelling



TOSCA Intent-Modelling Explained

- **Intent modelling!**

  «modelling by specifying a **highest-level** goal to be satisfied, **regardless** of how sub-level goals are satisfied»

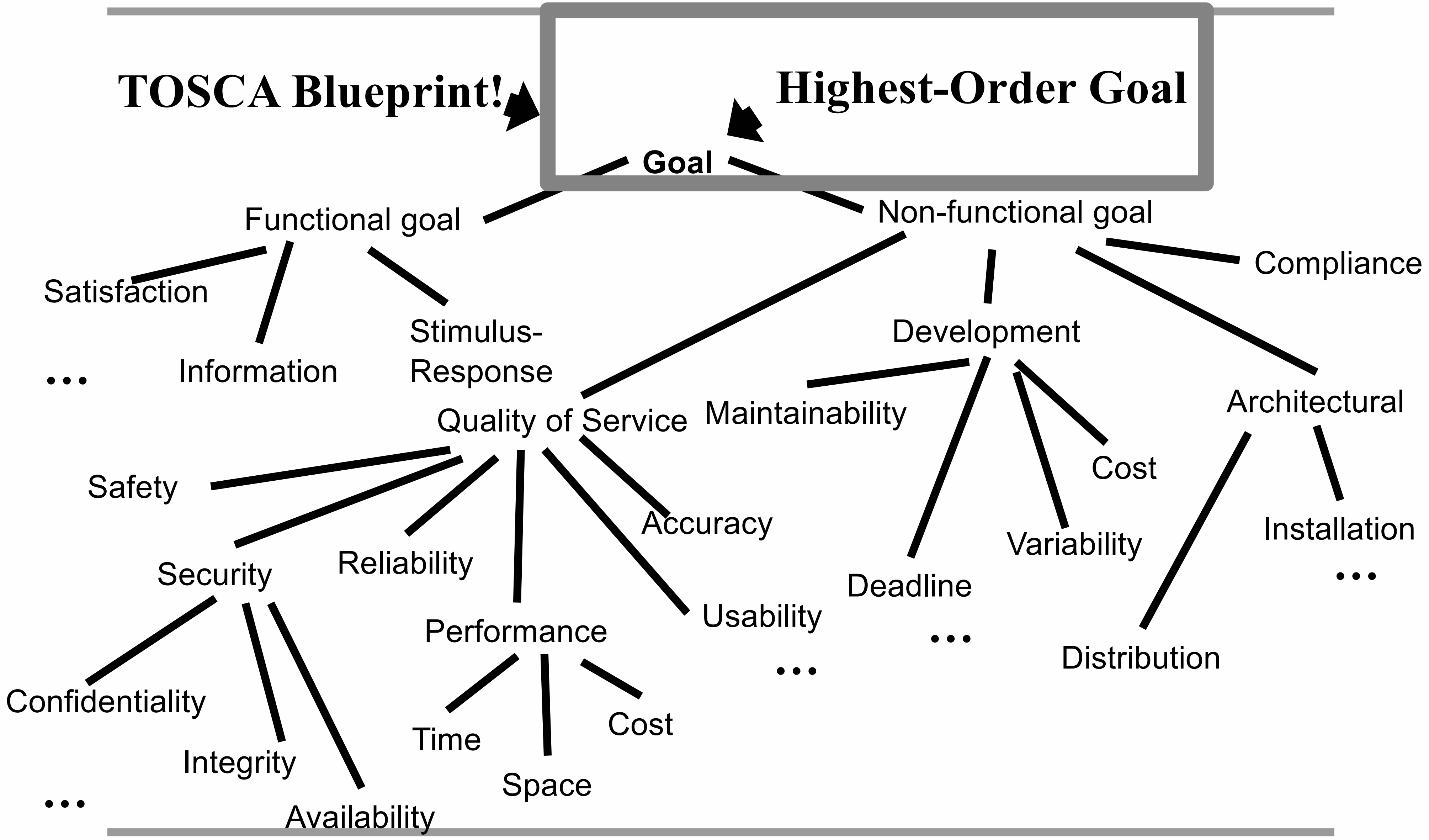
TOSCA Intent-Modelling Explained

- **Intent modelling.**

  «modelling by specifying a **highest-level** goal to be satisfied, **regardless** of how sub-level goals are satisfied»

This goes in the blueprint

TOSCA vs. Goal-Modelling



TOSCA Intent-Modelling Explained

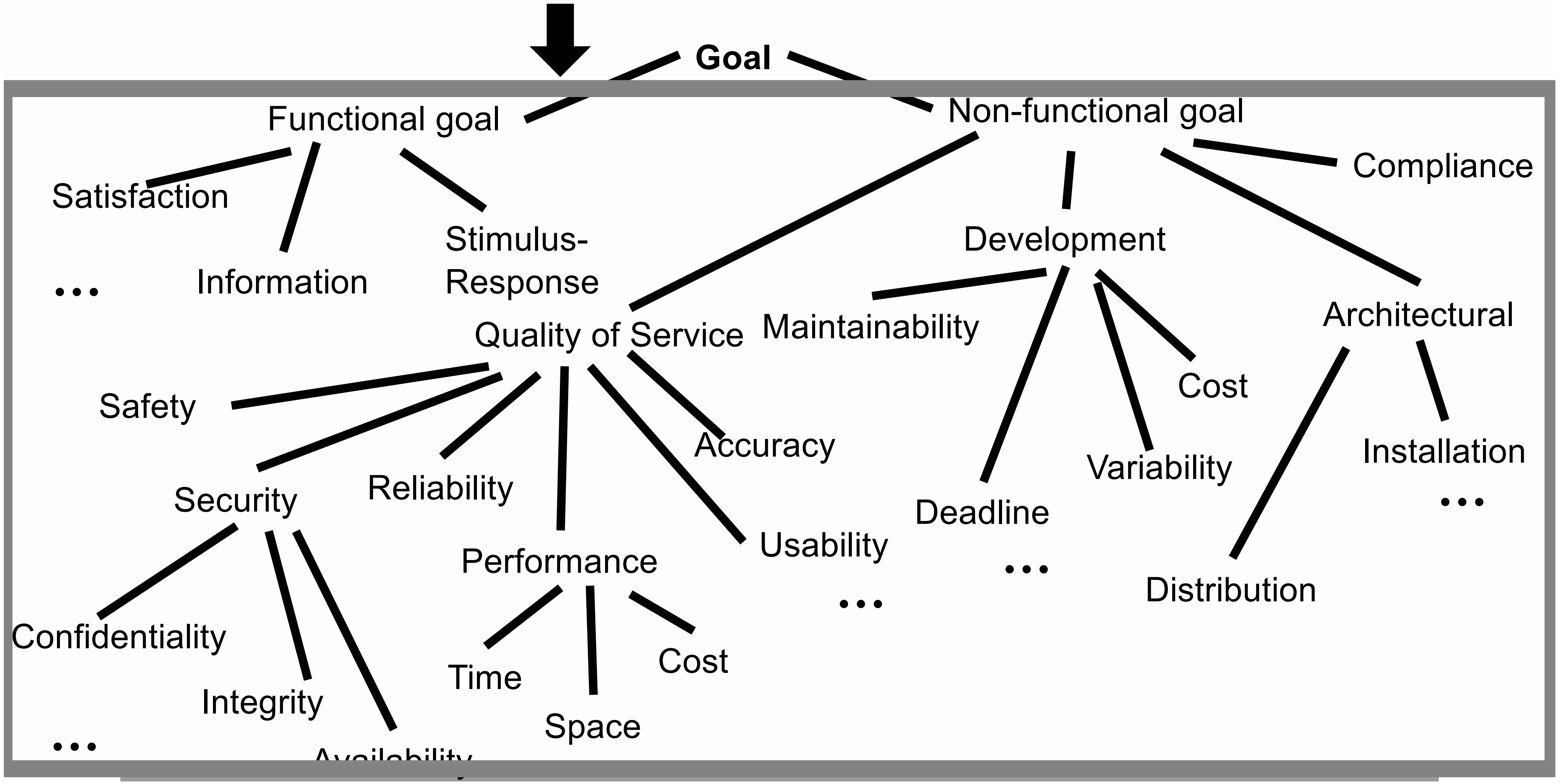
- **Intent modelling.**

  «modelling by specifying a **highest-level** goal to be satisfied, **regardless** of how sub-level goals are satisfied»

This is left to the orchestrator

TOSCA vs. Goal-Modelling

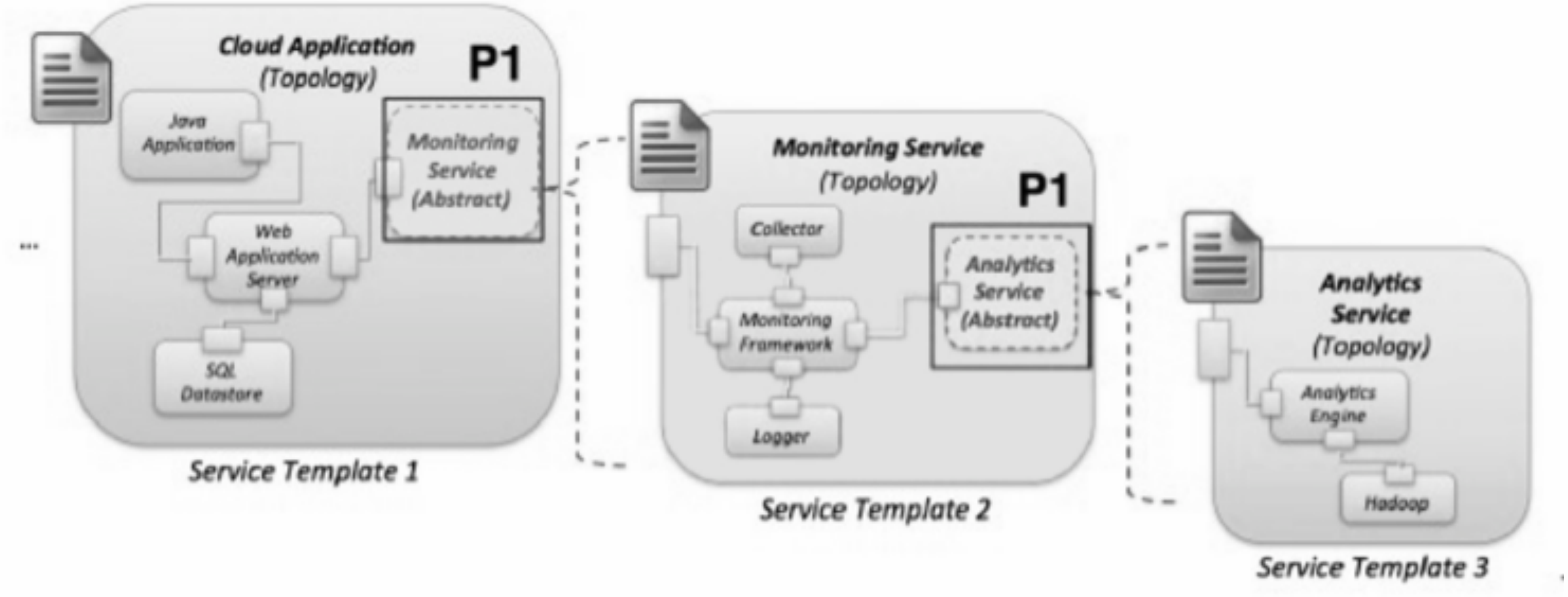
TOSCA Orchestrator!



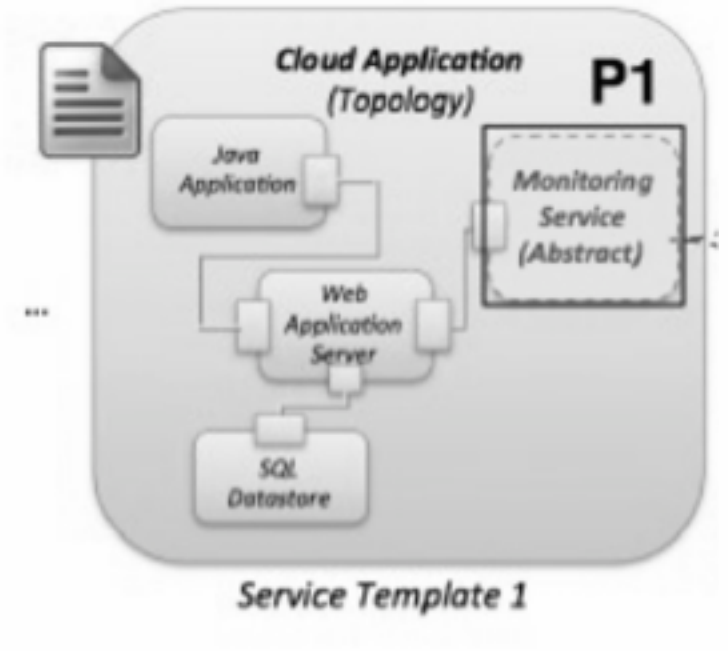
TOSCA Intent-Modelling: what does it mean?

- Empowering the **language** to empower the **orchestrator**

TOSCA Intent-Modelling: what does it mean? Here's an example!

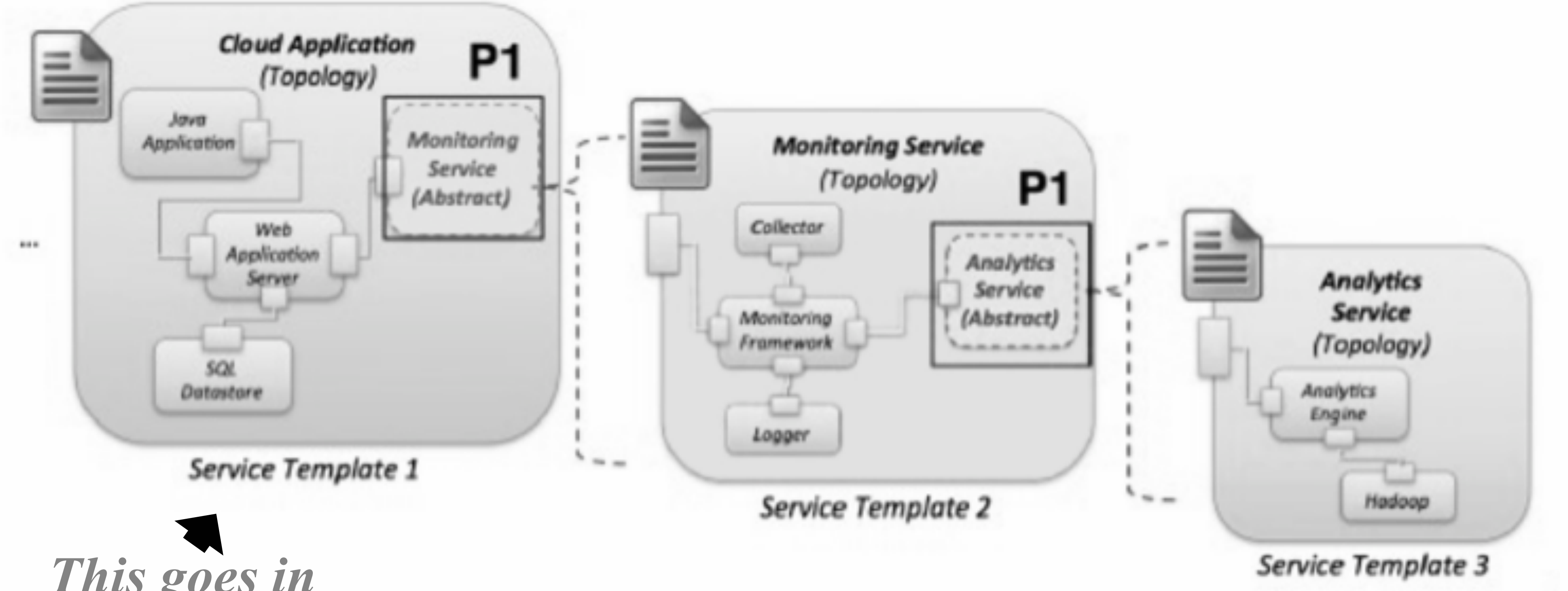


TOSCA Intent-Modelling: what does it mean? Here's an example!



This goes in my blueprint

TOSCA Intent-Modelling: what does it mean? Here's an example!



This goes in my blueprint

This is left to the orchestrator!

Some Intuitive Properties

- **Substitutability**
- **Opportunistic Hierarchization** (instance modelling)
- **Resource-Based Intent Evolution**

Some Intuitive Properties

- **Substitutability**
- **Opportunistic Hierarchization** (instance modelling)
- **Resource-Based Intent Evolution**

Substitutability. *Orchestrator can change any node as long as highest-level goal is maintained and policies are upheld*

Some Intuitive Properties

- **Substitutability**
- **Opportunistic Hierarchization** (instance modelling)
- **Resource-Based Intent Evolution**

Substitutability. *Orchestrator can change any node as long as highest-level goal is maintained and policies are upheld*

This is really nothing new (i.e., SoC lecture from Wolfgang this morning… thanks Wolfgang!)

Some Intuitive Properties

- Substitutability
- **Opportunistic Hierarchization (instance modelling)**
- Resource-Based Intent Evolution

Opportunistic Hierarchization. *Orchestrator creates a hierarchy dynamically at run-time by* ***approximating*** *as much as possible the higher-level goal.*

Some Intuitive Properties

- Substitutability
- Opportunistic Hierarchization (instance modelling)
- **Resource-Based Intent Evolution**

Intent Evolution. *Orchestrator maintains an intent as a steady-state, i.e.,* ***automated maintenance****!*

Conclusions

- Designing and programming for TOSCA involves intent modelling
- Intent modelling means empowering the orchestrator
- Several interesting properties emerge but many are not that new
  - E.g., for services design, QoS assessment/analysis, …
- **But some are *extremely * interesting and may need further research!**

That’s all folks!

Any Questions?

References

[1] Erder, Murat and Pureur, Pierre. Continuous Architecture: Sustainable Architecture in an Agile and Cloud-Centric World. Amsterdam: Morgan Kaufmann, 2016.

[2] Continuous Testing Paperback – January 2, 2014 by W. Ariola, C. Dunlop

[3] Part of the Pipeline: Why Continuous Testing Is Essential, by Adam Auerbach, TechWell Insights August 2015

[4] M. Fowler Continuous Integration, https://www.thoughtworks.com/continuous-integration

[5] *Chen, Lianping (2015) "Continuous Delivery: Huge Benefits, but Challenges Too" IEEE Software*. ***32*** *(2): 50.*

[6] http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/csd03/TOSCA-Simple-Profile-YAML-v1.0-csd03.html

[7] P. Lipton, D. Palma, M. Rutkowski, and D. A. Tamburri, "Tosca solves big problems in the cloud and beyond!" *IEEE Cloud*, vol. 21, no. 11, pp. 31–39, 2016.

[8] Bengtsson, PerOlof, Lassing, Nico, Bosch, Jan and van Vliet, Hans. "Architecture-level modifiability analysis (ALMA)." *Journal of Systems and Software* 69 , no. 1--2 (2004): 129--147.

[9] Tamburri, D. A.; Lago, P. & van Vliet, H. (2013), 'Uncovering Latent Social Communities in Software Development.', *IEEE Software* **30** (1) , 29-36 .

[10] M. Di Penta, D. A. Tamburri, Combining Quantitative and Qualitative Methods in Empirical Software Engineering Proceedings of the 10th Joint Meeting of the European Software Engineering Conference and the ACM Sigsoft Symposium of the Foundations of Software

Other Biblio

[11] Bass, L. J.; Weber, I. M. & Zhu, L. (2015), *DevOps - A Software Architect's Perspective*. , Addison-Wesley .

[12] Tamburri, D. A. & Nitto, E. D. (2015), When Software Architecture Leads to Social Debt., *in* Len Bass; Patricia Lago & Philippe Kruchten, ed., 'WICSA' , IEEE Computer Society, pp. 61-64 .

[13] Tamburri, D. A.; Kruchten, P.; Lago, P. & van Vliet, H. (2015), 'Social debt in software engineering: insights from industry.', *J. Internet Services and Applications* **6** (1) , 10:1-10:17 .

[14] Tamburri, D. A.; Lago, P. & van Vliet, H. (2013), 'Organizational social structures for software engineering.', *ACM Comput. Surv.* **46** (1) , 3 .