Semantic and Pragmatic Interoperability: A Model for Understanding

Stanislav Pokraev[1], Manfred Reichert[2], Maarten W.A. Steen[1] and Roel J. Wieringa[2]

[1] Telematica Instituut, P.O. Box 589, 7500 AN Enschede, The Netherlands
{Stanislav.Pokraev, Maarten.W.A. Steen}@telin.nl, http://www.telin.nl/
[2] Center for Telematics and Information Technology,
University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
{M.U.Reichert, roelw}@ewi.utwente.nl, http://www.ewi.utwente.nl/

Abstract. In this paper we present a conceptual model for understanding of semantic and pragmatic interoperability. We use the model to identify and classify the possible semantic interoperability problems.

1 Introduction

Interoperability is the ability of *systems* to provide and use each other's *services effectively*. In this paper we present some basic concepts, which we use to explain what interoperability means. We propose a model for the understanding of semantic and pragmatic interoperability, and use this model to discuss semantic interoperability problems.

2 Basic concepts

Webster's dictionary defines a *system* as "*a regularly interacting or interdependent group of items, components or parts, forming a unified whole"*. This definition allows us to distinguish between *internal* and *external* system properties. The internal system properties refer to "*a regularly interacting or interdependent group of items, components or parts*" and *external* system properties refer to "*forming a unified whole*".

The external system properties can be classified as *functional* and *non-functional* (or *quality)* properties. Functional properties describe the functionality (i.e., the services*)* offered by the system to its *environment*, and quality properties describe the *value* of the system services for the stakeholders of the system. We define the *environment* of the system as the collection of all logical or physical entities that are able to *interact* with the system.

Services represent the capability of the system to interact with its environment and perform tasks that produce an identifiable *effect*. The interactions with the environment (i.e. the external system interactions) are realized by the exchange of *messages* between the system and its environment. The services of the system are realized be-

cause the messages sent out by the system have an effect, e.g. they provide information about something or change the state of the environment.

Each message has a *subject domain,* i.e. "*the part of the world that the message is about*" [1]. The subject domain of a message may include physical entities, events, people, social norms, or meaning conventions that are *identifiable* by the system. Furthermore, the subject domain of a system is the union of the subject domains of all messages that may enter and leave the system.

Messages consist of *data* that represent property values of entities or phenomena from the subject domain. The data in the messages have *meaning* only when it is interpreted in terms of the subject domain of the system.

To produce an effect, the system may need to perform a number of interactions with its environment. The *behavior* properties of the system describe ordering of the external system interaction in time.

System interactions are realized by the exchange of messages. The *communication* properties of the system describe senders and receivers of these messages as well as the communication channels between them.

Quality properties indicate how valuable the services of the system are to the system stakeholders. These properties characterize how secure, efficient, useful and reliable system is.

Internal properties describe the composition of the system. In fact, this composition repeats the external properties at lower levels of aggregation.

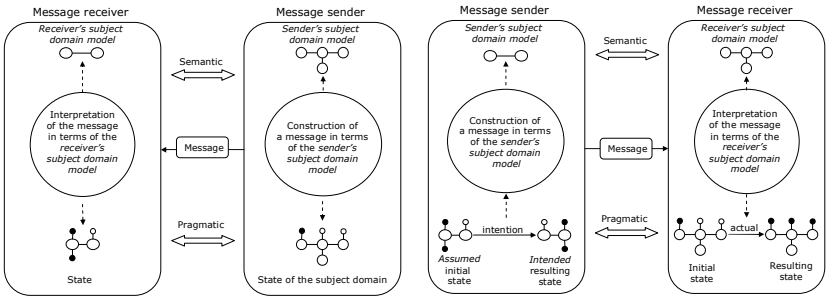
3 Semantic and Pragmatic Interoperability Model

Systems communicate with their environment by exchanging messages. For example, a clinical laboratory communicates with a healthcare professional by receiving a message that contains a request for blood analysis. The laboratory performs the tests and responds with a message that contains the results of the examination. A database system communicates with the user's application by receiving a message that contains an SQL statement, performs the requested action and responds with a message that contains a list of records that match criteria specified in the SQL statement. A cash dispenser system receives a message when the user inserts his card and responds with a message asking the user to enter his PIN.

Interoperability is about *effective* use of systems' *services*. The most important precondition to achieve interoperability is to ensure that the message sender and receiver share the same understanding of the *data* in the message and the same expectation of the *effect* of the message.

The data in the messages has a meaning only when it is interpreted in terms of the subject domain of the system. However, users do not always know the model of the subject domain of a system. Depending on their knowledge about the system they make *assumptions* about the subject domain of the system and use those *user subject domain models* to construct messages and communicate with the system. *Semantic interoperability problems* arise when *user subject domain models* differ from the *system subject domain model*.

When a system receives a messages it changes its state, sends a message back to the environment, or both [1]. In most cases, messages sent to the system change or request the system state, and messages sent from the system change or request the state of the environment. That is, messages are always sent with some *intention.* The *pragmatic interoperability problems* arise when the *intended effect* of a message differs from the *actual effect* of the message.



a) System sends a message to its environment to report about the state of the subject domain

b) an entity in the environment sends a message to a system to change its state.

Figure 1. Semantic and Pragmatic Interoperability Model

Figure 1a presents an interoperability model when a system sends a message to its environment to report about the state of the subject domain. A message is constructed in terms of the *sender's* subject domain model. A user of the system receives the message and interprets it in the terms of its *receiver's* subject domain model. If there is a difference between the *sender's* and *receiver's* subject domain models the user may misinterpret the message, which may result in a mismatch between the *intended* and *actual effect* of the message.

Figure 1b presents an interoperability model when a user sends a message to a system to change its state. A message is constructed in terms of the *sender's* subject domain model assuming some *initial state* of the system. It expresses an intention of the user to achieve change this initial state to some desired *resulting state*. The system receives the message and interprets it in terms of its *receiver's subject domain* model and executes some action that changes the *actual initial state* of the system. If there is a difference between the *sender's* and *receiver's* subject domain model the system may misinterpret the message which may result in a wrong actual resulting state. If there is a difference between the *assumed initial state* and the *actual initial state* of the system, the system may perform a wrong state transition. In such a case the intended effect of the message differs from the actual effect of the message.

4 Classification of Semantic Interoperability Problems

Semantic and pragmatic interoperability problems arise from the fact that message sender and receiver do not use the same model of the subject domain to construct their

messages. In this section we make a brief inventory of the possible disparities between the sender's and receiver's subject domain models.

A subject domain of a system consists of real-world *entities*. We classify such real-world entities into *abstract classes* (also called *entity types*). That is, we derive *concepts* that are sets of entities with similar *properties*. A concept is part of our "internal reality" and only exists in our minds. However, in order to communicate, we need a *designator* (i.e. a symbol) that designates the concept and thus denotes the entity from the subject domain. Such a symbol is the representation of the concept by a language or other means.

Systems exchange messages that represent property *values* of entities from the subject domain. Semantic interoperability problems arise when message sender and receiver use different representation of the same *value (value-level conflicts)*, or use different *entity types (type-level conflicts)* to interpret the same value.

4.1. Value-level conflicts

Data-value conflicts arise from different interpretations of the same data in different context. *Naming conflicts* arise when identical representations (*homonyms*) are used to represent different property values, or different representations (*synonyms*) are used to represent the same property value of the same entity in the sender's and receiver's subject domains. *Unit, scale and precision conflicts* arise when the property value of the same entity is represented in different unit systems, different scales, or different precision in the sender's and receiver's subject domain.

4.2. Type-level conflicts

Naming conflicts at type level arise when different terms that have the same representation (*homonyms*) designate the same entity type or relationship in the sender's and receiver's subject domain model, or when different terms that have different representation (*synonyms*) designate the same entity type or relationship in the sender's and receiver's subject domain model. *Generalization conflicts* arise when the meaning of an entity type or a relationship in the sender's subject domain model is more general than the meaning of the corresponding entity type or relationship in the receiver's subject domain model and vice versa. *Aggregation conflicts* arise when an entity type or a relationship in the sender's subject domain model aggregates two or more entity types or relationships in the receiver's subject domain model and vice versa. *Isomorphism conflicts* arise when the same entity type or relationship is defined by dissimilar properties in the sender's and receiver's subject domain models. *Overlapping conflicts* arise when an entity type or a relationship in the sender's subject domain model partially overlaps an entity type or relationship in the receiver's subject domain model and vice versa. *Identification conflicts* arise when the same entity or relationship is identified by different set of properties in the sender's and receiver's subject domain model. *Entity-Relationship conflicts* arise when an entity in the sender's subject domain model is modeled as a relationship in the receiver's subject domain model and vice versa. *Default value conflicts* arise when the default property value of an entity type in the sender's subject domain model differs from the default property value of the corresponding entity type from the receiver's subject domain model and vice versa. *Constraints conflicts* arise when the constraints on the property value in the

sender's subject domain model differs from the constraints on the corresponding property value from the receiver's subject domain model and vice versa. *Cardinality conflicts* arise when the allowed number of instances of an entity type in the sender's subject domain model differs from the allowed number of instances of the corresponding entity type in the receiver's subject domain model and vice versa.

5 Conclusions

Semantic interoperability problems arise when service users make wrong assumptions about the subject domain model of the system that provides the service. Pragmatic interoperability problems arise when the intended effect of the used service differs from the actual effect of the service.

Systems exchange messages that represent property values of entities from their subject domains. Semantic interoperability problems arise when message sender and receiver use different *representation* of the same value, or use different *entity types* to interpret the same value. In this paper, we have presented a categorization of different semantic interoperability problems at value and type level.

We would be very interested to get in touch with other members of the Interop NoE working on similar or closely related research questions. In our future work, we plan to further analyze the identified semantic interoperability problems and propose a set of requirements for the solution technologies.

This paper is the result of a collaboration between the Telematica Instituut and the University of Twente, the Netherlands, which is partially supported by the Commission of the European Communities under the sixth framework programme (INTEROP Network of Excellence, Contract N° 508011, http://www.interop-noe.org/).

References

1. R.J. Wieringa. Design Methods for Reactive Systems: Yourdon, Statemate, and the UML. Morgan Kaufmann, 2003. http://www.mkp.com/dmrs
2. R.J. Wieringa, H.M. Blanken, M.M. Fokkinga, and P.W.P.J. Grefen. Aligning application architecture to the business context. In Conference on Advanced Information System Engineering (CAiSE 03), 2003. Available at: http://is.cs.utwente.nl/GRAAL/Wieringa_etal_caise03.pdf