

System model documentation

An approach for modeling collaboration in industrial value networks

Authors: Alexander Korczok, Wendelin Gross

31.03.2022

1 Introduction and objectives

The system model serves as a central planning model in collaboration projects. The abstract representation of processes, data and responsibilities creates transparency, a common understanding about and commitment to the collaboration use case among all partners involved (see Figure 1). In addition, the system model enables people from outside the field to understand important issues and discuss them during the specification and implementation of the use case. For this reason, it is advisable for a consortium willing to collaborate to convert the collaboration vision into a first draft of the system model at the beginning of the collaboration project setup.

By illustrating the details of related processes in a simplified form, the system model supports the design and documentation of the ideas. The system model shows the nature and interrelationships of the processes by describing all collaboration-relevant components, such as companies, machines or IT infrastructure elements, in sufficient detail. Furthermore, it describes the mutual interactions of these components, and specifies the data sharing conditions between the partners and components.

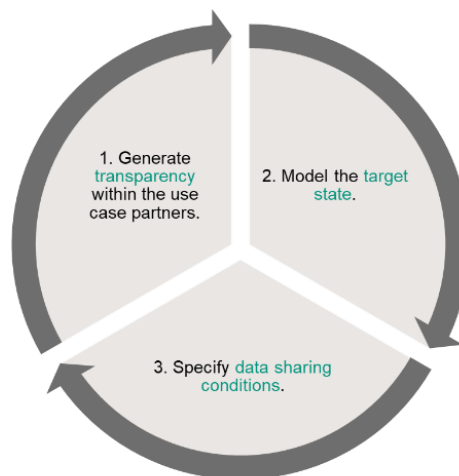


Figure 1: Basic functionalities of the system model

The system model supports the later phases of collaboration projects with further submodels specialized for the relevant issues. In this way, the system model is continuously developed and, in an iterative procedure, captures more and more sub-aspects that, taken together, define the system as a whole. The focus of the system model is use case-related data. This means that all submodels united illustrate which data is shared at which points in time, how it is used, for whom it can be viewed, and which decisions can be derived from it.

The system model has been successfully applied in five industrial use cases within the research projects ReKoNeT (funded by the German Federal Ministry of Education and Research) and SmartCoNeT (funded in the SMART EUREKA CLUSTER). Further information on the project can be found in the ReKoNeT/SmartCoNeT User Guide [1].

2 Meta model

The meta model describes the structure of the system model, which is created by combining four diagram types of the UML notation (Unified Modeling Language [2]), a general-purpose, developmental, modeling language. There are two diagrams describing the structure and two diagrams describing the behavior of components within the use case. For the modeling order it is recommended to work with the following procedure (Figure 2):

1. First the structure of the use case is modeled in the **component diagram**. It includes the hierarchy components and the collaboration-relevant data.
2. In the second step the **communication diagram** of the behavior is modeled. This comprises mapping the actors including the sender and receiver relationship and the sequence of their interaction.
3. The next step is to show all process-relevant components, data and activities. For this purpose, the **activity diagram** is used.
4. The **class diagram** is the last and most specific. It describes the data and their relationship in detail.

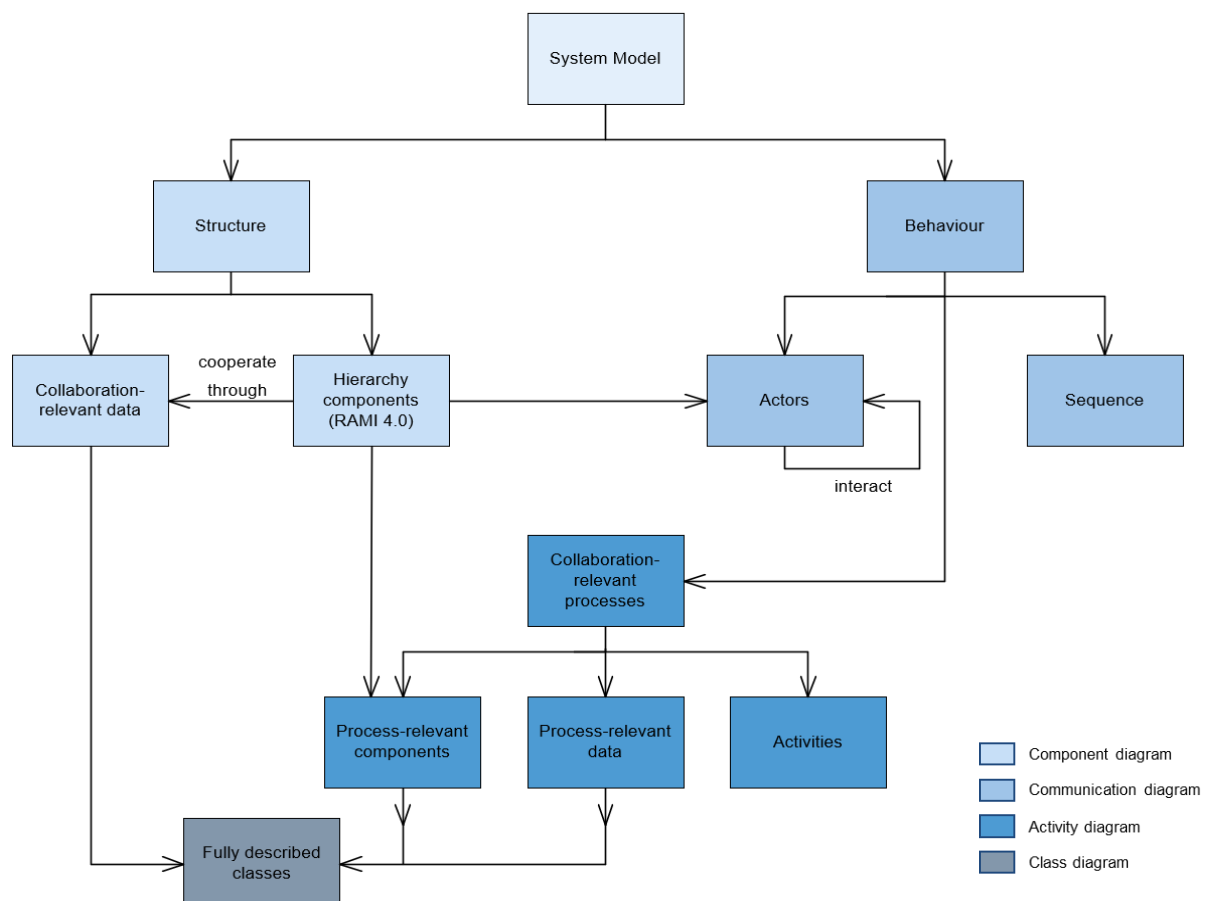


Figure 2: Meta model of the collaboration model

3 Component Diagram

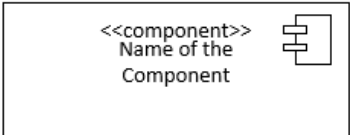
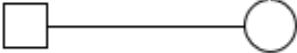
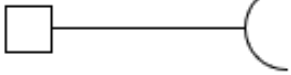
“A Component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. A Component is a self-contained unit that encapsulates the state and behavior of several Classifiers. A Component specifies a formal contract of the services that it provides to its clients and those that it requires from other Components or services in the system in terms of its provided and required Interfaces.” [3, p. 209]

The component diagram forms the highest level of abstraction in the system model and describes the rough structure of the use case. It shows the sender and receiver relationships through components that represent the abstract form of an actor. It is possible to map objects as an actor. For example, these can be companies, departments, machines or cloud systems. The diagram can sketch them on a high level and illustrate the following two aspects:

- The structure of the IT infrastructure - how do the systems communicate?
(Is there a direct exchange between the parties or is there a central server which transfers the data?)
- Dependency relationships - Who needs what type of data from a partner?
(Does redundant data exchange take place?)

The component diagram uses the notation elements shown in Table 1:

Table 1: Notation elements of the component diagram

	<p>Component</p> <p>A component is described as an exchangeable module. This feature makes it possible to map a variable number of actors. The instance of a component (the actor) must use the same data as input and output. The <i>inner behavior</i> of a component is not considered. [3, p. 209]</p>
	<p>Interface provided</p> <p>The interface provided shows the presence of an interface and the data that a component provides through the interface.</p>
	<p>Interface required</p> <p>The interface required shows the presence of an interface and the data that a component needs for its tasks as input.</p>

The example of a customer order in a collaborative value network in Figure 3 shows the structure of a simplified use case with five interconnected components. The focus of the sketch is the relationship between transmitter and receiver, including the transferred data.

The figure shows a mainly cloud-based data exchange, with only two exceptions. The cloud system plays a central role in this collaboration. Therefore, it is of high relevance for the planning of the infrastructure.

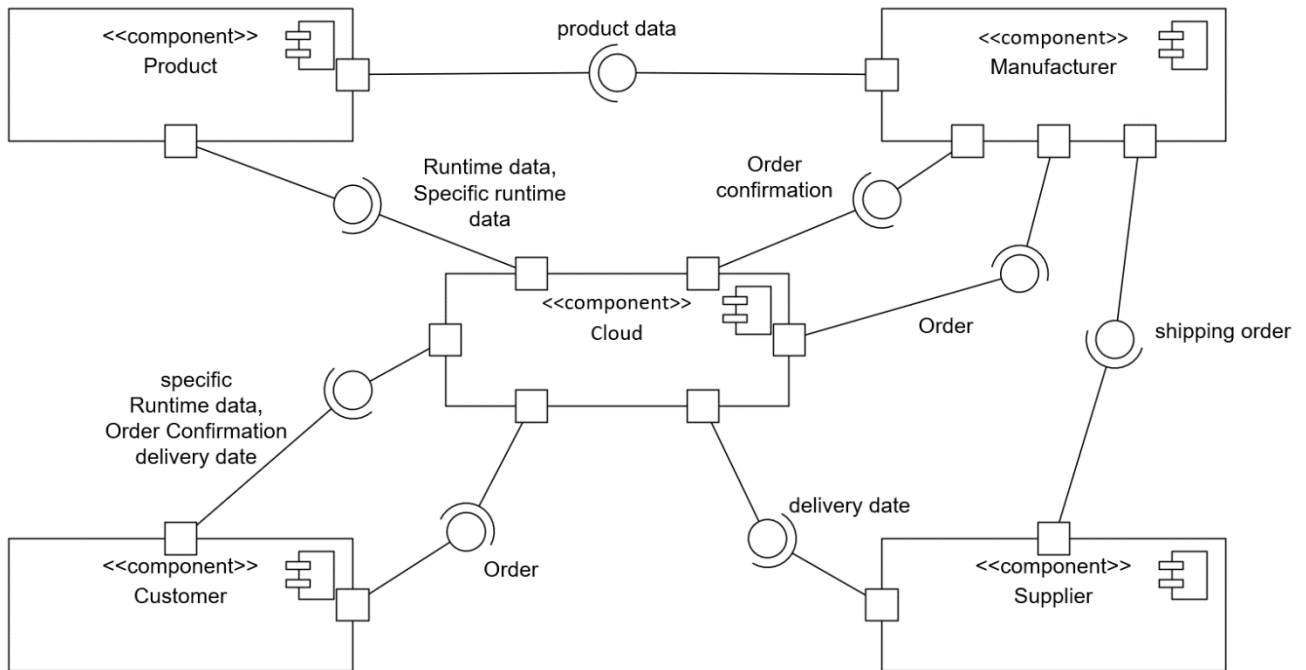


Figure 3: Example of a component diagram

4 Communication Diagram

Communication Diagrams focus on the interaction between instances of components, which are based on the component diagram. The instances are linked by messages. The sequencing of messages is given through a sequence numbering scheme.

The communication diagram specifies the component diagram. It further elaborates the previously sketched structure and explains the communication behavior of the use case, which contains the following points:

- Instances
- Sequential communication
- Sender and receiver

Figure 4 shows the implementation of two communication diagrams, which are based on the template of the component diagram from Figure 3. Due to the different use cases (left: submit order, right: sharing runtime data), the behavior is described separately. For the first time, the instances (*producer xy*, *customer xy*, etc. in Figure 3) resulting from the components outlined above are now described. Components can be assigned to multiple instances. For example, the component *cloud* could be followed by a computer network consisting of many instances.

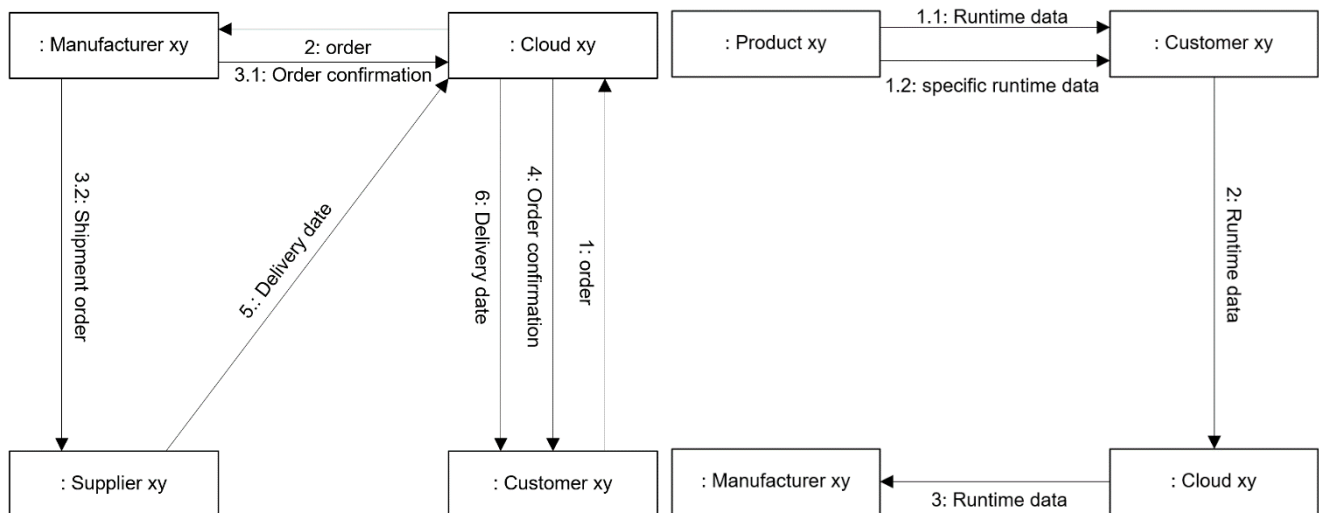


Figure 4: Example of two communication diagrams

5 Activity Diagram

“An activity is a behavior specified as sequencing of subordinate units, using a control and data flow model.” [3, p. 373] Units can be actions or objects, amongst others (see Table 2)

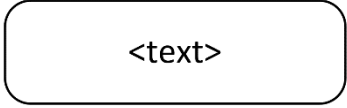
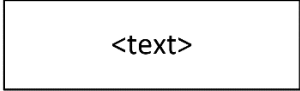
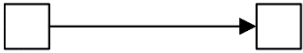
Together with the class diagram, the activity diagram forms the most specific representation of the use case within the system model. The focus is on the description of the behavior, which includes collaboration-relevant processes, components as well as data and activities. The diagram uses the previously sketched contents of the component and communication diagram.

The diagram answers the following questions:

- What activity is taking place, what data do they require?
- At which point do the partners exchange their data?
- Which decisions are made and when?
- Which activities take place in parallel?
- What causes a flow to terminate?

The activity diagram uses multiple notation elements, which are listed under [3, p. 374]. The following table shows only the activity and object notations:

Table 2: Notation elements of the activity diagram

	Action Within the represented process the action describes an activity. Actions can be sequential or parallel and bound to objects.
	Object node An object node represents the classes that were previously described in the component and communication diagram.
	Object pin An object pin is a reduced representation of an object node and is associated with an action, which depends on the object.

The example in Figure 5 shows a part of the activity diagram for the use case of the customer order. The actors are represented by swim lanes and are the same as in the communication diagram in Figure 4. The swim lanes allow to identify data exchange between the actors. The object pins show which objects are necessary for which actions. For example, the action *Calculate delivery date* can only be executed if the corresponding object *Shipment order* has been received. It is possible to model parallel processes, data-based decisions, and feedback in the activity diagram.

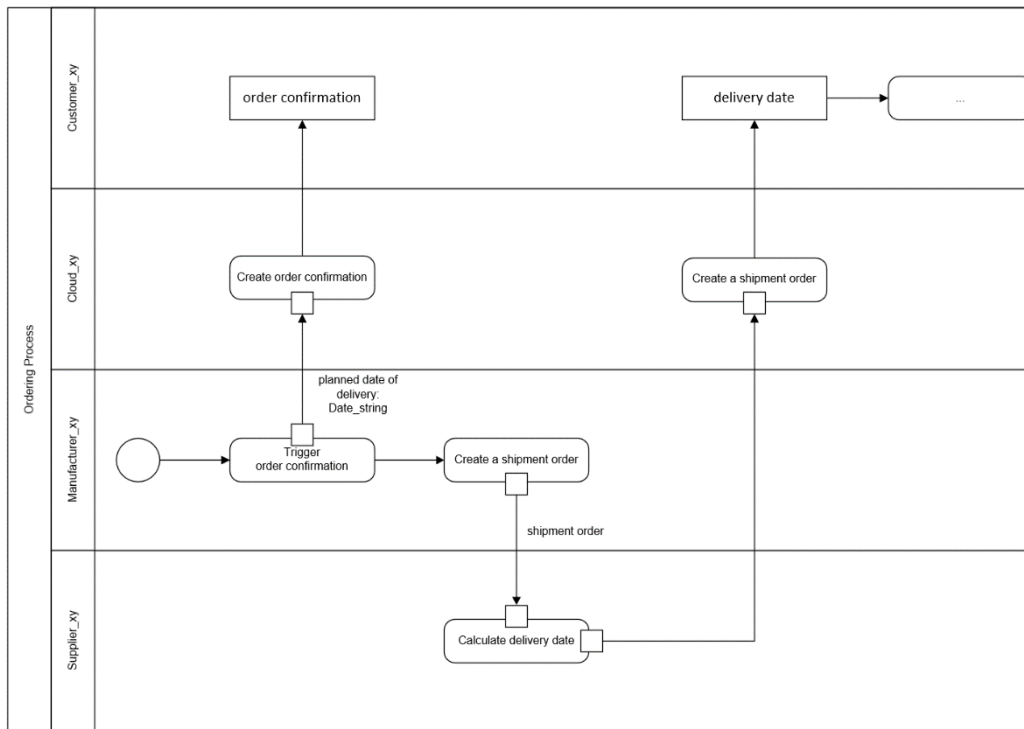


Figure 5: Example of an activity diagram

6 Class Diagram

“The purpose of a Class is to specify a classification of objects and to specify the features that characterize the structure and behavior of those objects.” [3, p.194]

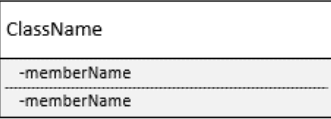
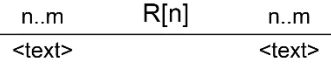

The class diagram contains the most concrete structural information of all diagrams. As a result, it usually has the largest scope of all diagrams used in the system model. It contains all the classes, their characteristics, and relationships. Basically, the class diagram can be designed individually for different questions, for example to place data protection issues at the center of consideration.

The previously sketched diagrams refer to the information of the class diagram, which also shows the following information:

- Are there any unique characteristics (for example, ID, URI, etc.)?
- In which data types are the characteristics available?
- How many times can a class exist?
- What is the relationship between the classes?

The class diagram offers a large selection of notation elements, which are described in [3]. The following notations were used for the use case:

Table 2: Notation elements of the class diagram

	Classe A class represents a technically relevant concept, a set of people, things, or ideas that are mapped in the system.
	Association The association describes the relationship between the classes. In addition to a consecutive numbering (R [n]), it contains the cardinality of relationships (n..m) as well as the description of the relationship (<text>).
	Individually If a characteristic of a class is described as {l}, then this characteristic uniquely exists within the class.

The example of Figure 6 shows the implementation of a class diagram. The classes represent actors as well as data [4]. The relevant data in the use case are assigned to the respective classes in the form of their characteristics.

For example, the class *Cloud xy* contains a variety of individual data, some of which are marked as individual ({I}) and others marked as class-external data ({R [n]}). Through the associations multiple classes are connected to the class *Cloud xy*.

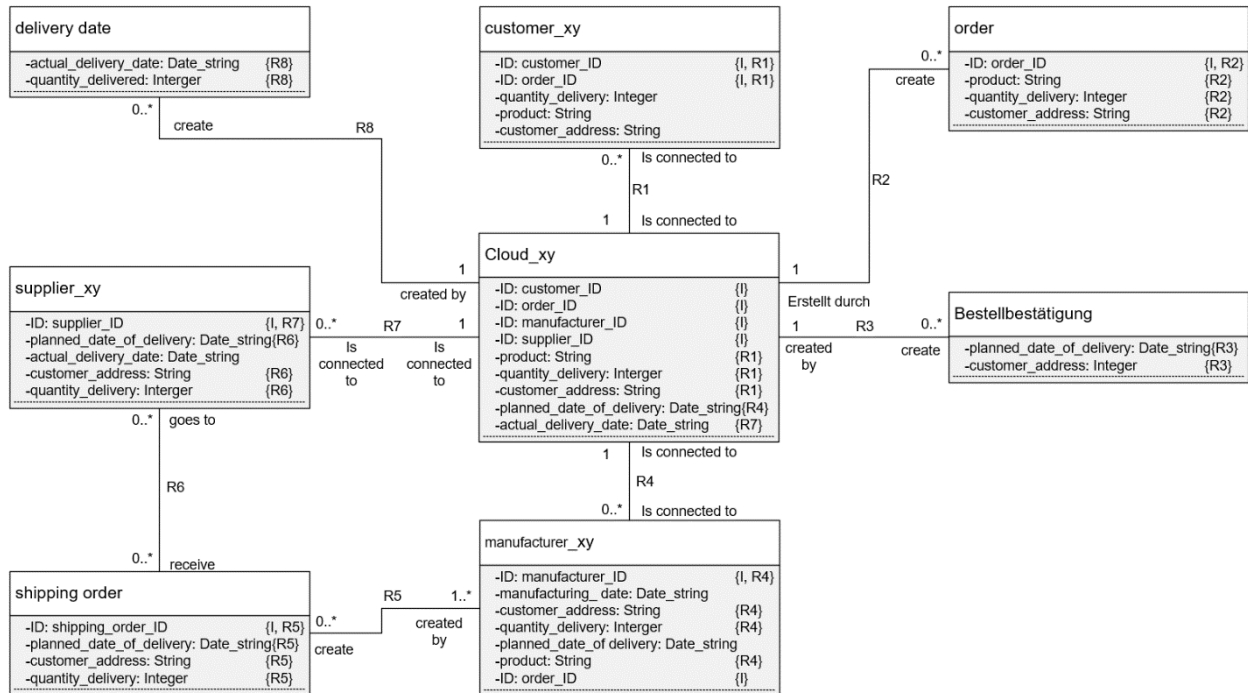


Figure 6: Example of a class diagram

References

- [1] R. Silbernagel, F. Stamer, B. Sautter, in: G. Lanza (ed), „ Successful Collaboration in Global Production Networks - fair, secured, connected.” DOI: 10.5445/IR/1000144272, 2022.
- [2] H.-E. Eriksson and M. Penker, “Business modeling with UML,” *New York*, pp. 1–12, 2000.
- [3] “Unified Modeling Language, v2.5.1,” *Unified Modeling Language*. [Online]. Available: <http://www.omg.org/spec/UML/2.5.1/PDF>. [Accessed: 16-Oct-2019].
- [4] P. Grässle, H. Baumann, and P. Baumann, *UML projektorientiert: Geschäftsprozessmodellierung; IT-System-Spezifikation und Systemintegration mit der UML*, 1. Aufl., 1. Nachdr. Bonn: Galileo Press, 2001.