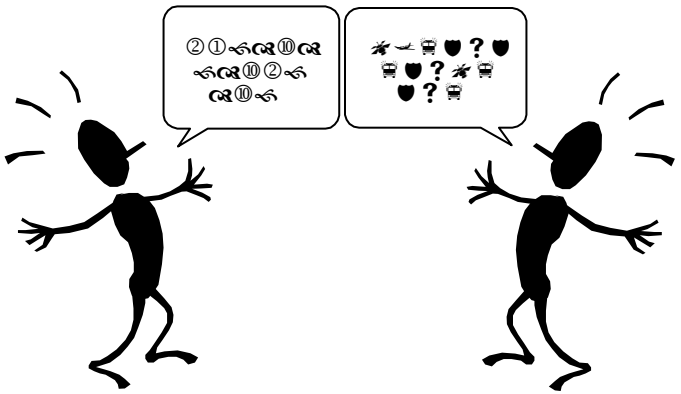
SDS R3.1
An Overview



Philippe Spaas
Executive IT Architect
July 24th 2012

- **What Is The Purpose Of This Presentation?**
  - This document is meant to provide a general overview of the System Description Standard (SDS). It is not intended to provide a formal definition of the SDS or a full description of the meta-model. This document should be sufficient for most people who just want to understand why we need SDS and what the main concepts are which are defined in the SDS.
- **Who Should Read This Document?**
  - The primary audience for this document is IT architects and IT specialists who either want to familiarize themselves with the key concepts used in describing systems or want to read up on the SDS concept definitions as part of their preparation for attending a course from the IT Architecture curriculum (e.g. architectural thinking, component modeling, operational modeling, …).
- **When Should This Document Be Used?**
  - This document is an introduction to the core concepts embodied in the SDS. Therefore, it is the first document you should read before exploring other sources of SDS information.

Agenda

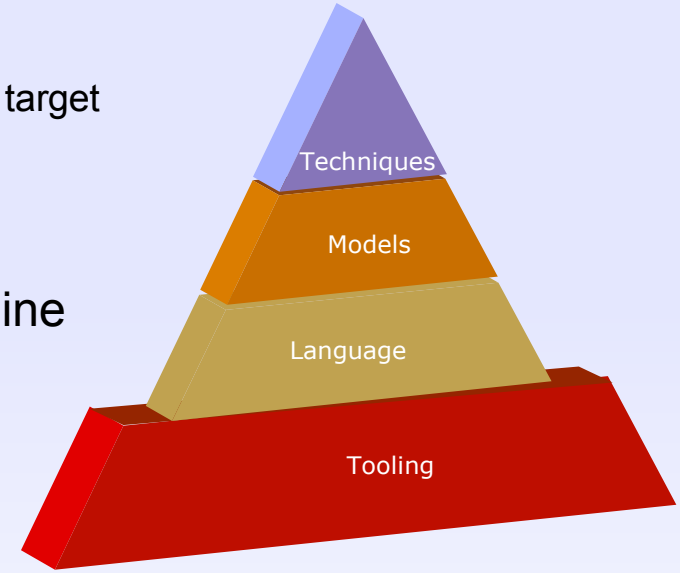
- **Our challenge: Help addressing the systems' engineering challenge through establishing**
  - **A common language**
  - **A single model of the system**
- Applicability of the System Description Standard
- Introduction to the SDS concepts
- Frequently asked questions
- SDS Reference Material
- References

The system engineering process, starting from requirements and working towards the delivery of a system, is a complex process involving many different stakeholders.

The system engineering process relies on a number of standardized artifacts in order to bridge the gap between the requirements and a delivered target system.

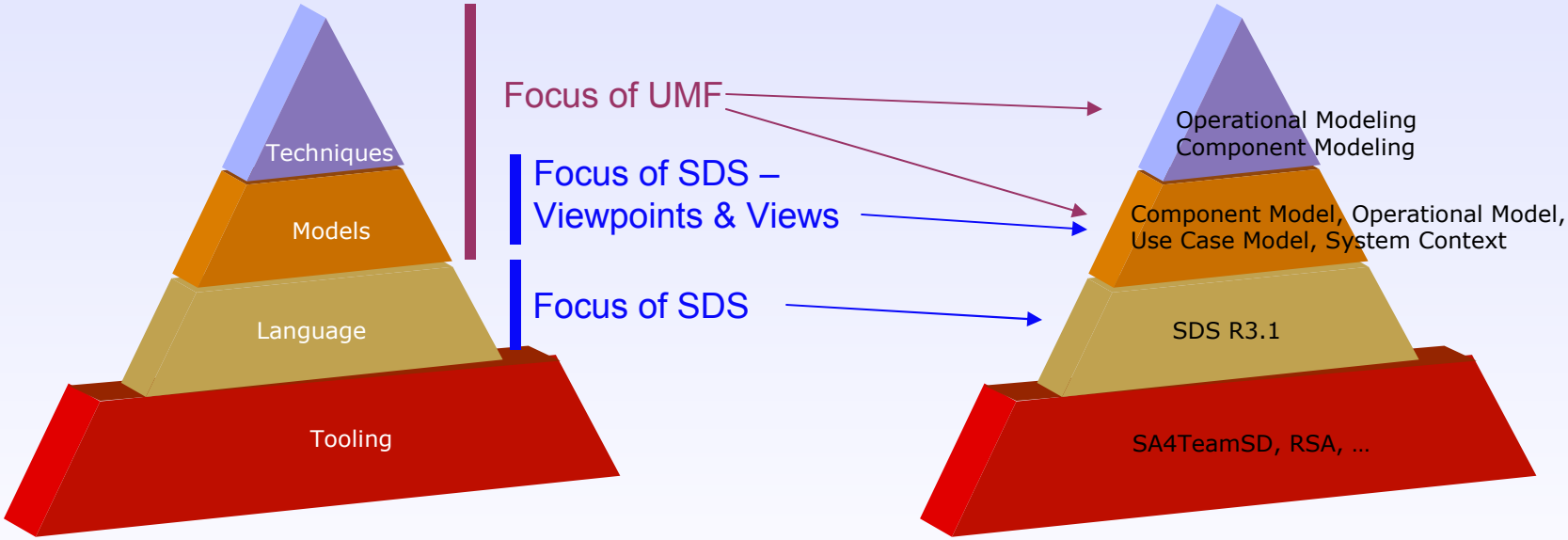
- These artifacts and associated guidance are defined in the Unified Method Framework (UMF)
  - These are used as part of a stepwise approach to deliver the target system
  - They support the required coordination between the involved disciplines
- Standardizing artifacts, guidance and cross-discipline coordination implies agreement on
  - Techniques, applied to
  - Models, expressed in a
  - Language

  whereby a common language is a pre-requisite for common models which are a pre-requisite for integrated techniques.



SDS focuses on standardizing the language used during the system engineering process and, the context within which models, relevant for this process, are defined.

- SDS
  - Defines, together with UML2, all concepts and their relationships required to describe systems
  - Extends the notions of viewpoints and views, as defined in the IEEE 1471, to be able to support the views required by the stakeholders



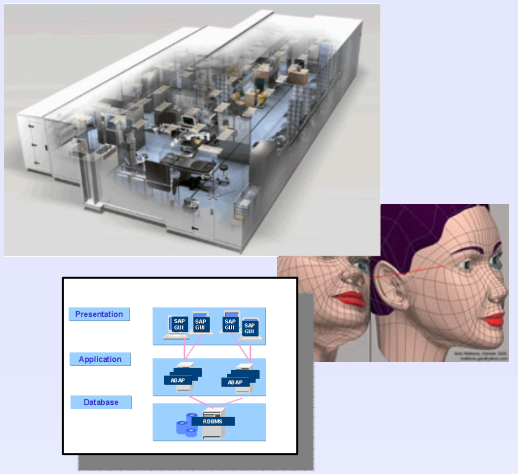
Why did we feel the need to introduce a *new* language instead of adopting an existing one ?

- Although UML2 was taken as the starting point for the definition of ADS/SDS, it required enhancements in specific areas in order to support the artifacts and associated guidance of the UMF. These enhancements mainly concerned the need to be able to model
  - Deployment Units
  - Locations
  - Nodes
  - Non-functional requirements
  - Users
  - Zones
- The advantage of introducing a new language is that there is no excess luggage, i.e. only those concepts and associations which directly serve the purpose of the language are present. Hence, it is much easier to convey the relevant set of concepts rather than having to point out specific interpretations of generalized concepts belonging to a generic modeling language. This however does not exclude building profiles to express SDS's concepts and relationships through a specific UML2 based profile.

Coordinating across disciplines and consistently addressing stakeholders' requirements relies on the use of a single model of the system, i.e. a System Model, expressed in a single language.

An IT system

described via



The System Model

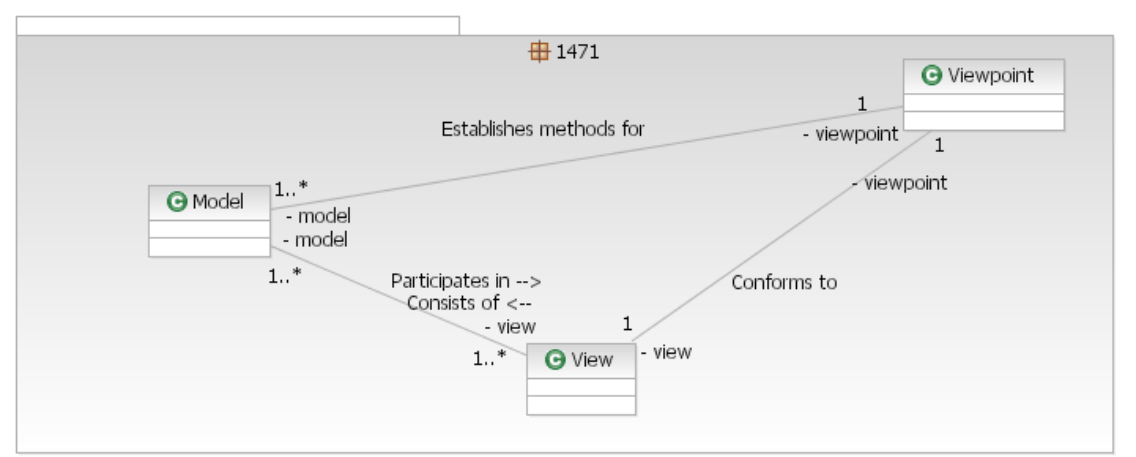
- The System Model
  - is expressed in a single language, SDS
  - contains all model elements and model element relationships which are relevant for our system engineering effort,
    - Implying that the System Model contains the relevant parts of the target system and external systems including their relationships.

    and these elements are present in the System Model for every step of their individual engineering cycle
    - Implying that e.g. logical nodes and their physical implementation (described via physical nodes) are both part of the System Model.

During the system engineering effort, practitioners of different disciplines and stakeholders are interested in different subsets of the System Model.

- As an example, business analysts will only be interested in requirements related model elements, and they will add/detail requirements related elements to/in the System Model. These model elements then become available to others.
- Standardizing engineering activities implies we need agreement on the views on the overall System Model to be used for a particular engineering activity, hence we define views on the basis of
  - a specific focus
    - In terms of skills (requirements, component modeling,…)
    - Defining overall approach/structure (architecture) vs. detailed considerations
  - at various points of the engineering cycle
    - E.g. logical (no decision on technology/product taken) or physical (decision on technology/product taken)
  - filtered according to different criteria (e.g. application vs technical)

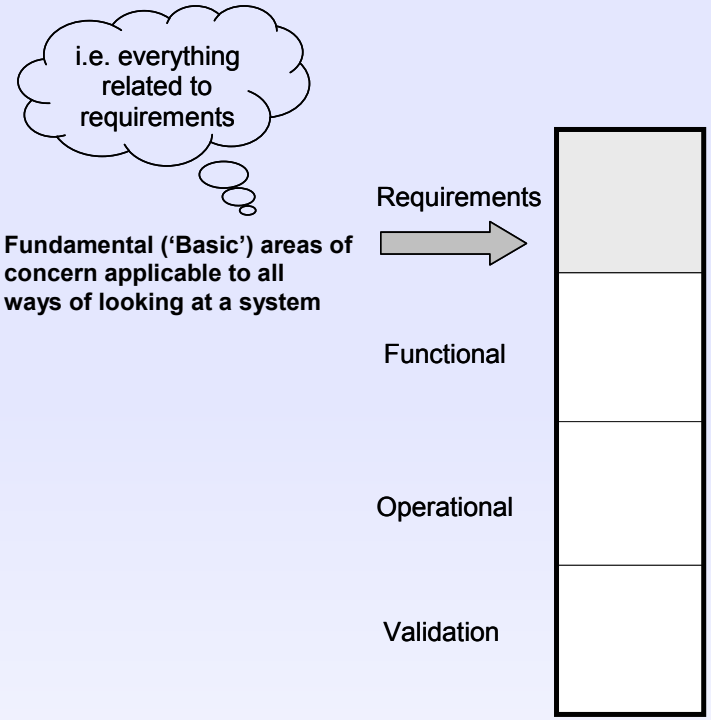
In order to support the definition of all required views on the single underlying System Model, we extended the open standard for the description of software intensive systems (IEEE-1471/ ISO-42010).



- IEEE 1471 defines the following concepts:
  - Viewpoint: defines the set of related stakeholder concerns and contains conventions for the creation and notation guidance supporting the creation of views adressing these concerns
  - View: refers to a set of models which have been created according to the rules defined in the applicable viewpoint
  - Model: is a collection of model elements which addresses a particular set of concerns

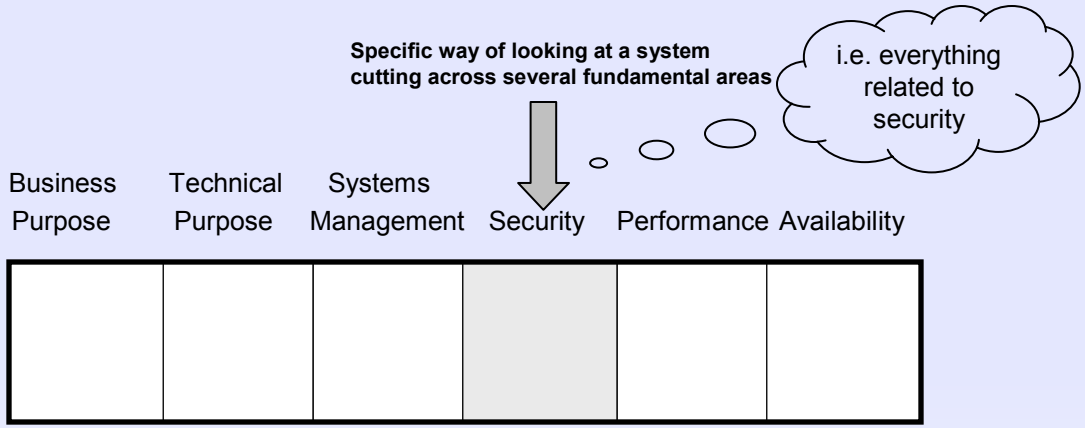
- IEEE 1471 is extended by:
  - Emphasizing that the 'Model' refers to a subset of the overall System Model
  - Introducing two different kinds of viewpoints derived from the IEEE 1471 Viewpoint:
    - Basic viewpoints
    - Cross cutting viewpoints

SDS' Viewpoints & Views introduces four basic viewpoints (BVP) upon which all stakeholder views of the System Model are based.



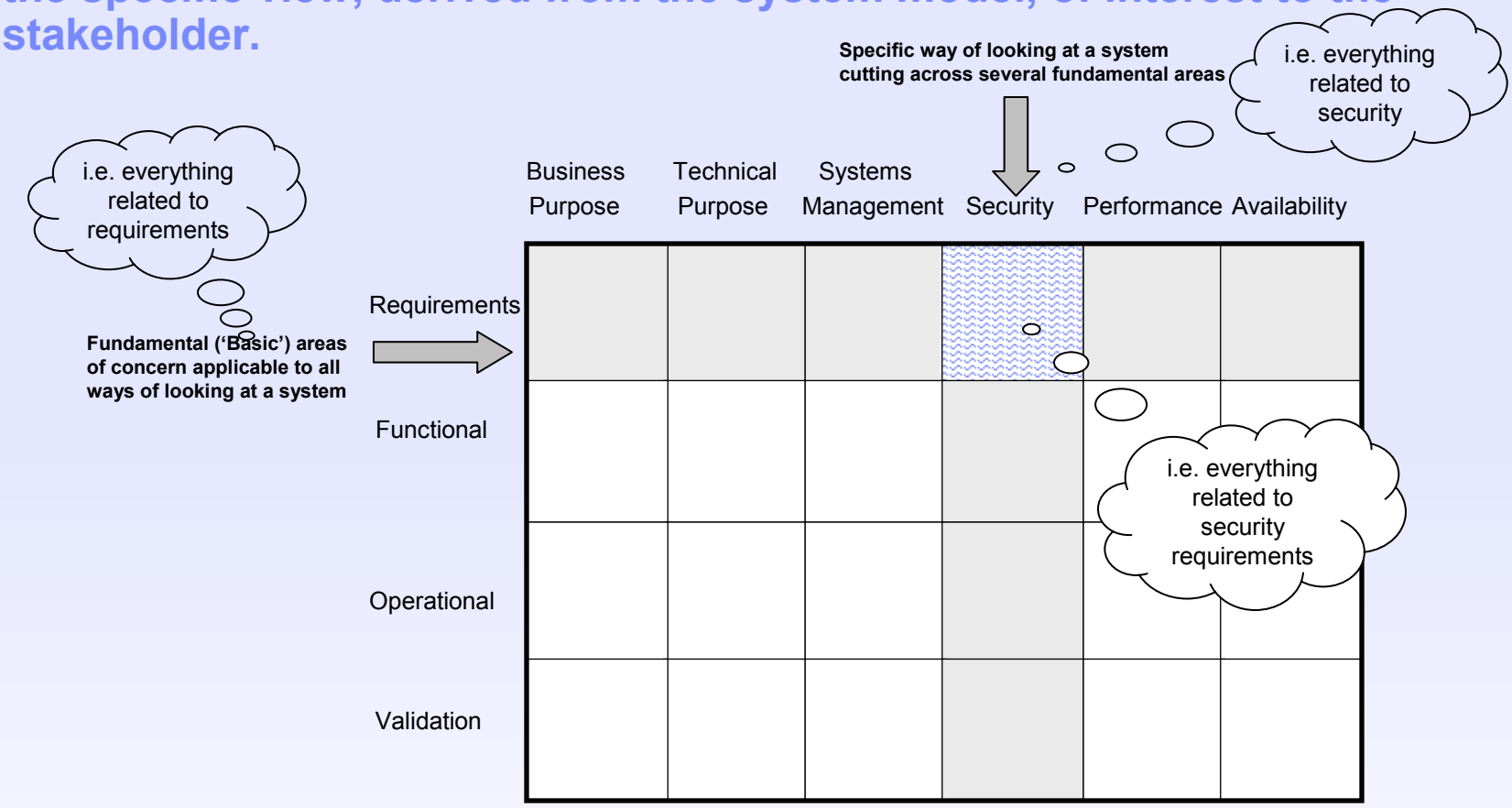
- The basic viewpoint defines the model elements/model element relationships from the System Model which are relevant for that particular viewpoint.
- The four BVP's defined by SDS are:
  - Requirements BVP: focus on the concepts involved in the modelling of system requirements
  - Functional BVP: focus on the structural elements from which the system is built and their relationships
  - Operational BVP: focus on how the target system is built from its structural elements and integrates into its environment
  - Validation BVP: focus on the concepts related to assessing whether a system will deliver its intended functionality with the expected quality of service

SDS' Viewpoints & Views identifies some candidate cross-cutting viewpoints (XVP) which can be further tailored to specific stakeholder needs.



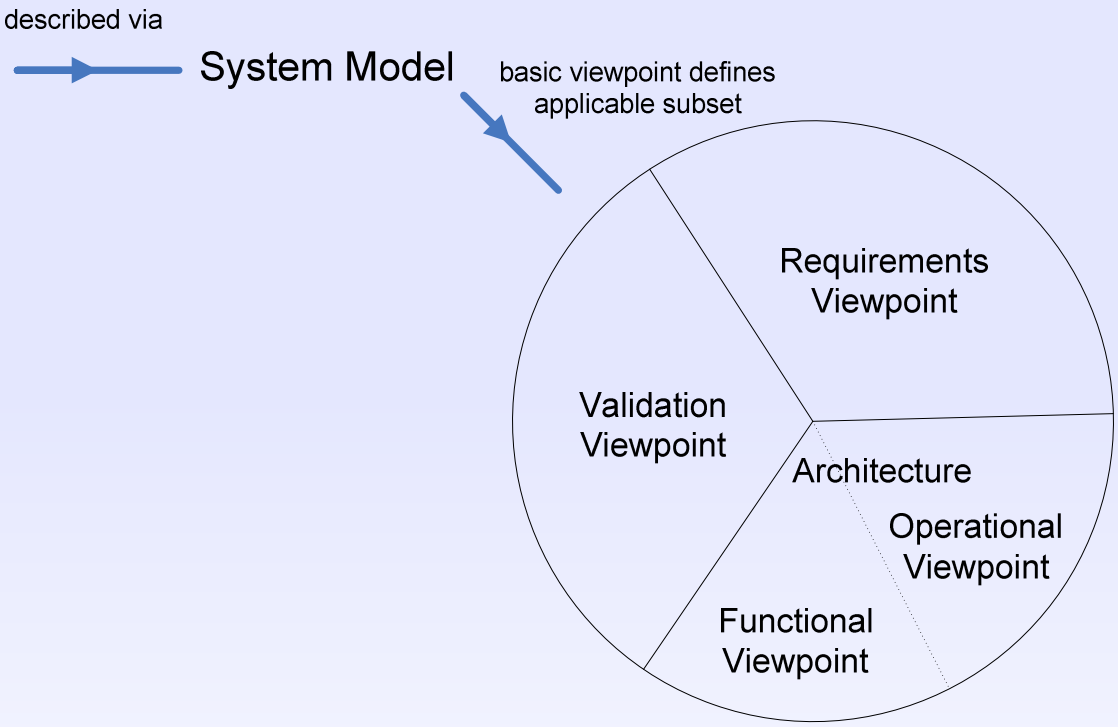
- The cross-cutting viewpoint defines the model elements/model element relationships from the System Model which are relevant for a particular stakeholder. SDS identifies a number of candidate XVPs which can be further extended as needed:
  - Business Purpose XVP: focus on all business purpose related concepts
  - Technical Purpose XVP: focus on all the supporting concepts (i.e. not directly related to the business purpose)
  - Systems Management XVP: focus on all systems management related concepts
  - …

The intersection of the relevant basic and cross-cutting viewpoint defines the specific view, derived from the system model, of interest to the stakeholder.



- During the engineering cycle, a practitioner skilled in security will construct the security requirements view (based on the System Model) relying on:
  - Generic modeling techniques (i.e. applicable to the models created for the requirements basic viewpoint)
  - Security specific modeling techniques and know-how (extensions to the generic modeling techniques, specific to the security cross-cutting viewpoint)

The description of SDS itself is structured according to the standard set of basic viewpoints.



- Concepts are defined within the basic viewpoint with which they have become associated through custom and practice.

Agenda

- Our challenge: Help addressing the systems' engineering challenge through establishing
  - A common language
  - A single model of the system
- **Applicability of the System Description Standard**
- Introduction to the SDS concepts
- Frequently asked questions
- SDS Reference Material
- References

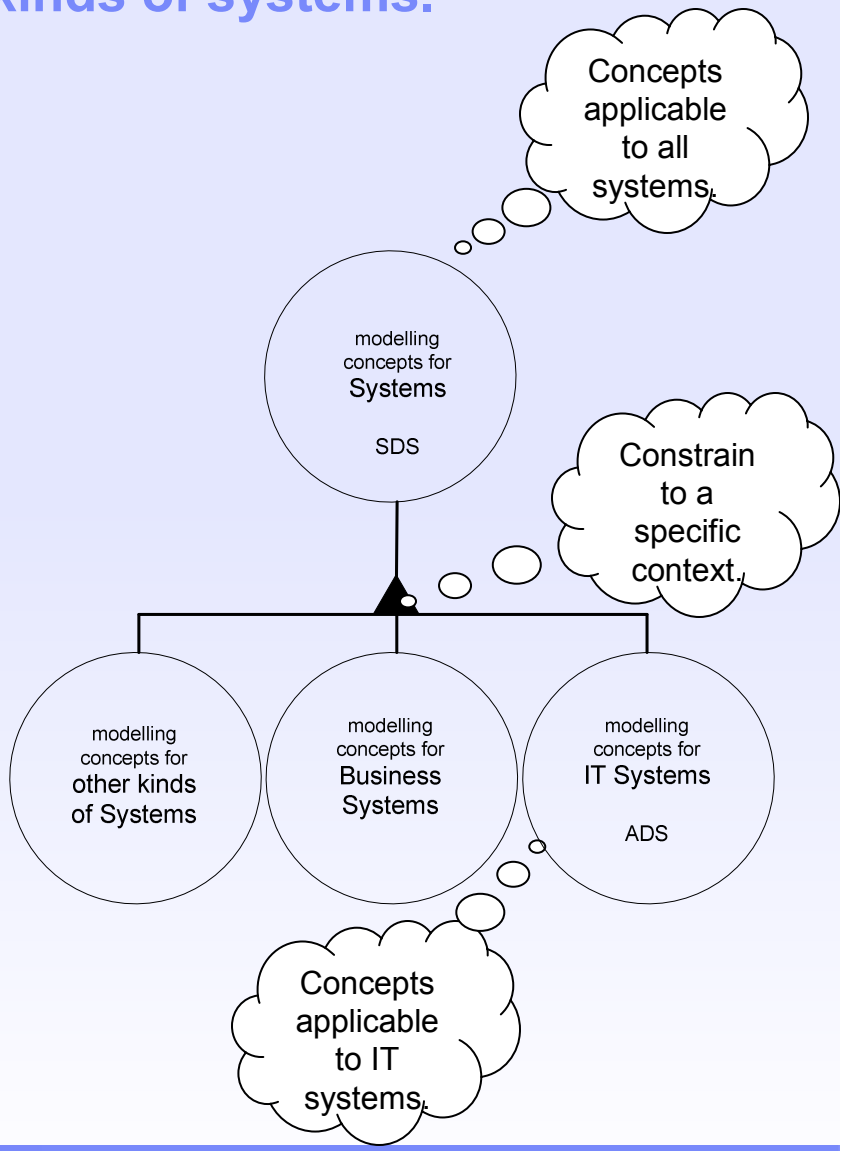
SDS has the objective of defining a concise language for the description of systems addressing the needs of all involved stakeholders.

- Addressing the needs of all stakeholders implies also covering models supporting both enterprise and solution architecture.
- SDS tackles these two potentially conflicting objectives of conciseness and broad coverage through
  - the flexible use of its concepts by supporting
    - different kinds of systems
    - solution and architecture building block models
    - engineering perspectives
  - supporting stakeholders through a single model (System Model)

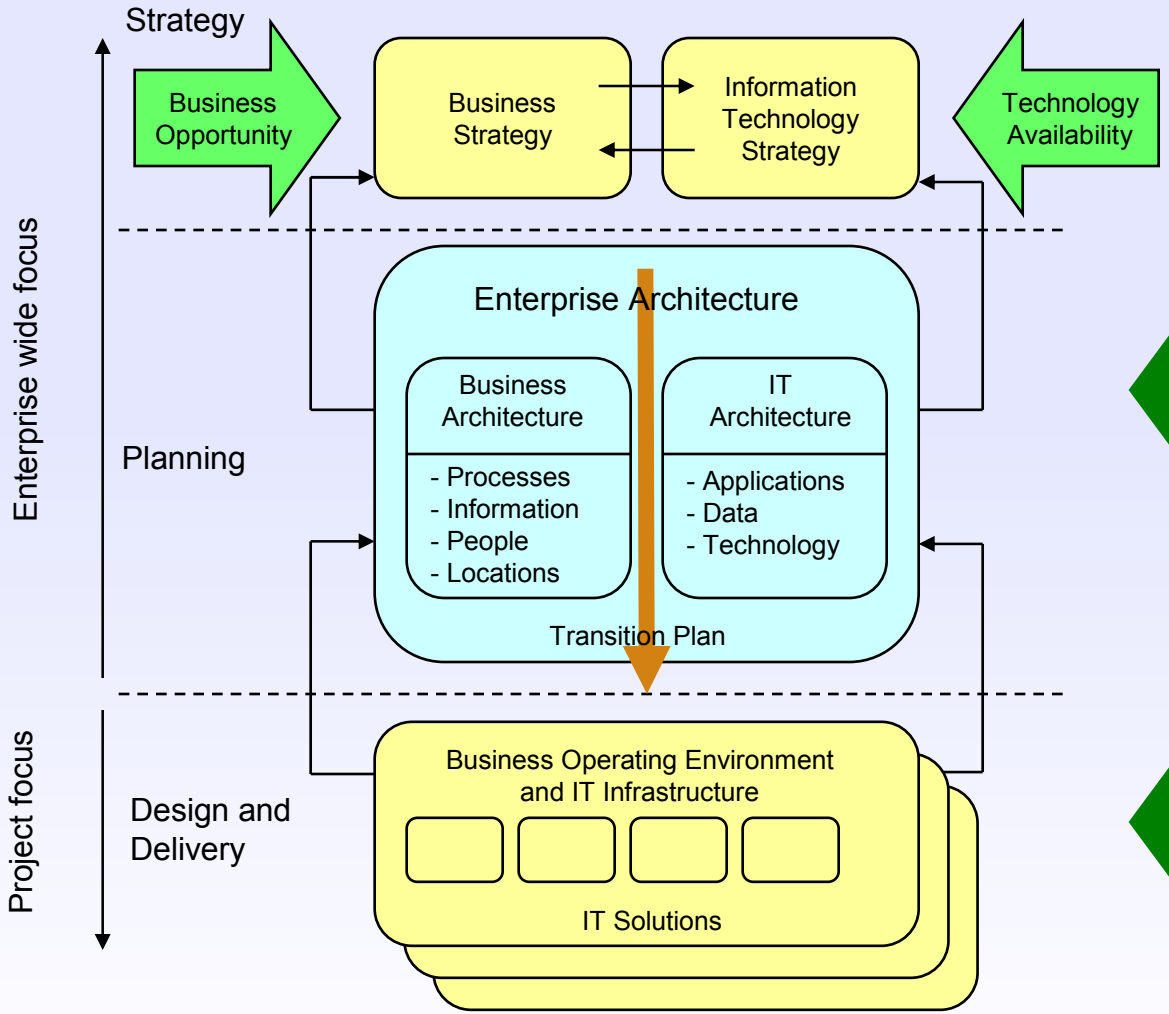
The conciseness of SDS is supported by focusing on generalized concepts applicable to many different kinds of systems.

- The SDS modelling language supports the description of the architecture of systems, irrespective of the actual type of the system, i.e. it supports the description of a (general) system, an IT system,…
- The specific interpretation of the concepts will vary slightly across contexts
  - e.g. the notion of a component delivering services in a business system refers to the combination of human, IT and other resources, while in an IT system it refers to a combination of function and data

nevertheless these concepts express the same basic idea (in this example a collection of resources delivering services) which remains relevant and useful across different contexts.



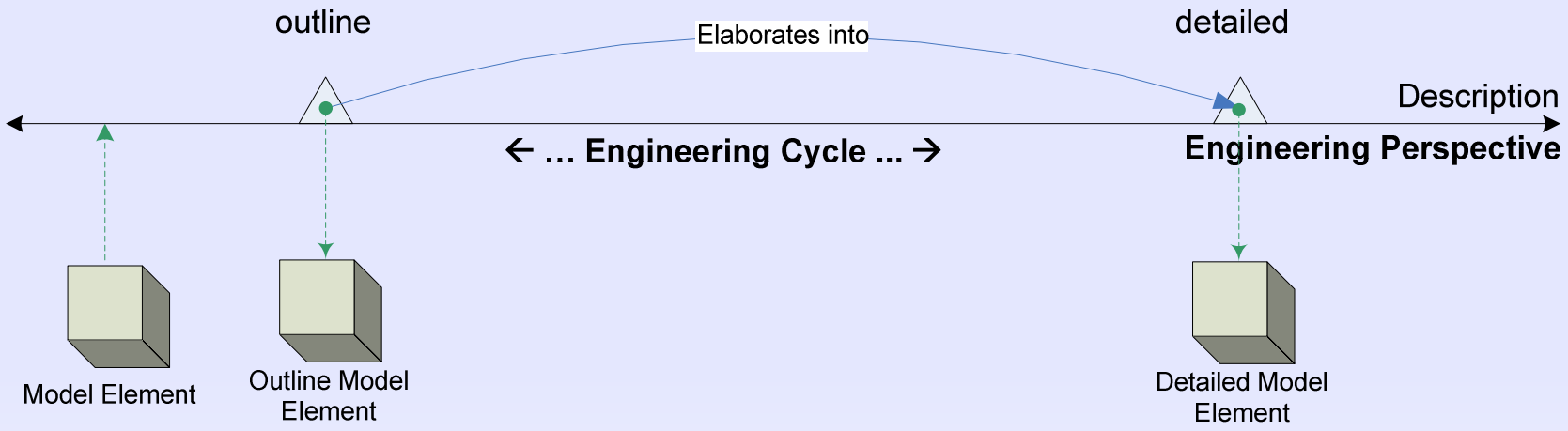
We use a single language to address the needs of enterprise architecture (building block oriented models) and solution architecture (solution models).



Enterprise Architecture
a focus on the description of a 'catalogue of parts' documenting the parts from which systems and their models can be built (architecture building block models)

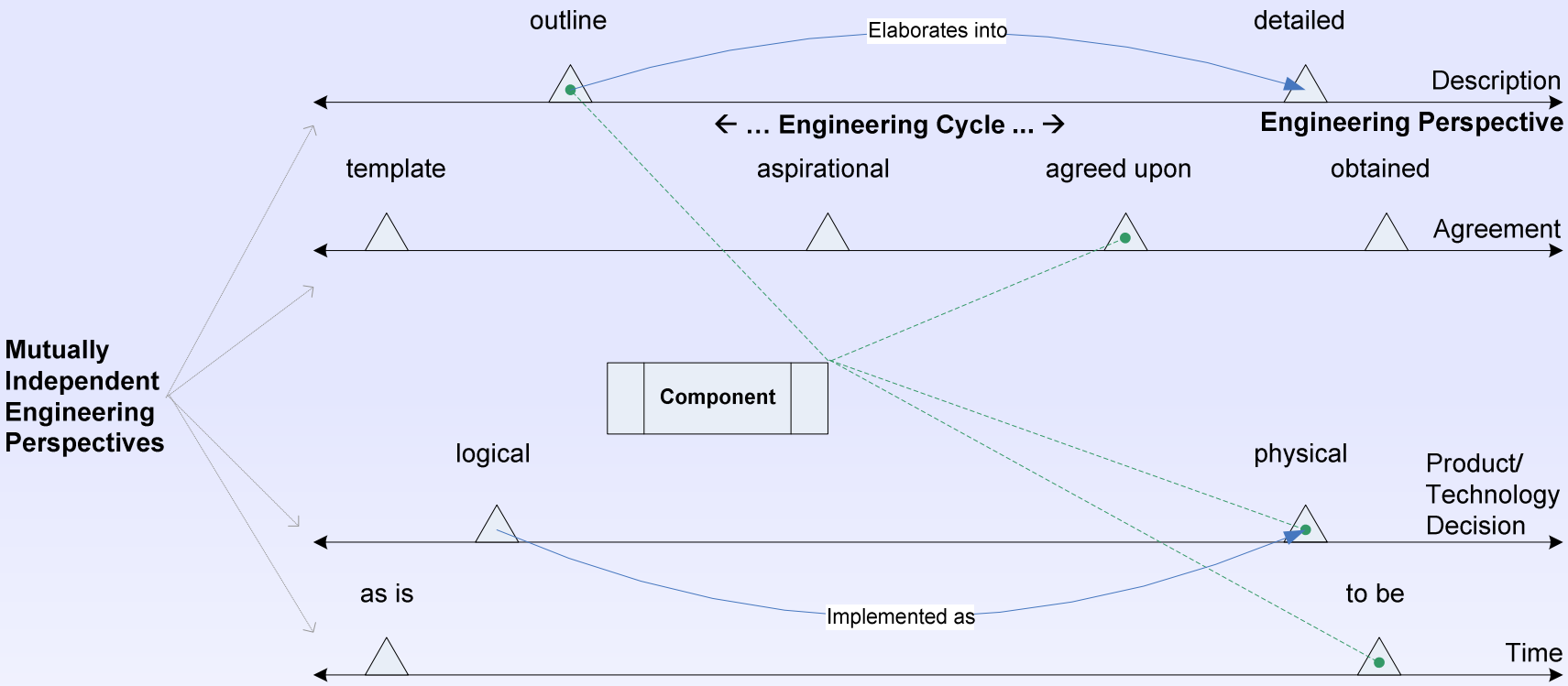
Solution Architecture
the models supporting the engineering of a single system (system solution models)

SDS Engineering perspectives allow us to distinguish model elements at different points of the engineering cycle without introducing ad hoc names.



- During the course of the engineering of a system model, elements from the system model evolve as they reflect the results of the engineering effort, e.g. product & technology selections are made, model elements are progressively detailed, ….
- An ‘engineering perspective’ limits our focus to (a) specific (set of) attributes of a model element e.g. level of detail of documentation, product/technology decision taken, …
- To be able to position concepts at different points along the engineering perspective, we will explicitly qualify them (e.g. we can distinguish between an ‘as-is’ component vs. a ‘to-be’ component).
- Note that viewpoints allow us to define subsets of the system model relevant for a particular stakeholder, while engineering perspective allow us to track individual model elements.

Model elements can be considered simultaneously from multiple engineering perspectives.



- Example:
  - An agreed upon outline of a physical to-be component

Agenda

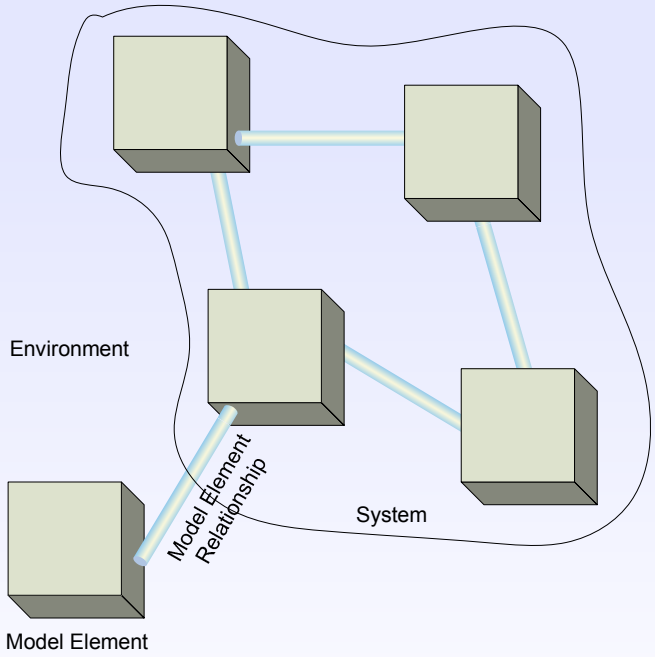
- Our challenge: Help addressing the systems' engineering challenge through establishing
  - A common language
  - A single model of the system
- Applicability of the System Description Standard
- **Introduction to the SDS concepts**
- Frequently asked questions
- SDS Reference Material
- References

Concerning the introduction to the SDS concepts

- The following slides present an overview of the main SDS concepts. This section focuses on the explanation of the concepts whereby
  - Only the most important concepts have been included
  - (abstract) Base classes used in structuring the overall SDS model have been omitted
  - The notation has been chosen in support of this particular purpose and is not the notation as would be used in a project (nor would we recommend this notation for use on a project !)
- We will focus on the generalized concepts, i.e. the concepts applicable to systems in general. Whenever we use a notion which is specific to an IT System, we will explicitly indicate it.
- For a more formal discussion of these concepts and their relationships, we refer to the SDS Semantic Specification. This document also provides the SDS metamodel documented by means of UML2 models.

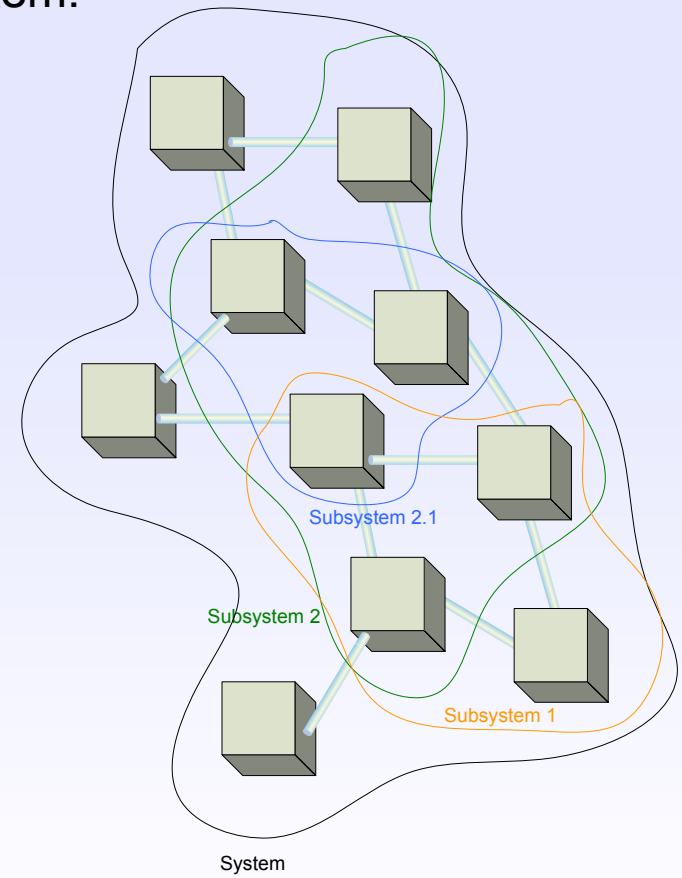
The notion of a System.

- A system is a collection of model elements and model element relationships which taken together demonstrate some behaviour within an environment. This behaviour is apparent through the interactions the system has with its environment.



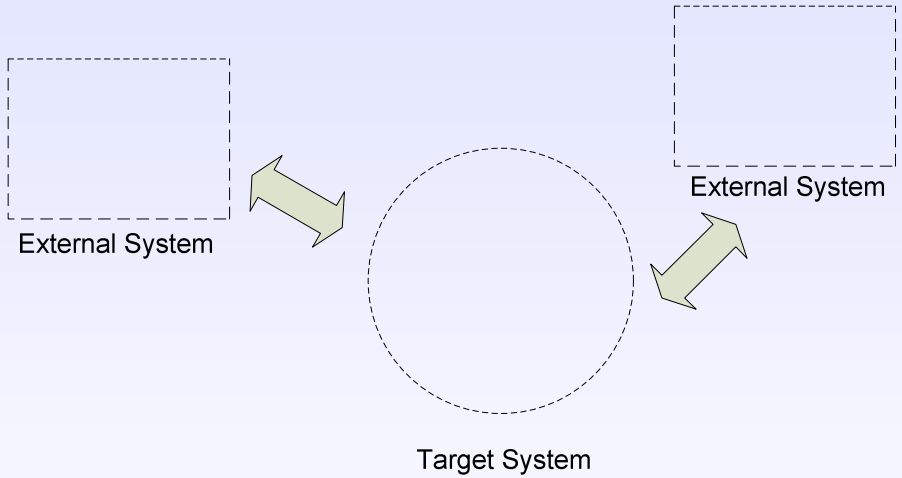
Subsystems are arbitrary collections of the model elements/model element relationships of a system.

- Subsystems of a given system can overlap and can further decompose an already defined subsystem.



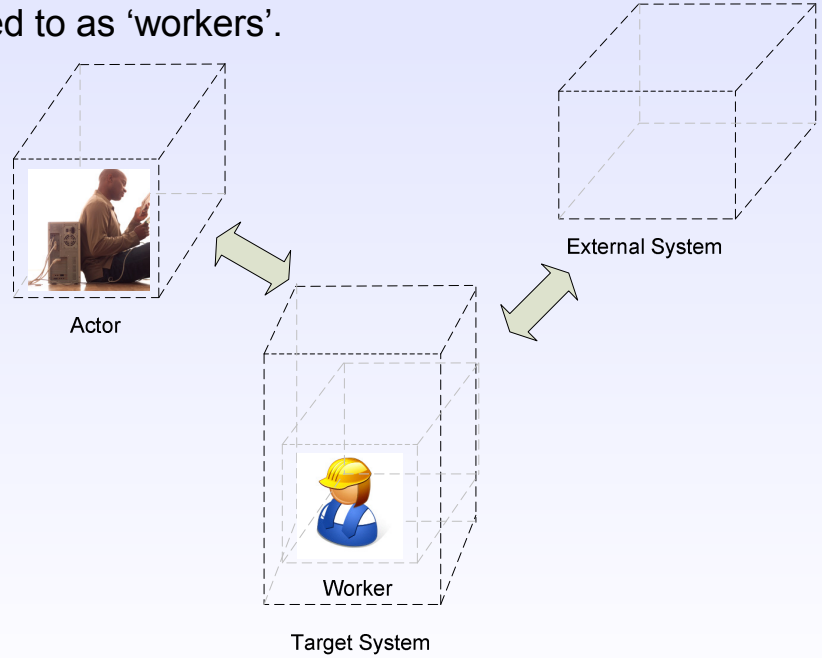
The boundary of the target system identifies both the system on which we focus and the external systems.

- The labeling of a system as being the ‘target system’ indicates both our specific focus and the scope within which we can decide on the model elements and their relationships.
- The key parts of the environment with which our target system will interact are referred to as ‘external system(s)’.



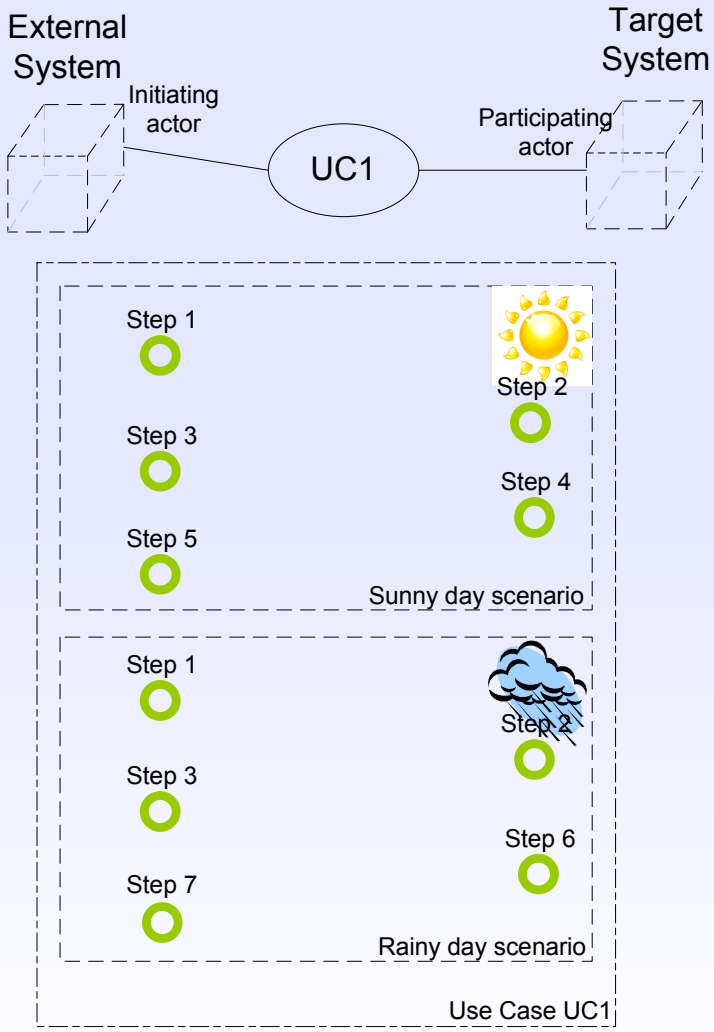
People are a special kind of system and are looked upon differently depending on which side of the system boundary they find themselves.

- We draw a distinction between people
  - interacting with a target system, i.e. part of the environment. They are modeled as an external system in their own right, referred to as ‘actors’ of we deal with logical external systems, or as ‘users’ when we deal with physical external systems..
  - being part of the target system, i.e. people of whom the tasks and responsibilities are determined through the engineering effort. These people are considered as subsystems of the target system, and are being referred to as ‘workers’.



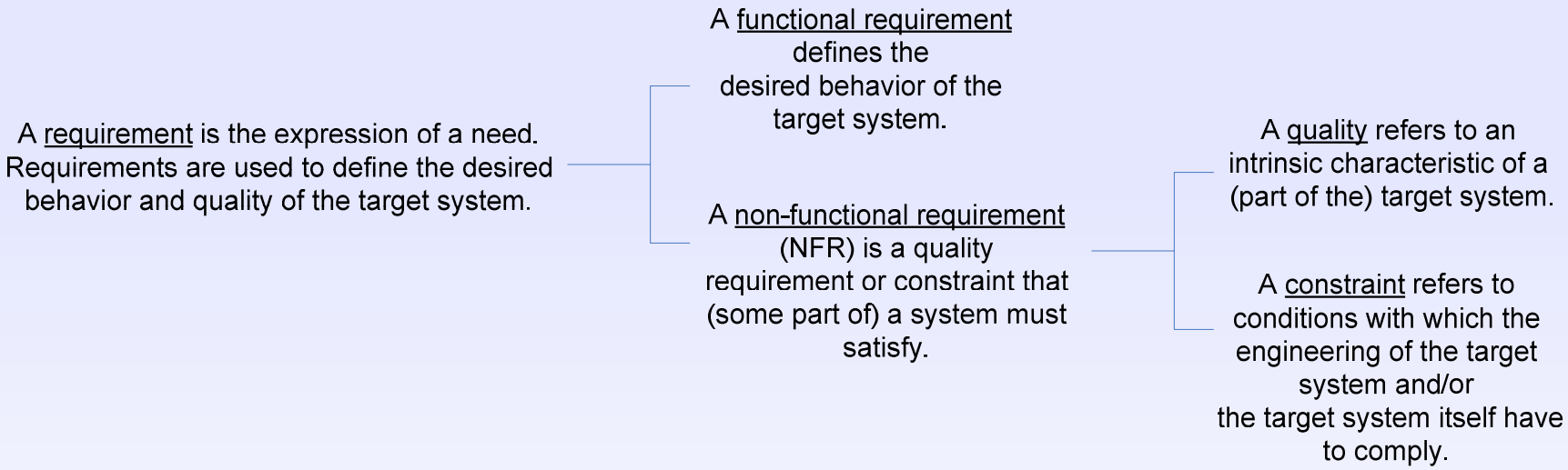
The expected behavior of both the external and target systems are described through use cases and scenarios.

- A use case is an identifiable and externally observable behaviour of (a part of) the target system.
  - Use Case UC1
- A scenario is the trace of an execution of a use case under well specified circumstances.
  - Sunny day scenario, rainy day scenario
- A step is an elementary piece of behaviour.
  - Step 1, step 2, …



External systems impose a variety of requirements upon a target system.

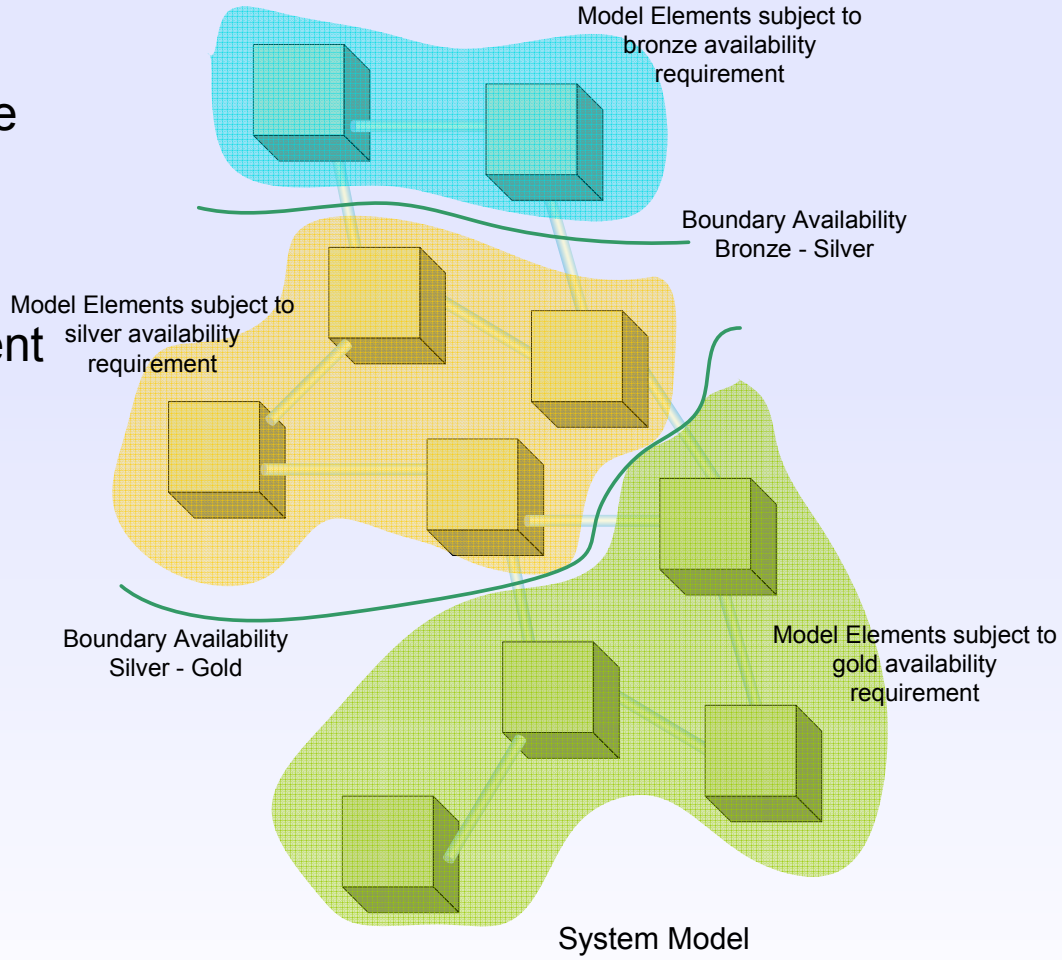
- It is customary to draw a distinction between :



- Each of the model elements/model element relationships present in the System Model can be associated with any of these requirements.

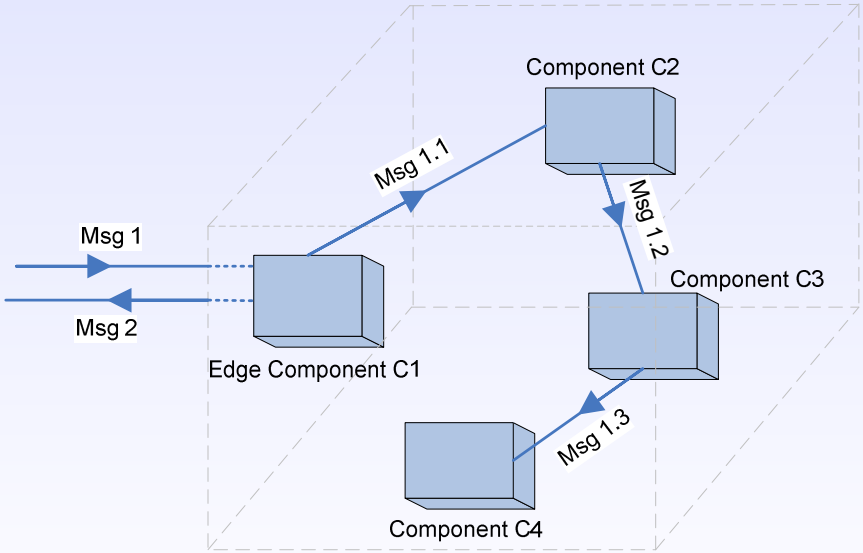
Common (range of) values for requirements can be used to organize the elements from the system model.

- Zones allow to make all model elements tagged with the same range of values for a given requirement explicit.
- Boundaries indicate where the range of values of a requirement changes.



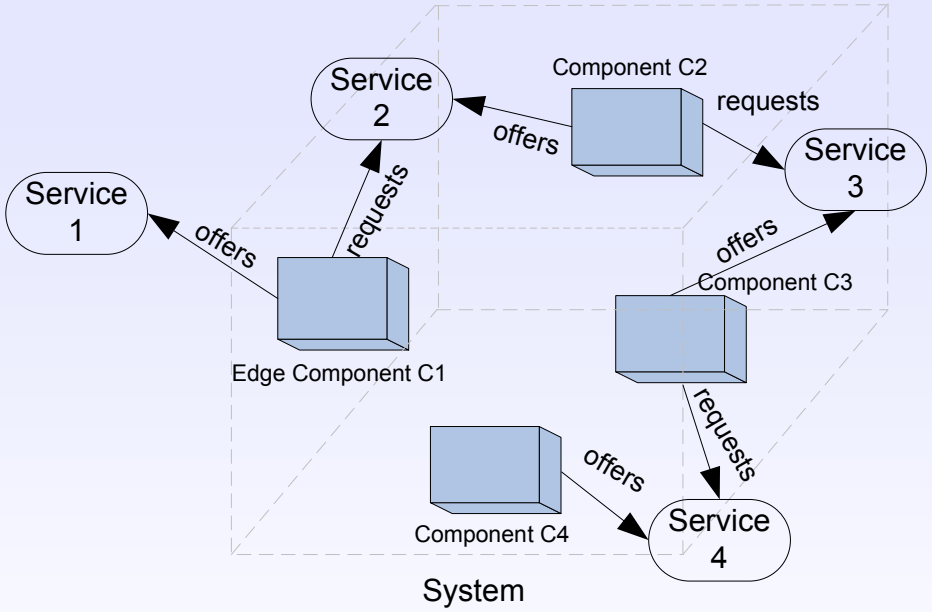
Systems demonstrate behavior through the involvement of their contained components.

- A system responds to an external message (i.e. message 1) received by an edge component through the involvement of a number of components via message sends. Each component represents a modular unit of functionality and the involved components contribute towards the system's behavior (in this case: sending back message 2).
  - An edge component is the first component receiving the message from the environment



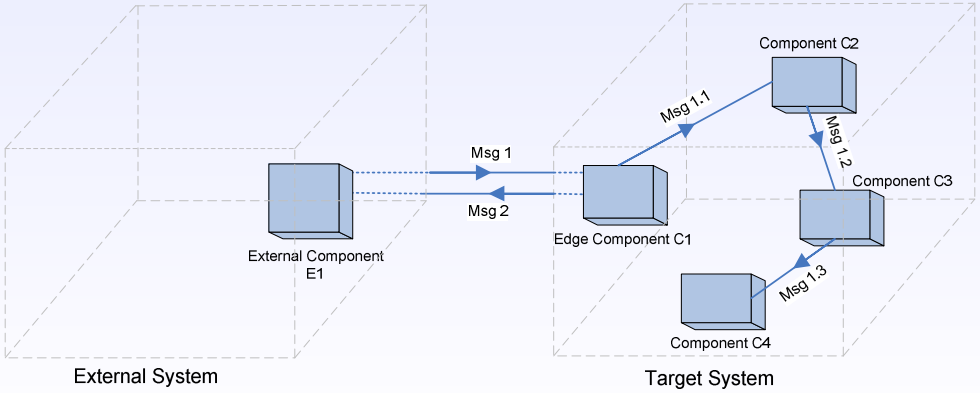
Components contribute to the overall system's behavior through the services they deliver.

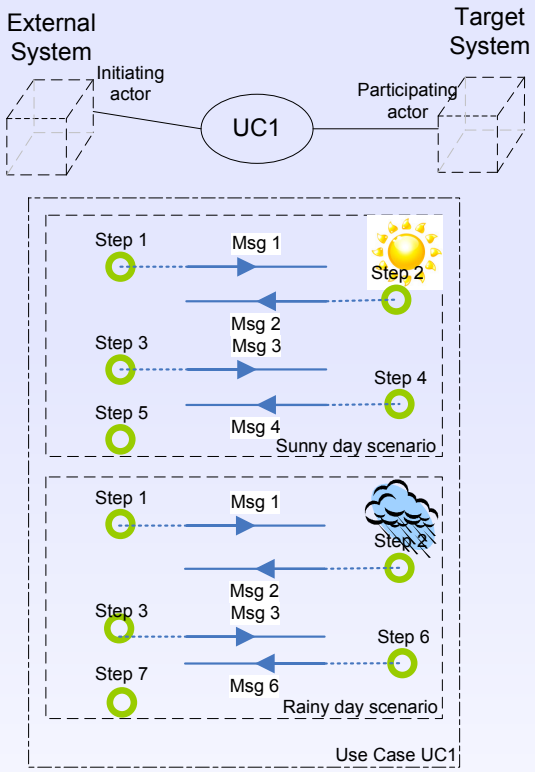
- A Service refers to the delivery of something of value according to a previously arranged agreement. The service delivered by the overall system depends on a sequence of matching requested and offered services delivered by the components contained in the system.



The exchange of messages between the external systems and the target system brings about the intended behavior.

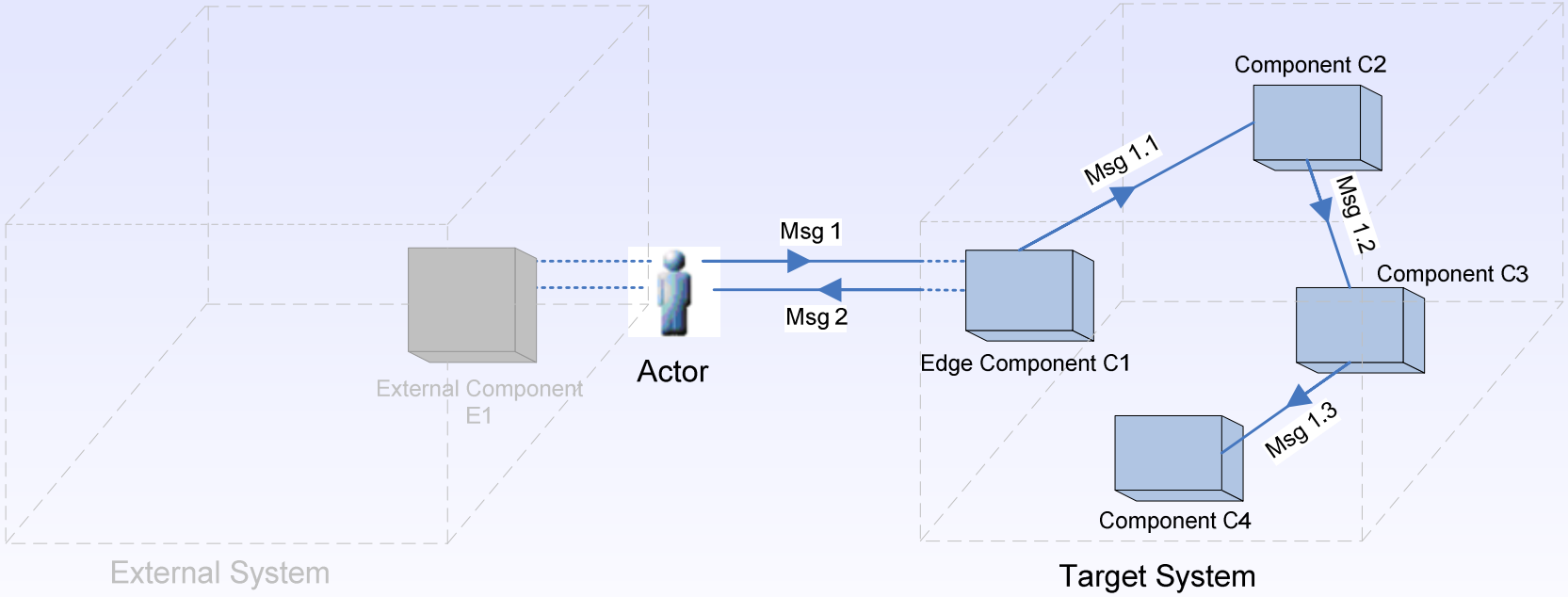
- The behavior of the initiating actor causes messages to be sent to other participating actors. So the target system responds to an external message (i.e. message 1 sent as result of step 1 taken by the external system) by performing step 2 resulting in sending message 2 to the external system. The behavior demonstrated 'across the system boundary' involves two components:
  - An edge component with which the external system interacts
  - An external component with which the target system interacts





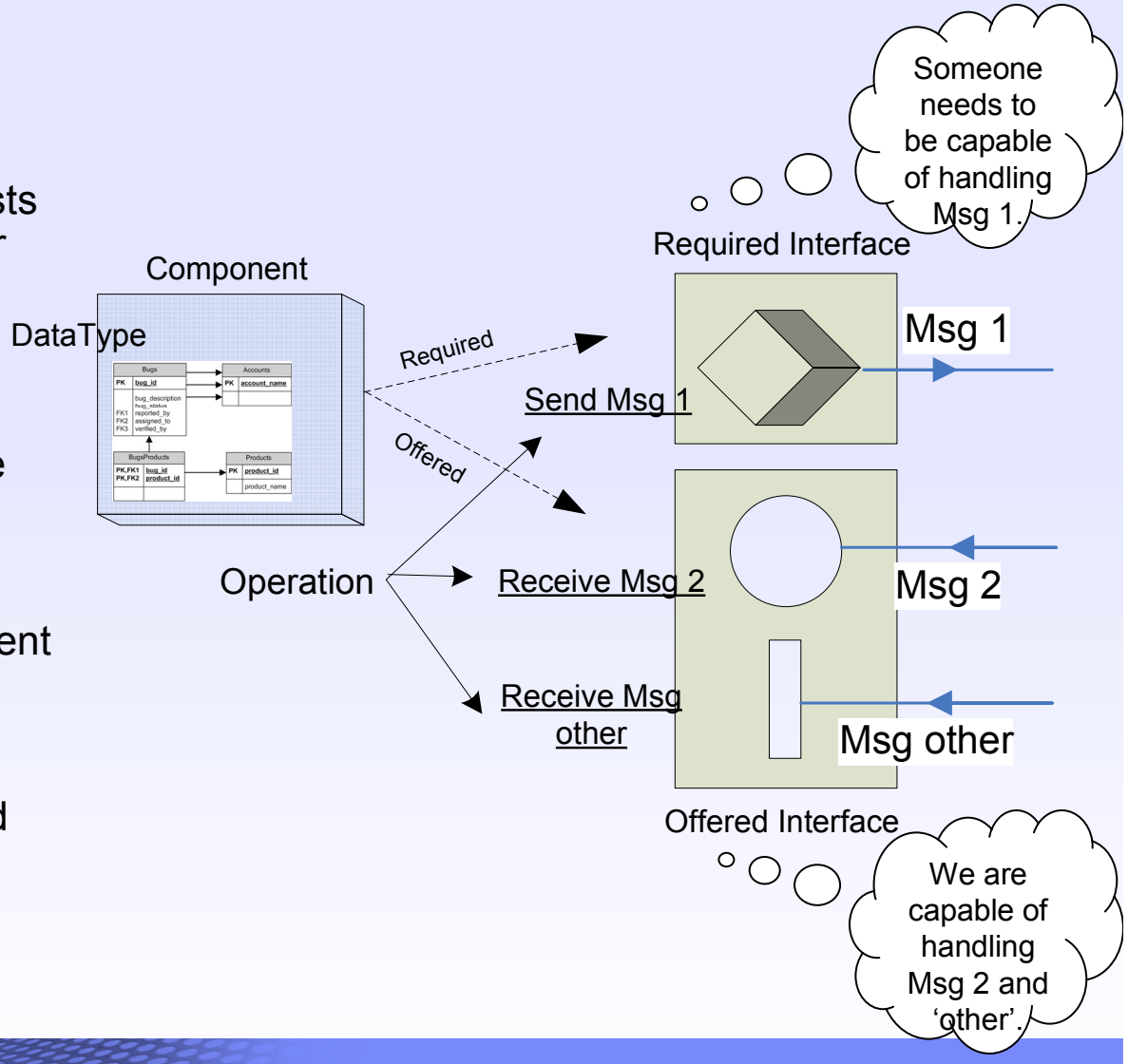
Specific patterns of exchanging messages with the target system are captured by introducing the notion of an 'actor'.

- An actor describes a logical external system interacting with the target system. In this context, the actor is understood as representing an external component.



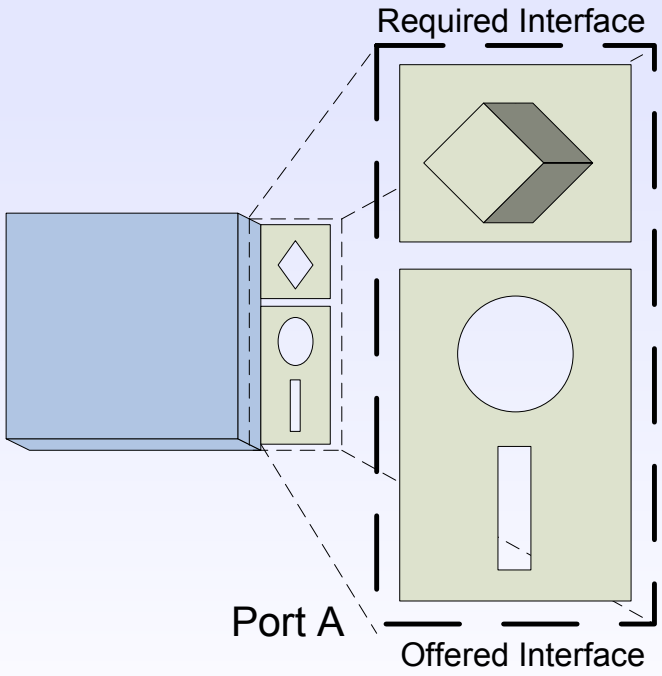
Components indicate what messages they either send or receive via the operations grouped in their interfaces.

- Components make their functionality available via their (offered) interfaces. The requests for functionality offered by other components are sent out via required interfaces.
- Each interface consists out of a number of operations which are triggered by the receipt of a specific message.
- Via these interfaces, a component makes the data it manages available to other components.
- Note: Data is the IT constrained version of IO Entity



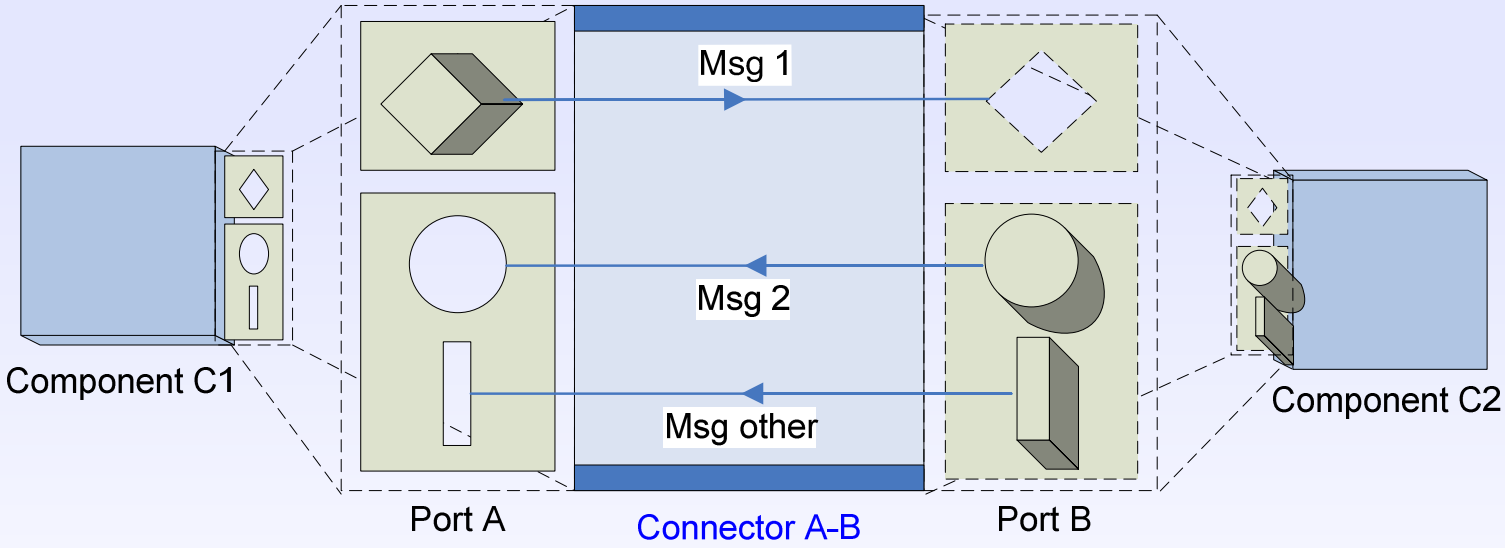
The offered and required interfaces used during specific interactions can be made explicit via their composition into a port.

- A port is the combination of one or more required and/or offered interfaces. It defines the messages, via its associated interfaces and their operations, which a component can exchange with another component during an interaction.



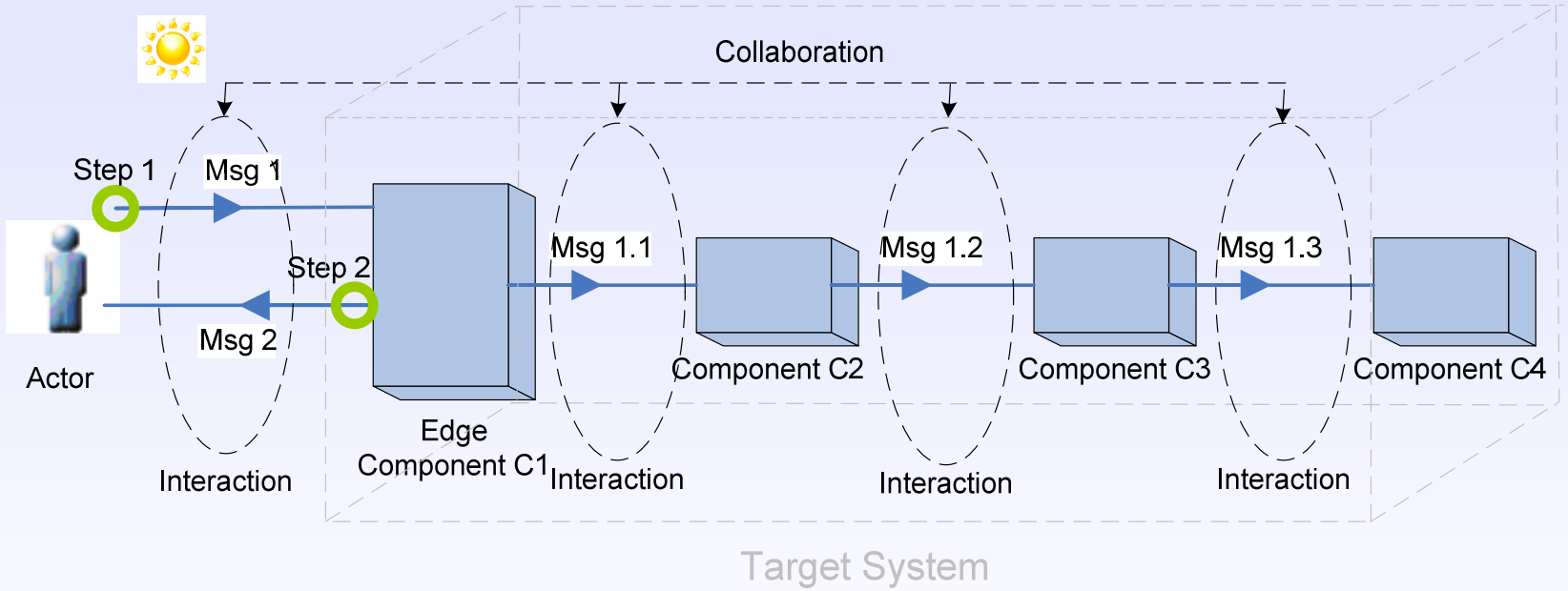
The exchange of messages between ports is enabled via a connector.

- A connector enables the exchange of these messages. Connectors can only be established between compatible ports, i.e. the required interface of Port A has to match the offered interface of Port B and vice versa.



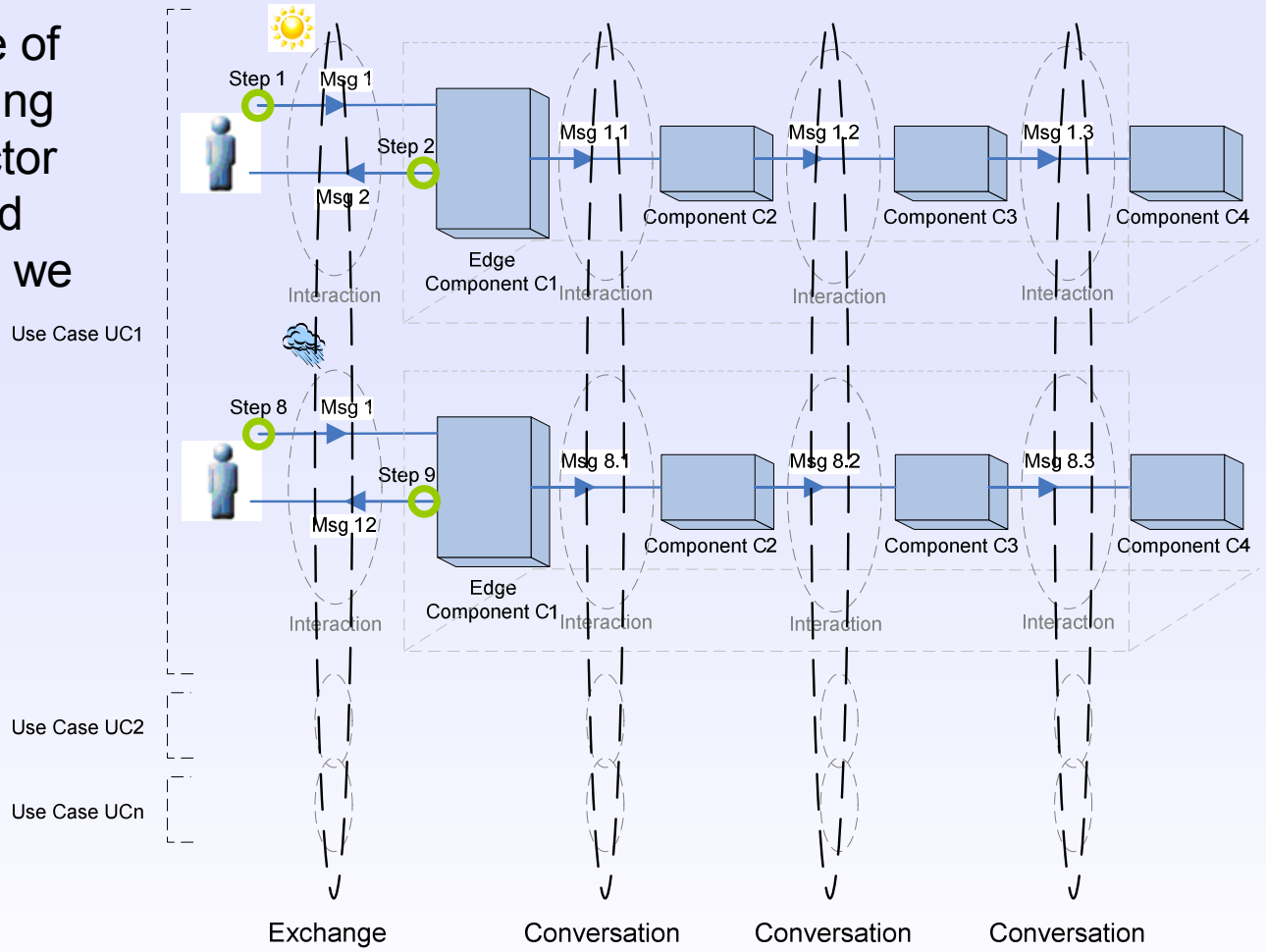
Collaborations capture the exchange of messages between the components involved in a particular scenario.

- Each step (e.g. step 1) results in sending a message (Msg 1) to the other system triggering a collaboration between components resulting in the desired reaction (Msg 2). Interactions capture all messages exchanged between 2 participants of the collaboration.



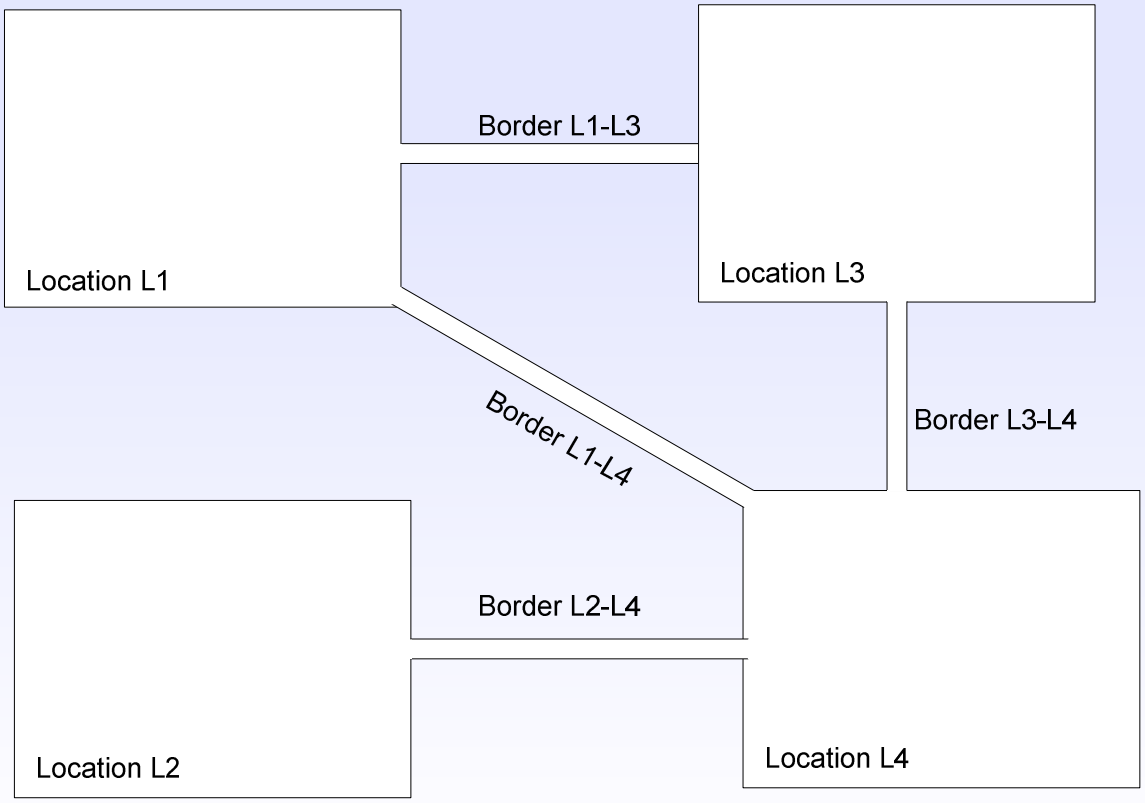
Conversations refer to all interactions taking place between actors and/or components across all collaborations.

- In the particular case of the conversation taking place between an actor (external system) and an edge component, we use the term ‘exchange’.



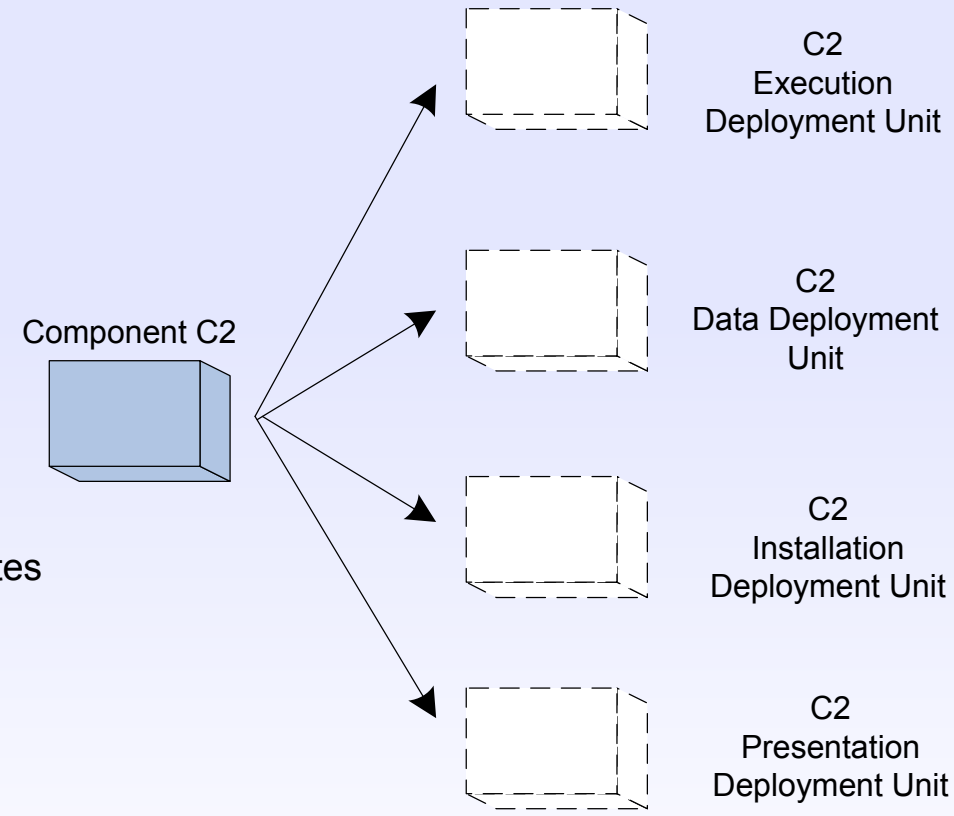
Locations and borders are used to lay out the geographical structure of the target system.

- Locations represent a geographical area or a position.
- A border refers to the connection between two locations.

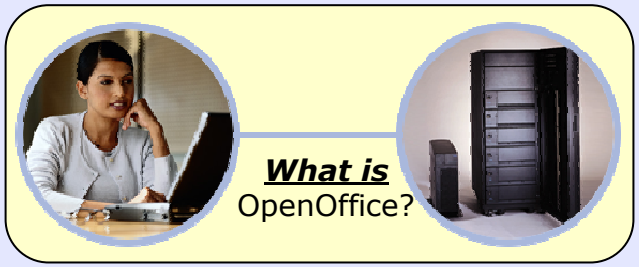


In order to define what a node should be capable of we need to take 4 different ways into account in which components impact these capabilities.

- We associate non-functional requirements with each of the following :
  - The Execution Deployment Unit, representing the component while it handles a particular message.
  - The Data Deployment Unit, representing the data managed by the component.
  - The Installation Deployment Unit, representing the installation of the component.
  - The Presentation Deployment Unit, representing how the component integrates in the environment.
- Note : deployment units are only applicable to IT Components.



Introducing the deployment units associated with OpenOffice.



Actors

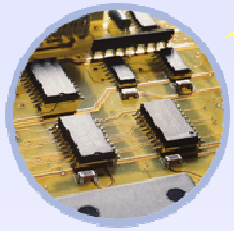
Components

Users

Systems

Presentation

Execution



Installation

Data

What are the characteristics of the component's **execution**? E.g. how quickly does the 'spell check' need to run.

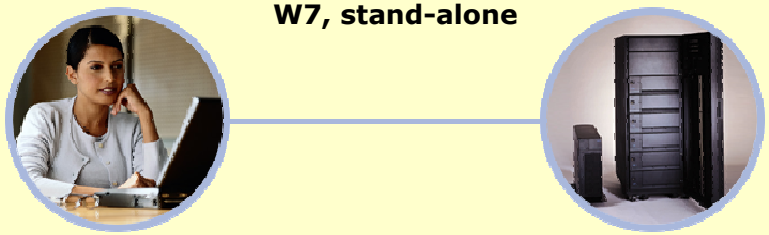
What are the characteristics of the component's **installed** files? E.g. how much space does the OpenOffice package need ?

What are the characteristics of any persistent **data** managed by the component? E.g. how much space do the standard templates take up ?

What are the characteristics of the component's **presentation** of its functionality to outside users or other IT systems? E.g. how much bandwidth is available for the user to connect to OpenOffice ?

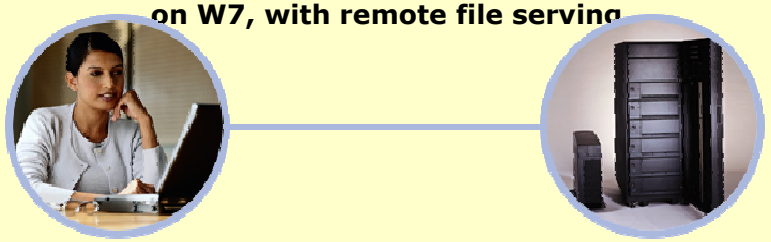
The following example illustrates the impact of the different allocation options of OpenOffice's deployment units.

Option 1: OpenOffice, running on W7, stand-alone



Presentation
Execution
Data
Installation

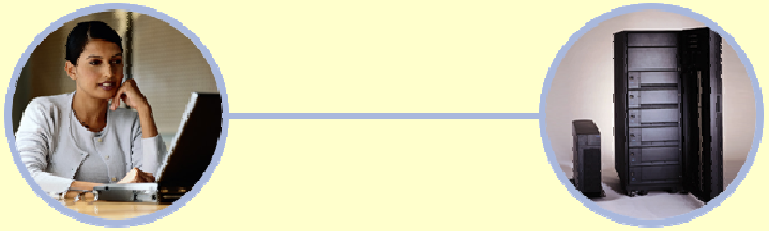
Option 2: OpenOffice, running on W7, with remote file serving



Presentation
Execution

Data
Installation

Option 3: OpenOffice, running on Citrix, with local data



Presentation
Data

Execution
Installation

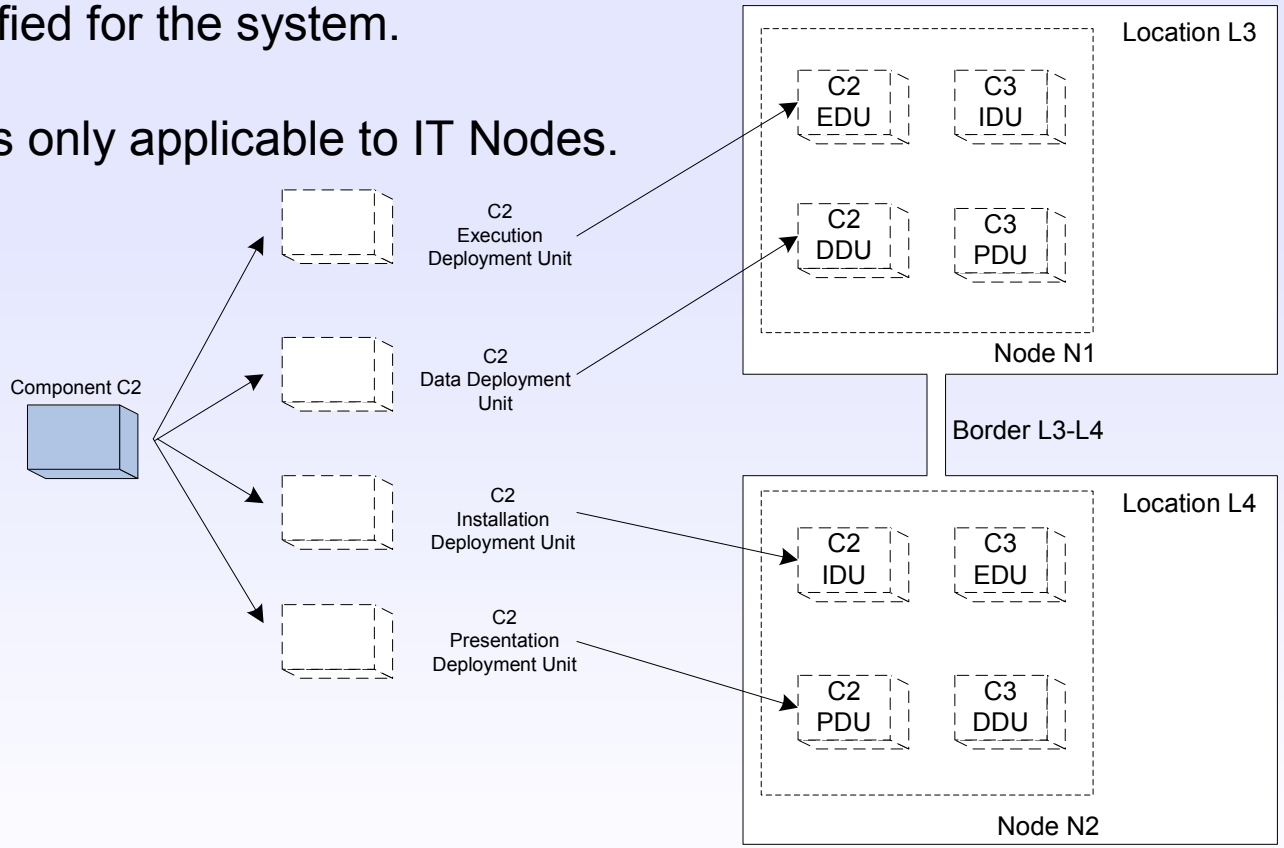
Option 4: OpenOffice, running on Citrix, with remote file serving

Presentation

Execution
Data
Installation

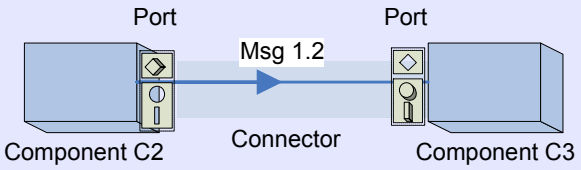
Nodes help us determine what capabilities need to be present at a given location in the target system.

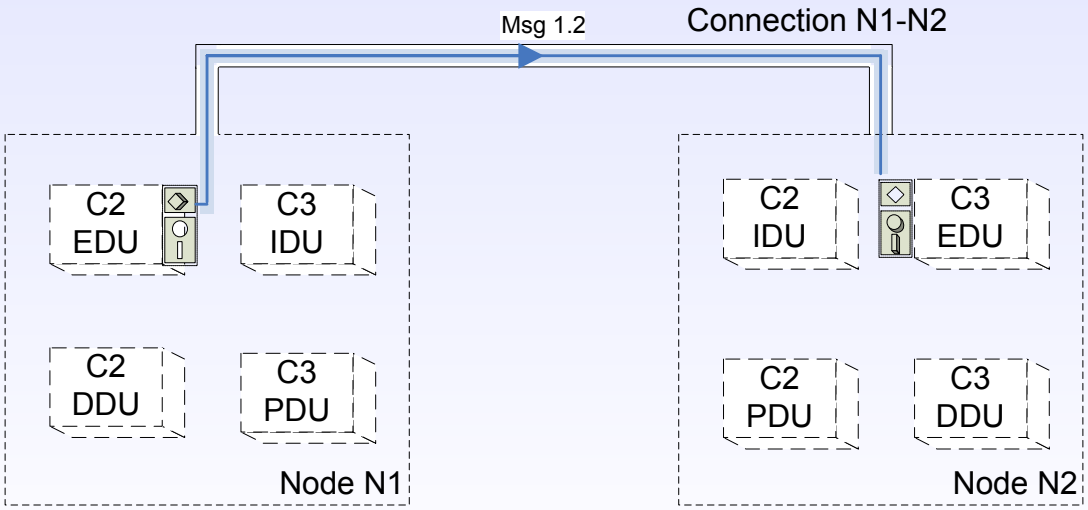
- Nodes are collections of deployment units which will end up at the same location in the system. They are positioned within one or more of the locations identified for the system.
- Note : this definition is only applicable to IT Nodes.



Connections are introduced to enable the interactions between components of which the deployment units have been grouped in different nodes.

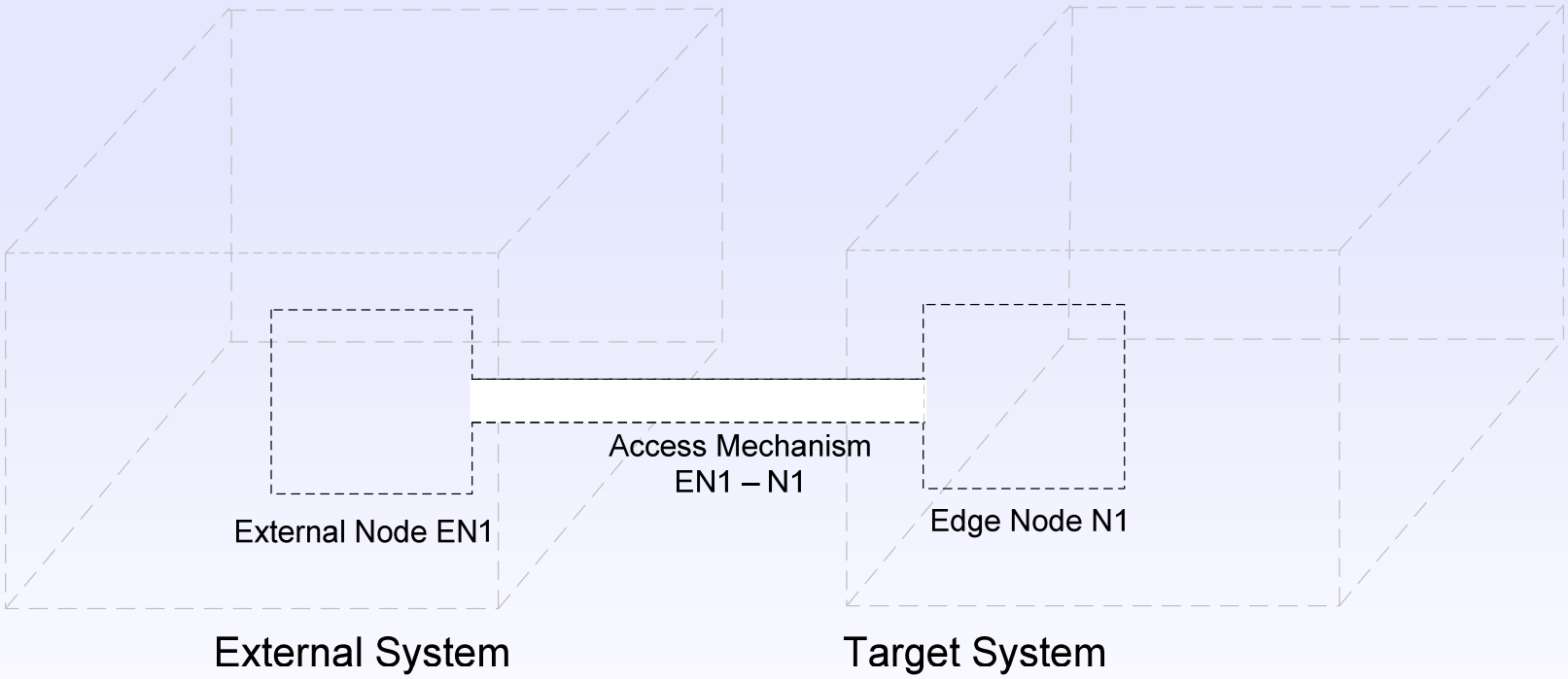
- Connections support the connectors which enable the exchange of messages between components. Or put alternatively, connections allow the exchange of messages between nodes.





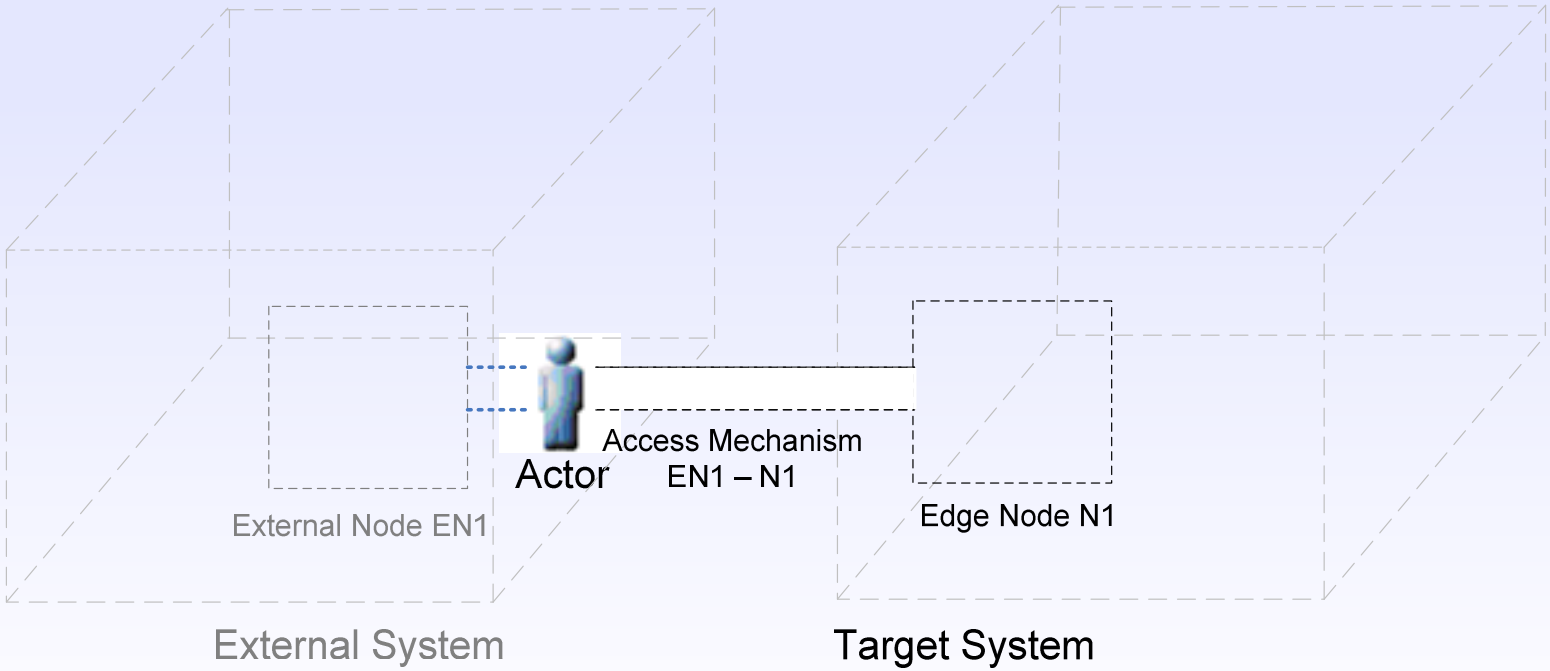
Connections crossing the system's boundary are referred to as 'access mechanisms'.

- Access mechanisms enable the exchanges across the system's boundary, i.e. between the external systems and the target system.

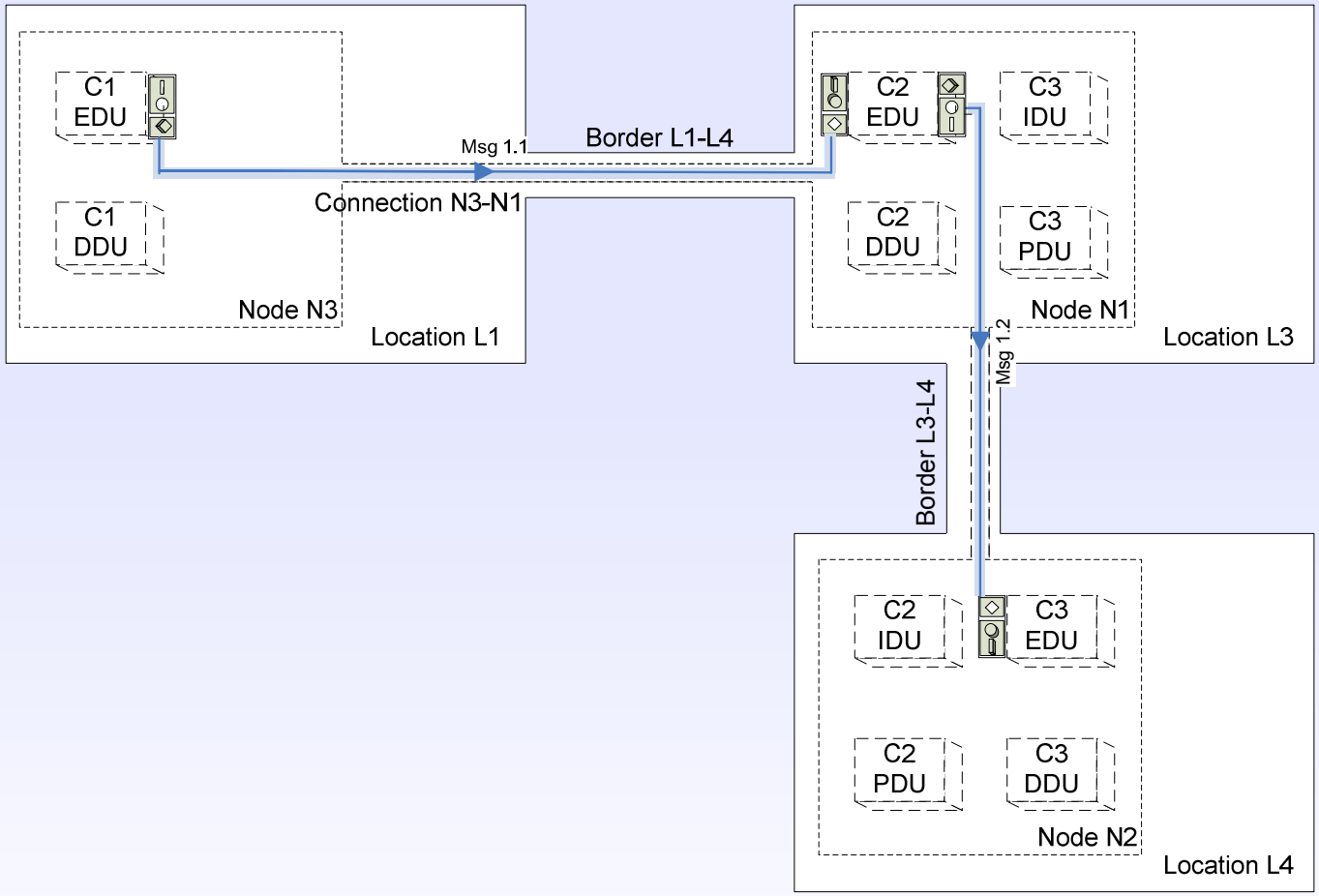


Specific integration patterns with the target system are captured by introducing the notion of an ‘actor’.

- An actor describes a logical external system which interacts with the target system. In this context, the actor is understood as representing an external node.



The connections between locations (the borders) support the required connectivity between nodes placed in different locations.



During the engineering of the target system, specific actions are taken to validate the obtained results.

- These actions are often part of a viability assessment and can lead to the identification of the following elements:
  - A finding: the result of an investigation
  - An issue: a generic term for a matter of concern on a project
  - A risk: a potential event or future situation that may adversely affect the project

Agenda

- Our challenge: Help addressing the systems' engineering challenge through establishing
  - A common language
  - A single model of the system
- Applicability of the System Description Standard
- Introduction to the SDS concepts
- **Frequently asked questions**
- SDS Reference Material
- References

Frequently Asked Questions

- Overview of the ADS/SDS Releases
- Evolution from ADS R2

Overview of the ADS/SDS Releases

- ADS R1 (1998):
  - Context: Creation of the SI method
  - Objective: Unify the language between WSDDM-OT & ISD (aka application and infrastructure integration)
- ADS R2 (2002):
  - Context: Initiated as a result of the Architecture Resources Academy Study
  - Objective:
    - Refine results from R1
    - Ensure ADS R2 can serve as the basis for a tooling implementation
- ADS/SDS R3 (2008):
  - Context: Support the development of the UMF method
  - Objective: Unify the language between GSM (ADS R2) and RUP – RUP SE (UML2 with extensions)
- SDS R3.1 (2012):
  - Context: Support for the SA4TeamSD tooling development effort
  - Objective: Minor modifications to support TeamSD

Evolution from ADS R2

- R3 introduces a number of significant changes, mainly related to a broadening of the context within which the concepts defined by the description standard can be used.
- SDS R3 introduces a specific framework (based on IEEE1471-2000) which defines the relationships between the views, viewpoints and models which can be used for describing systems ([3]). Whereas the actual models are defined implicitly through the artifacts of the UMF, the concepts from which these models are built are defined within the present document. Through this framework, the various contexts within which the SDS concepts can be used and the relationships between these contexts have been made explicit.
- An important change relates to the context within which the SDS concepts are intended to be used, since a large number of the SDS concepts are equally applicable within any number of different system contexts, as opposed to being only applicable to an IT systems context. Consequently, the Semantic Specification is organized with the general concepts first, and then only those which require redefinition in more specific contexts are further discussed (i.e. explicitly taking into account a more restrictive context of e.g. a IT system vs. a system). Concepts are only refined when this allows a richer set of relationships and meanings than would be the case in a more generalized context.

Evolution from ADS R2

- This broadening of the context is also the motivation for the name change from ‘Architecture Description Standard (ADS)’ to ‘System Description Standard (SDS)’.
- SDS R3 also includes a definition of the main concepts and relationships underpinning the requirements and validation basic viewpoints. This aligns SDS’s scope with the basic viewpoints defined in the UADF.
- Starting with this release (R3), all UML2 mapping considerations have been removed from the Semantic Specification, thereby enabling the creation of various UML2 or SysML based profiles. As a consequence, the Model Core package present in ADS R2 has been substantially simplified and now only includes those concepts which are of direct relevance to the SDS R3 concepts.

Agenda

- Our challenge: Help addressing the systems' engineering challenge through establishing
  - A common language
  - A single model of the system
- Applicability of the System Description Standard
- Introduction to the SDS concepts
- Frequently asked questions
- **SDS Reference Material**
- References

SDS Reference Material

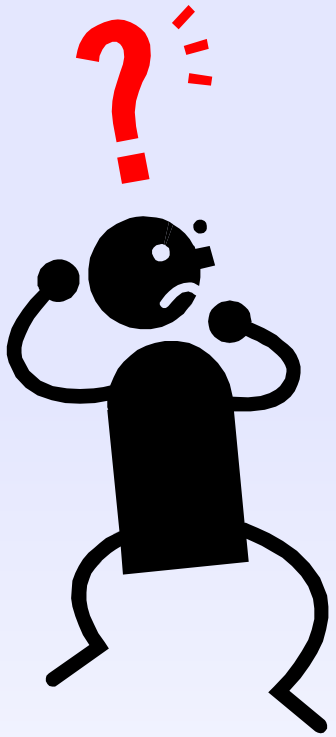
- All the following material is available in the SDS Community
- SDS R3.1 Glossary
  - The glossary gives an informal definition of the concepts required to describe systems.
- SDS R3.1 Semantic Specification
  - The SDS Semantic Specification aims to provide the clear and unambiguous definition of the concepts involved in modeling the requirements, architecture and validation of systems and extends, where necessary, the concepts defined in UML2.
- SDS related material
  - ADS Functional Aspects Notation
  - ADS Operational Model Notation
  - An introduction to the IBM Views and Viewpoints Framework for IT systems – Denise Cook, Peter Cripps, Philippe Spaas – January 8th 2008 (available from developerWorks)

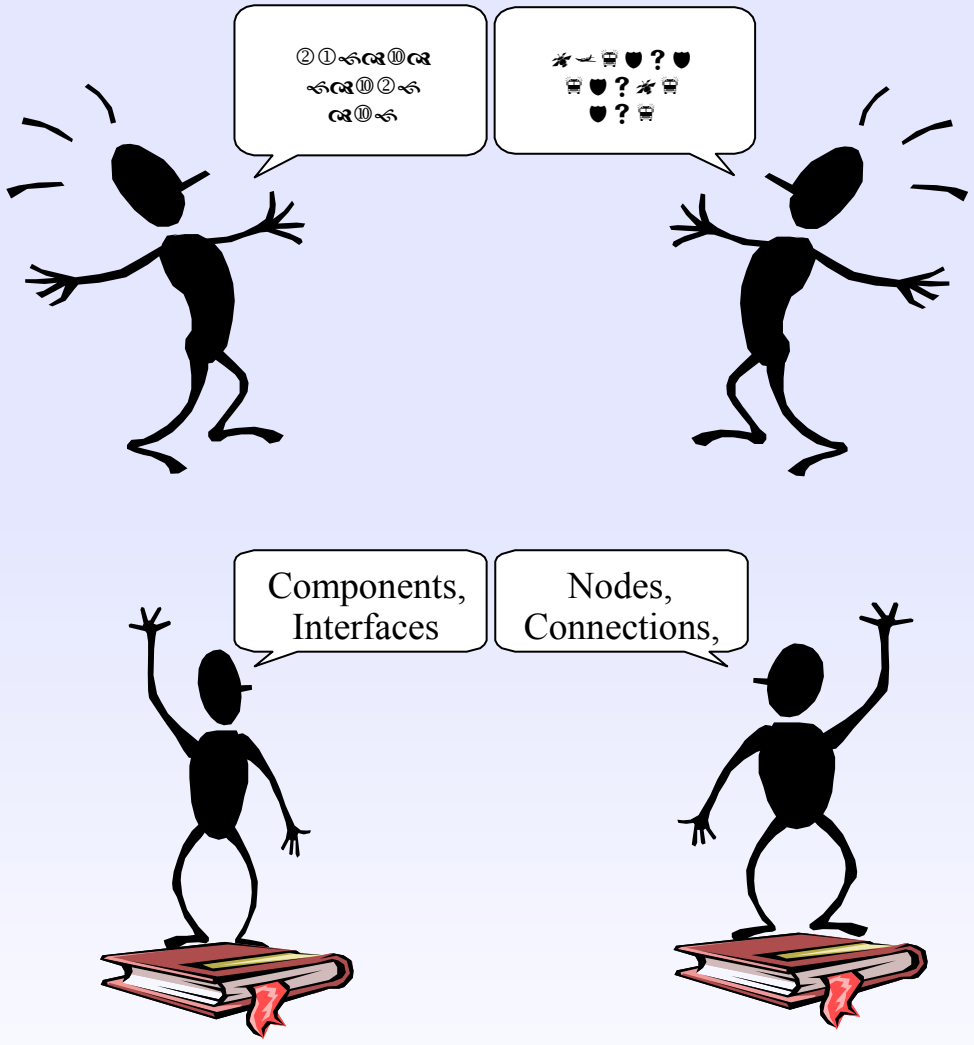
Agenda

- Our challenge: Help addressing the systems' engineering challenge through establishing
  - A common language
  - A single model of the system
- Applicability of the System Description Standard
- Introduction to the SDS concepts
- Frequently asked questions
- SDS Reference Material
- **References**

References

- An introduction to the IBM Views and Viewpoints Framework for IT systems – Denise Cook, Peter Cripps, Philippe Spaas – January 8th 2008 (available from developerWorks on http://www.ibm.com/developerworks/rational/library/08/0108_cooks-cripps-spaas/index.html)
- IEEE Architecture Working Group, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, IEEE Std 1471-2000, IEEE, 2000.
- OMG. *Unified Modelling Language: Superstructure Version 2.0*, OMG Formal Specification formal/05-07-04.
- System Context Modeling – Ian Charters, white paper published as part of the UMF Internal Content.





Questions can be addressed to
Philippe Spaas1\Belgium\IBM