Functionalities of Ubiquitous Service Platforms

Dr. Stephan Steglich
stephan.steglich@fokus.fraunhofer.de
Fraunhofer Institute FOKUS
February, 2005

Functionalities of Ubiquitous Service Platforms

What functionalities are required for ubiquitous service platforms?

Non-functional aspects including:

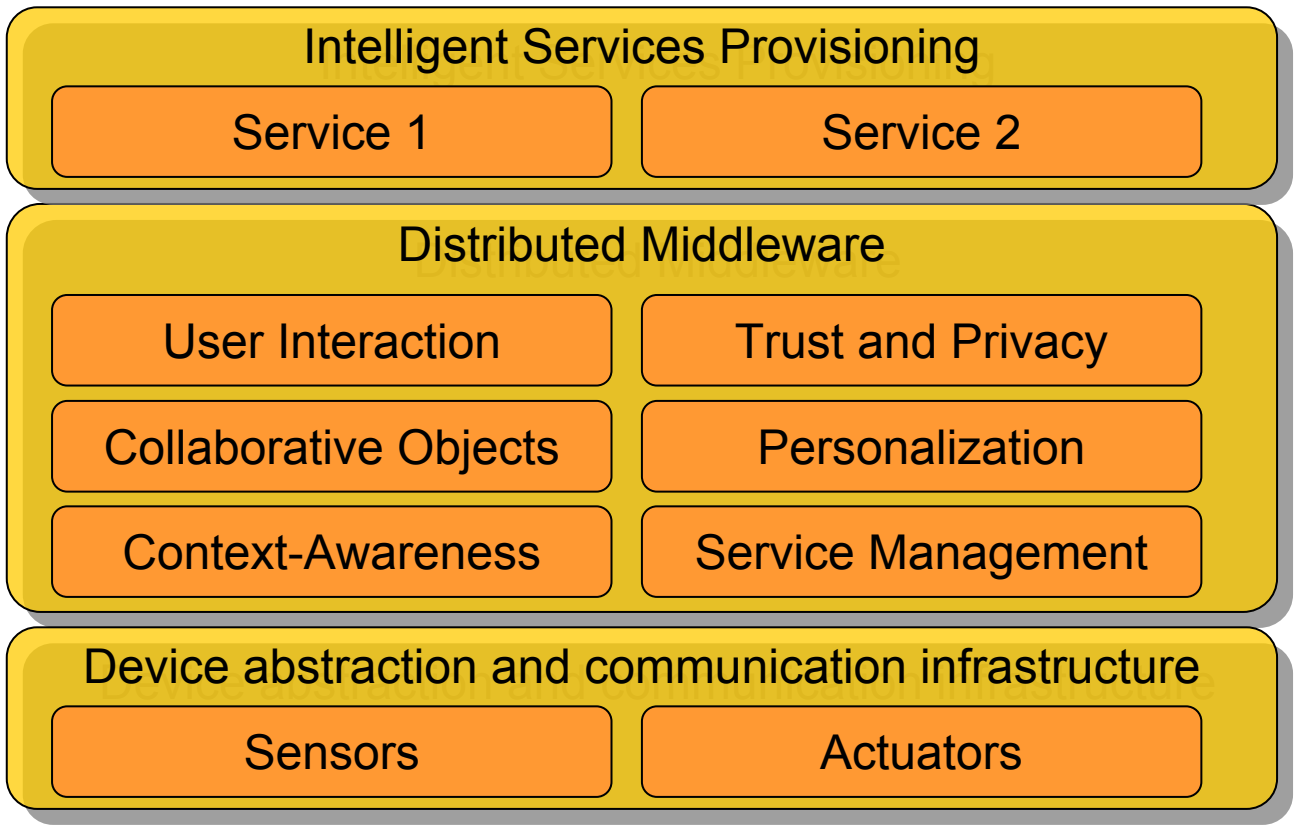
- Innovative & flexible business models
- Service lifecycle management
  - Rapid deployment
  - Distributed (Component & Service) Lifecycle management
- Component-based Management
- Service management
  - Traditional FCAPS
  - Trust and Privacy management and guarantee
- Etc.

Functional aspects including:

- Adaptive Behavior
  - Learning
  - Providing the necessary data in suitable format
- Distribution of functional components
  - But ensuring cooperation of distributed functional components
  - Ad-hoc cooperation
- Rule-based programming

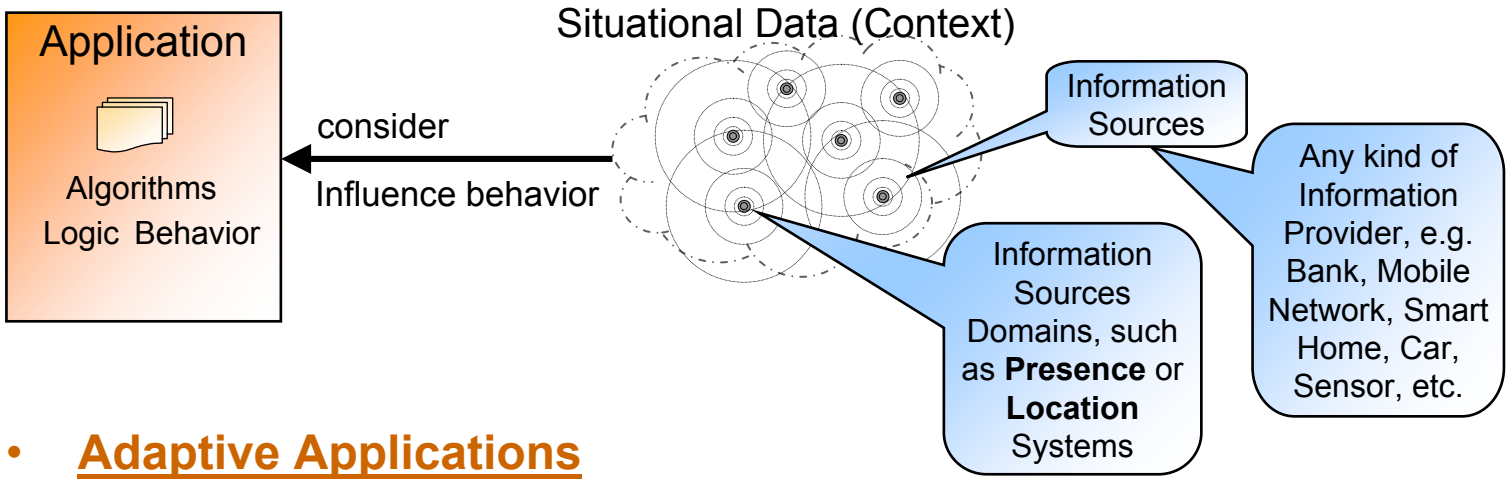
Essential Functions

- Ubiquitous Service Platforms have to support a number of **essential functions** to enable application and service developers to exploit the potential of the mobile environment to the full extend.
- Enabling **Adaptive Applications and Services**



Adaptive Applications / Smart Applications

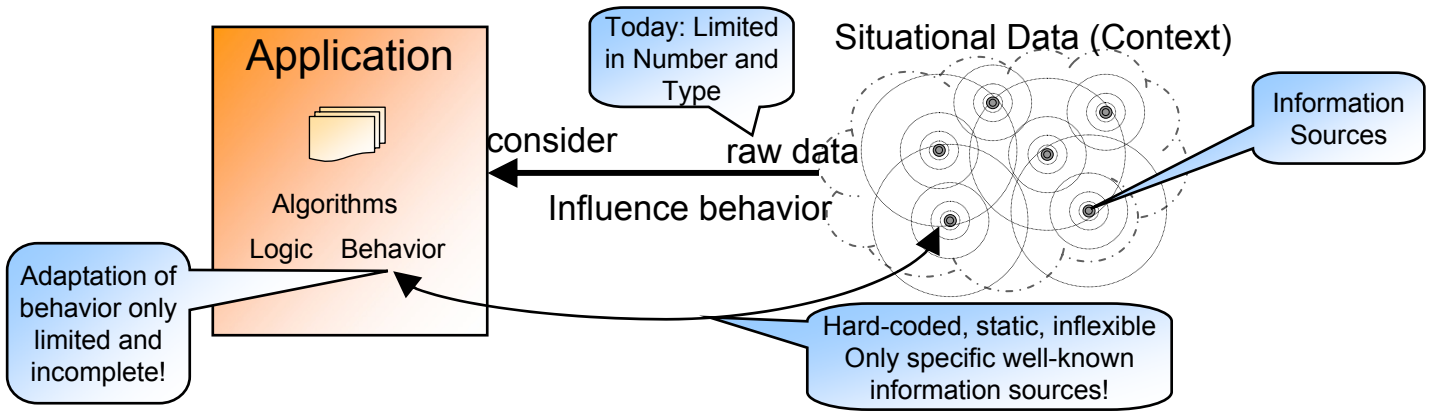
Appl execute certain algorithm (program logic) → static, non-adaptive



- **Adaptive Applications**
  - Consider situation and user data (→ contextual information)
    - Static information, such as general user preferences and application parameters, but more important are
    - Dynamic real-time data ➔ known as **Context**
  - **Adapt** behavior to them **like living beings do**
  - "Intelligent" / "smart" behavior
  - Users prefer **Adaptive Applications** over non-adaptive, static, dense applications because they behave **natural** and (up to a certain degree more) **intelligent!**

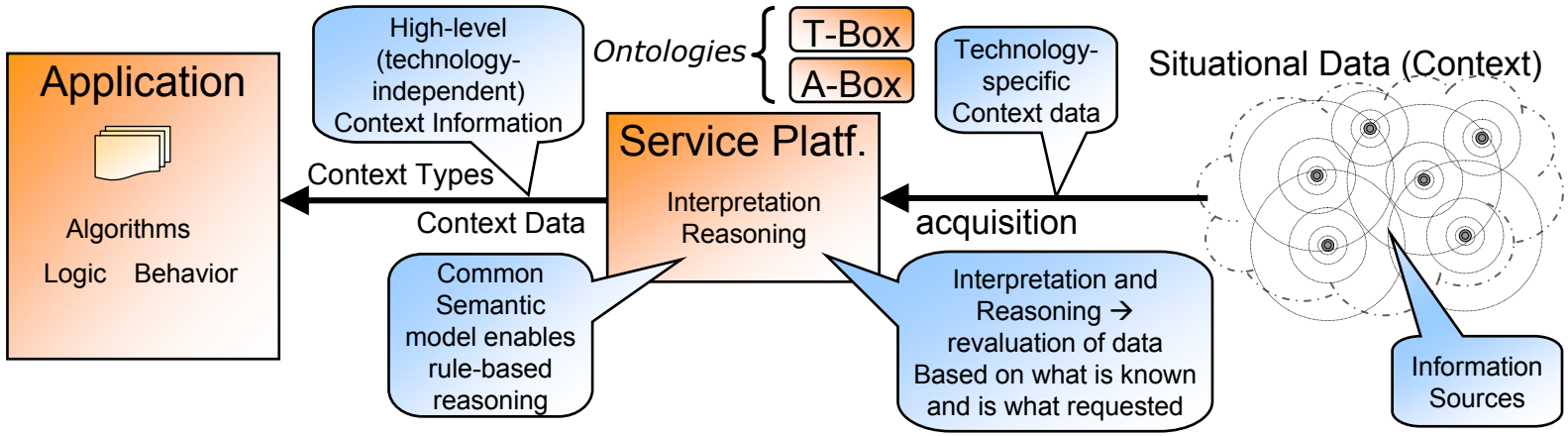
Current Adaptive Applications / Smart Applications

- Nowadays ➔ Adaptive applications consider only a limited number of situational data
  - Directly built into the application logic → static and inflexible
  - There is no semantic interoperability among different applications
  - Lack of optimization potential, all relevant raw data must be transferred to application
  - Adaptation solely implemented in the application code



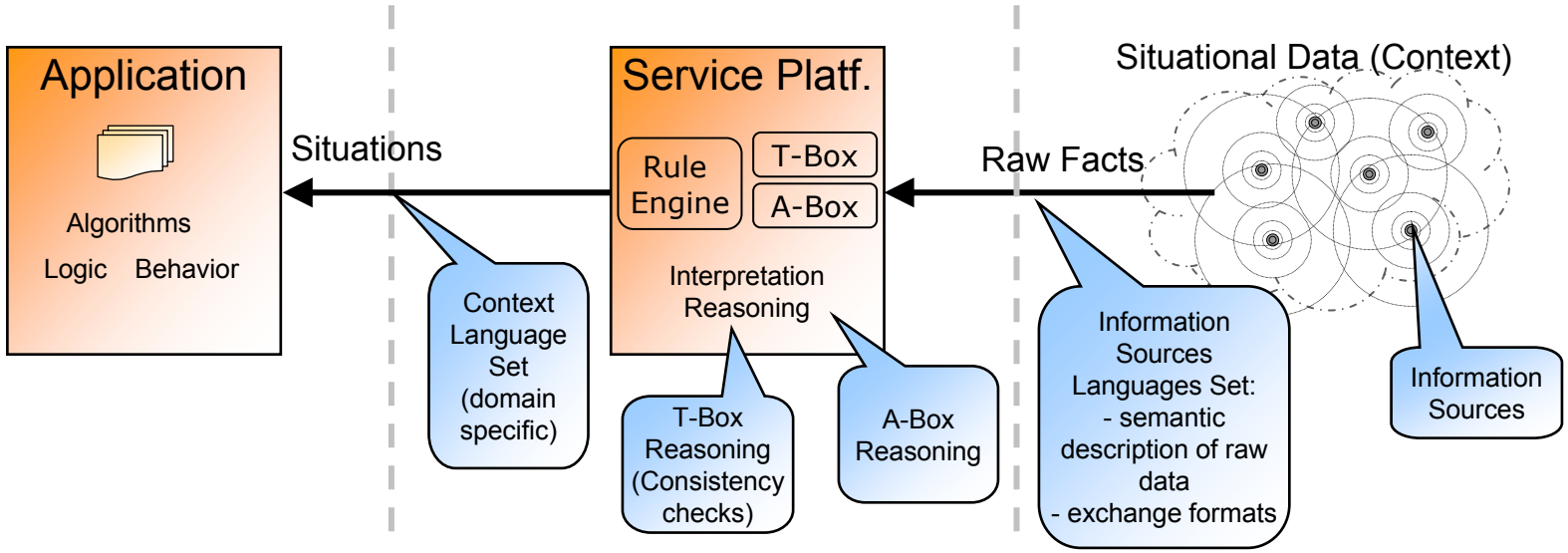
- Today: **Bottom-up approach**: Which available data can be considered (based on available technologies etc.)
- Instead of: **Top-down approach**: Which data (contextual information) is essential and desired! → semantic description of relevant contexts, independent of concrete information sources

- **Top-down approach**: Application specify the **context types**, which is relevant for reasonable service adaptation without limiting the specific technology!
  - independent from any specific type of information source!
  - using “real-world” descriptions
- **Ubiquitous Service Platforms** will translate and mediate between application and information sources by **Interpretation** and **Reasoning**

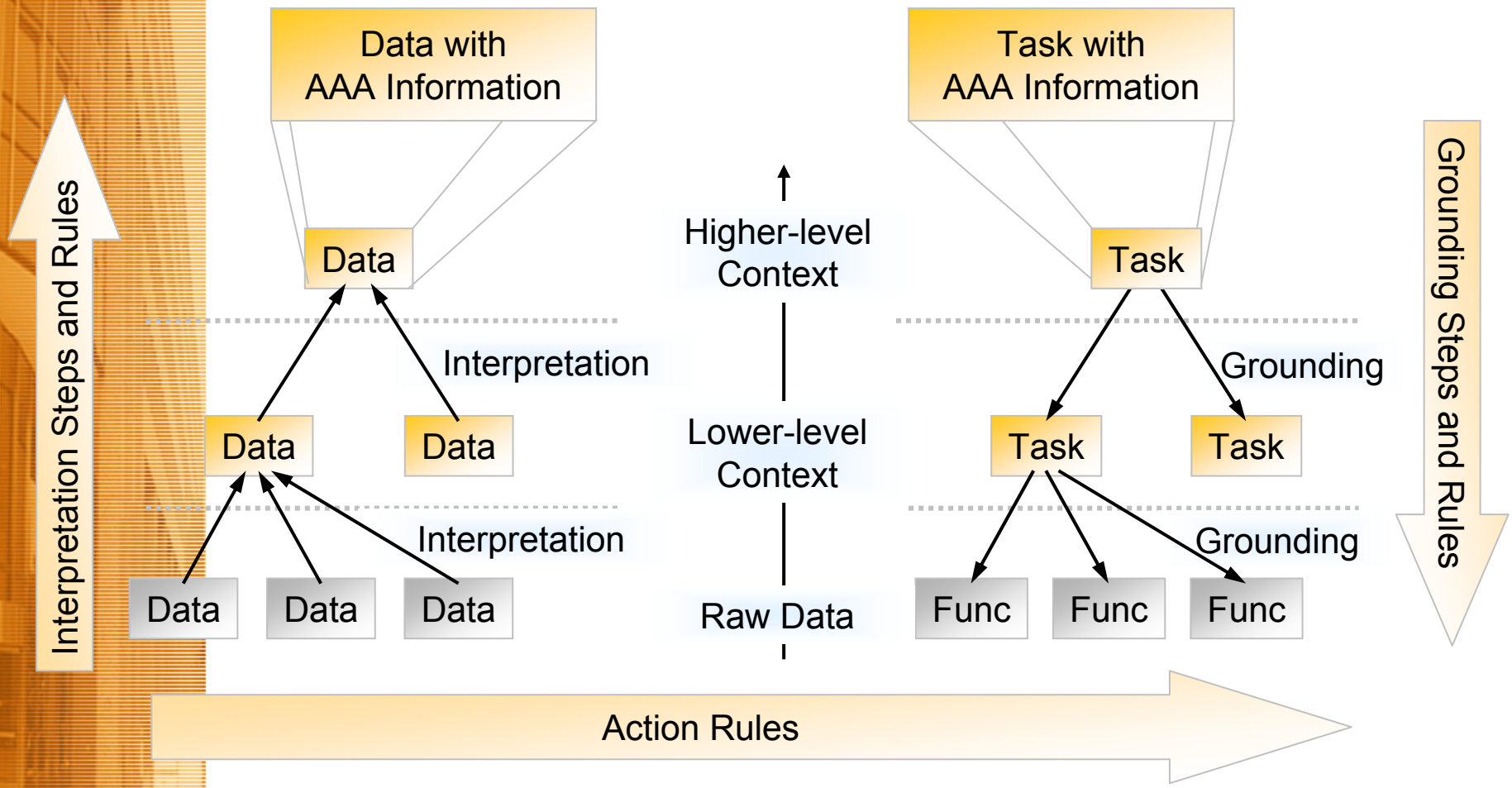


- Application developers do not have to implement technology support
- Instead they can specify which contextual data of interest (in a top-down approach)
- The Service Platform knows how these requested data can be derived and extracted from available information sources
- Contextual framework of sets of chained rule-based context interpreters → easy to manage and extensible because of using semantic models

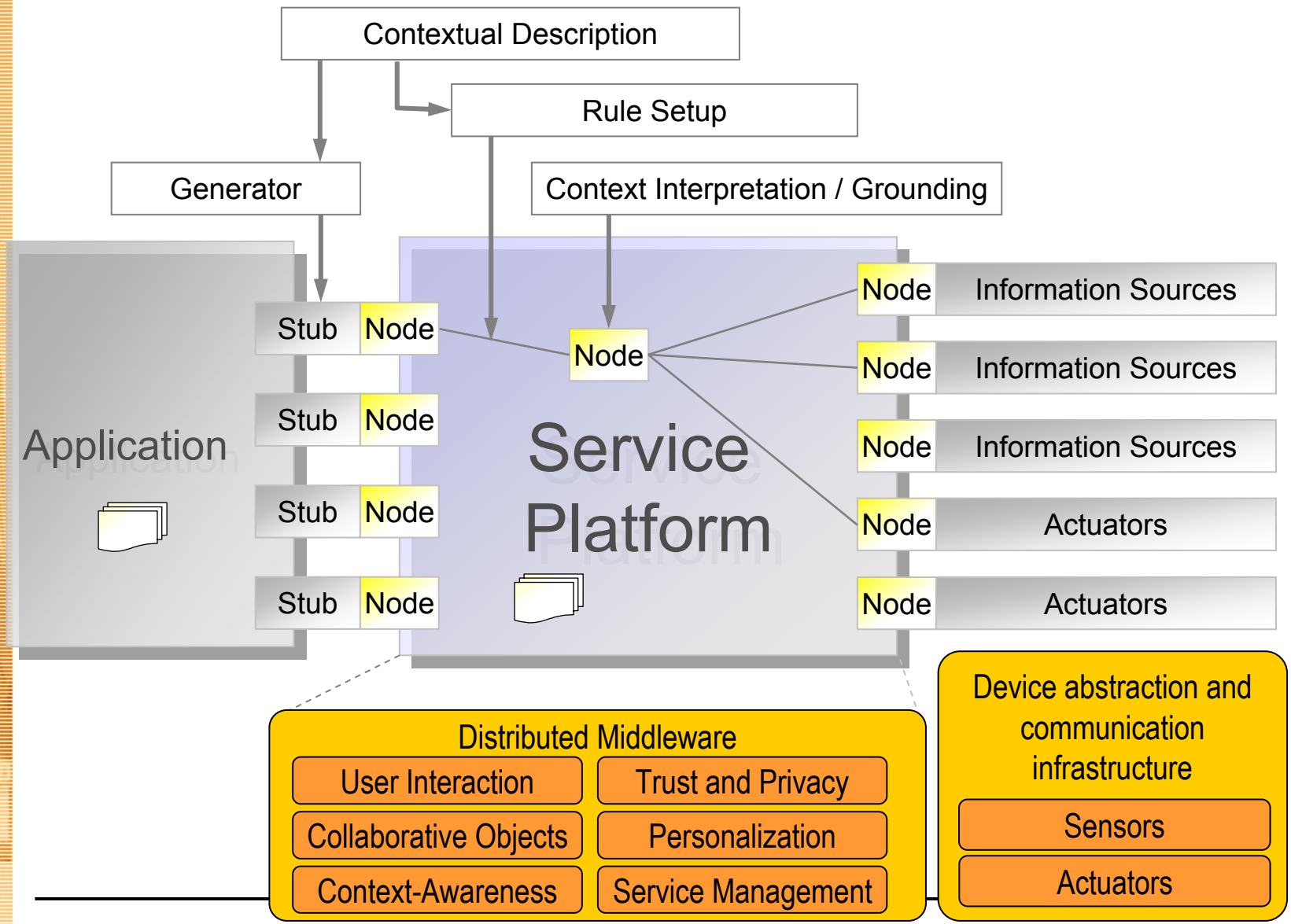
- Application specify **what** is of interest (→ subscribe)
- Information sources specify **what** can be provided (→ publish)
- Reasonable interpretation paths → only relevant information is gathered from information sources
  - Efficient
  - Need for languages and models to describe semantically
    - the application's requested context and
    - the information sources



Context Acquisition & Context Establishment



Service Platform supporting Rule-based Programming



Thank you for your attention!

stephan.steglich@fokus.fraunhofer.de