Complex Systems Informatics and Modeling Quarterly (CSIMQ)
eISSN: 2255-9922
Published online by RTU Press, https://csimq-journals.rtu.lv
Article 89, Issue 15, July 2018, Pages 72–89
https://doi.org/10.7250/csimq.2018-15.04

Automated Creation and Provisioning of Decision Information Packages for the Smart Factory

Eva Hoos[1,2⋆], Pascal Hirmer[1] and Bernhard Mitschang[1,2]

[1]Institute of Parallel and Distributed Systems, University of Stuttgart, 70569 Stuttgart, Germany
[2]Graduate School of Excellence Advanced Manufacturing Engineering, University of Stuttgart, 70569 Stuttgart, Germany

Eva.Hoos@ipvs.uni-stuttgart.de, Pascal.Hirmer@ipvs.uni-stuttgart.de, Bernhard.Mitschang@ipvs.uni-stuttgart.de

Abstract. In recent years, Industry 4.0 emerges as a new trend, enabling the integration of data-intensive cyber physical systems, Internet of Things, and mobile applications, into production environments. Even though Industry 4.0 concentrates on automated engineering and manufacturing processing, the human actor is still important for decision making in the product lifecycle process. To support correct and efficient decision making, human actors have to be provided with relevant data depending on the current context. This data needs to be retrieved from distributed sources like bill of material systems, product data management and manufacturing execution systems, holding product model and factory model. In this article, we address this issue by introducing the concept of decision information packages, which enable to compose relevant engineering data for a specific context from distributed data sources. To determine relevant data, we specify a context-aware engineering data model and corresponding operators. To realize our approach, we provide an architecture and a prototypical implementation based on requirements of a real case scenario. This article is a revised and selected version of the previous work.

Keywords: Industry 4.0, Context-awareness, Data Provisioning.

1 Introduction

Industry 4.0 is a new trend that drives the digitization of production environments. Especially data-driven cyber physical systems, advanced data analytics and Internet of Things [1], [2], [3] enable new approaches, such as self-organization of production processes. However, human interaction and decision making is still mandatory and beneficial in the smart factory [4], [5]. This especially holds in the domain of pre-production plants that manufacture the first prototypes of a product. Hence, lots of failures may occur that have to be quickly resolved by human workers. An efficient decision making process to find appropriate solutions to

⋆ Corresponding author

Reference: E. Hoos, P. Hirmer, B.Mitschang "Automated Creation and Provisioning of Decision Information Packages for the Smart Factory," Complex Systems Informatics and Modeling Quarterly, CSIMQ, no. 15, pp. 72–89, 2018. Available: https://doi.org/10.7250/csimq.2018-15.04

Additional information. Author's ORCID iD: P.Hirmer – orcid.org/0000-0002-2656-0095. Article PII S225599221800089X. Received: 14 February 2018. Accepted: 04 June 2018. Available online: 31 July 2018.

these failures has to consider different kinds of data. Examples are 3D-geometry data or a *bill of material* (BOM) representing the product structure, simulation data, process data, or measurement data of the exact dimensions of the product. These data are distributed onto heterogeneous Information Technology (IT) systems that are not integrated yet [6], [7], [8]. At the moment, workers have to manually collect all data relevant for decision making. We call this set of relevant data *decision information packages* (DIP). The composition of relevant data is a cumbersome task, since workers are trained to solve domain-specific problems, and not to find relevant data. This requires comprehensive IT skills. To address this issue, we provide an approach to automatically compose DIPs in a compact manner in order to relieve the decision maker from browsing complex IT systems and to redirect his focus on domain-specific problem-solving. We augment and go beyond existing approaches in providing relevant data via context-aware filtering [9], [10], [11] and data integration in engineering [6], [8] in order to fulfill domain-specific requirements. Key requirements are generation of DIPs without comprehensive IT knowledge; and integration of context-aware filtering into the existing engineering IT landscape. This article is the realization of the vision introduced in [12]. Detailed contributions of the article are:

(1) Definition of DIP Structure: We performed a case study in the pre-production plant to identify requirements regarding DIPs. The IT landscape is composed of heterogeneous IT systems. Each of them represents a specific view of product data. To combine these views and assign the appropriate data to it, we design a generic schema for DIPs reflecting the product structure as well. The focus on the product structure is important so that workers can easily interpret relevant data with respect to their current context.

(2) Context-Aware Composition of DIPs: Since there is a large amount of engineering data, it is important, for an efficient decision making process, that the amount of data is reduced and only relevant data is composed. Context data can be used to filter meaningful data. In the pre-production plant, for instance, the context of shop floor workers is dependent on their current task, location and of the state of the environment, which often can, e.g. be captured by sensors [13]. Hence, context data has to be linked to engineering data to derive meaningful data for decision making. This should be possible without comprehensive IT knowledge. To address these issues, we develop a context-aware engineering data model and operators to process the data. They abstract from technical details of IT systems and context data, which facilitates the linking by domain experts.

(3) Architectural Realization for the Engineering Domain: The architecture to automatically compose and acquire DIPs needs to be integrated into the existing engineering IT landscape which is composed of legacy systems and dynamically appearing IT systems. Therefore, we define a system architecture that realizes our approach. The architecture serves as a basis for our prototypical implementation, which is evaluated based on our real-world case scenario.

(4) Design and Prototypical Implementation of an Application to Provide DIPs: To provide DIPs to the worker in a location-independent manner, a mobile application is needed which applies the automatic generation of DIPs and displays the corresponding DIP to the user. In order to improve the decision making process, we define a goal regarding the implementation of a mobile app and perform requirements analysis. On this basis, we define the functionality and the user interface for the app. Furthermore, we prototypically implement the app as a proof of the concept.

This article is a revised and extended version of the paper "Context-Aware Decision Information Packages: An Approach to Human-Centric Smart Factories" as presented at the 21st European Conference on Advances in Databases and Information Systems (ADBIS) in 2017 [14]. The extension comprises a detailed scenario how to generate DIPs as well as a prototypical implementation of a mobile application to provide DIPs in the Smart Factory.

This article is structured as follows: Section 2 introduces the use case scenario and defines the structure of DIPs. Section 3 contains the main contribution of our article: an approach to compose DIPs. Section 4 introduces a system architecture, which serves as the basis for the evaluation of our approach conducted in Section 5. In Section 6, we introduce DIPPing, a mobile application to manage DIPs. Finally, Section 7 covers related work and Section 8 summarizes the article.

2 Decision Information Packages in Engineering

This section introduces our case study at a German car manufacturer, which emphasizes the demand and utility of DIPs. Based on this, we derive a generic schema for DIPs in the engineering domain.

2.1 Case Study: Pre-production Plant

The case study is part of the manufacturing of car prototypes, also known as *pre-production test*. During the production in a preproduction plant, the cars pass several assembly stations of a production line. At each station, a shop floor worker assembles multiple parts. Since the product and the process are not as well defined as in series production, plenty of failures may occur. For instance, parts cannot be assembled because the tolerances do not match. In the following, we describe a case scenario, in which the part "console" does not fit into the apparatus of the *front-end assembly station*, that assembles the front section of the car shell: The worker at the front-end assembly station recognizes the problem and has to resolve it. There could be many of causes for this problem. In the following and due to space reason, we only look into three representative kinds of information to identify an error:

1. Basic information about the part is required to identify name, version and material. These are stored in the *bill of material* (BOM) system, which contains information about all parts necessary to manufacture the product [15].
2. The visualization of the 3D-Geometry model is necessary to check the correctness of the part's geometry. The worker has to access the *product management system* (PDM) to find the appropriate 3D geometry file with respect to different versions and variants [16].
3. Measurement reports are necessary to check tolerances. Measurement reports are stored in a file system in PDF format and contain the exact dimensions of an assembly.

The corresponding DIP is shown in Figure 1 and discussed in the next subsection.

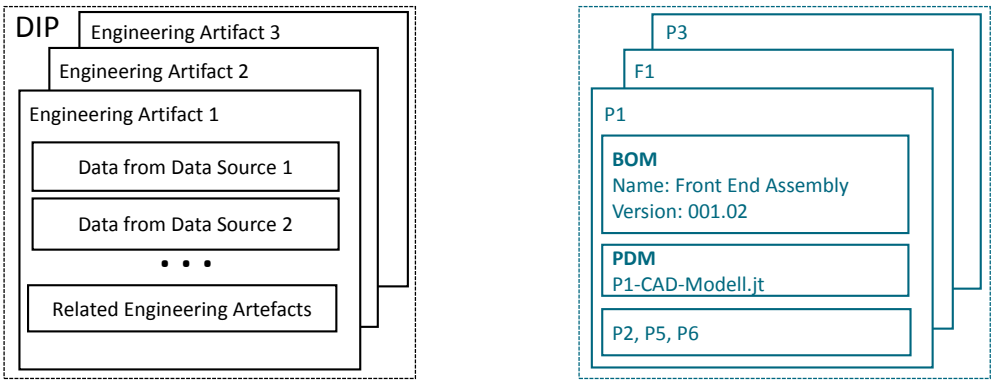


Figure 1. Left: Generic schema of the DIP, right: DIP of the case study

2.2 Decision Information Packages

DIPs provide required information for decision making. They compose data from multiple, heterogeneous data sources with respect to the context of the problem. Thereby, different views of the data should be combined. To enhance the comprehensibility of the worker, DIPs should reflect the product structure. The product structure defines the relation between parts. Accordingly, we use engineering artifacts as structuring elements. Engineering artifacts virtually represent all parts and components required to build a product, also including manufacturing parts. They have a unique identifier already used in product model and factory model. Figure 1 shows an abstract model of a DIP. A DIP encapsulates data of multiple *engineering artifacts*. For each engineering artifact, information from different data sources is composed. The data format of the data sources varies from relational data and XML data to file-based data. Also, related engineering artifacts are provided. For instance, related engineering artifacts of a part can be subcomponents or machines at which the part is manufactured.

In the depicted example, on the right of Figure 1, the DIP contains information about the engineering artifacts *P1*, *F1* and the *P3*. These are the unique identifiers. The name of the part *P1* and the current version number are acquired from the BOM. From the PDM system, the file containing the 3D model is included, called P1-CAD-Modell.jt, which is necessary to visualize the part. *P1* is related to *P2*, *P5*, and *P6*, which are subcomponents.

3 Context-Aware Composition of Decision Information Packages

This section presents, first, the context-aware engineering data model (CAEM) to link engineering data and context data, and, second, context-aware operators to compose the DIPs based on this model.

3.1 Context-Aware Engineering Data Model

In order to link engineering data and context data in the CAEM, we introduce a context model which structures the context data and enables to define situations of users. We use the definition of context models as introduced in [17]. The context model is based on *context entities* and *context attributes*, characterizing the entities. Furthermore, it contains relations between entities. Figure 2 shows a simple context model for the engineering domain. This simple model is sufficient since the focus of the article is not to provide a comprehensive context model, but rather to link the context with engineering data. Context entities are *station*, *actor*, and *project*. The actor is characterized by its context attributes *name*, *role*, and *task*. Each new product, e.g. a new car model, is defined as *development project*, which is divided into various phases such as pre-production test batches. The values of the attributes define the state of an entity. Situations derived from this context model can describe at which station an actor works, and also in what project. An exemplary situation for our case scenario is: (i) the actor has the role *prototype engineer*, (ii) the phase is the *pre-production test batch 2*, and (iii) the station is the *front-end assembly station*.

The *context-aware engineering data model* is shown in Figure 3. It models context values, engineering artifacts and data sources as entities as well as links between these entities. Note that the context value is the value of the context attribute defined in Figure 2, such as that "front-end-assembly station" is the value of the context attribute "station name". *Data sources* represent an abstraction of data sources, in which engineering data are stored, e.g. the bill of material. Engineering links describe the semantic relation between engineering artifacts, such as "is part of" or "is manufactured by". Data links describe in which data sources information about the engineering artifact is stored. Context links describe in which

context the engineering artifact is relevant. They can be established between context elements and data sources.

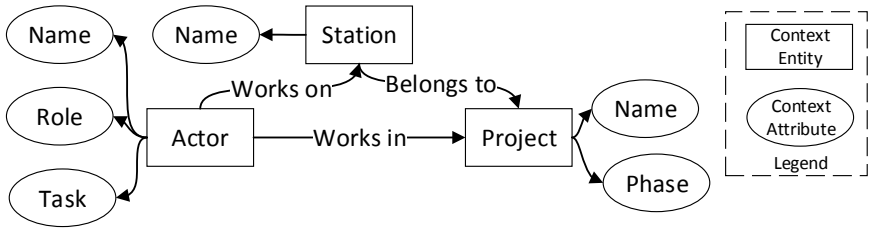


Figure 2. Our context model

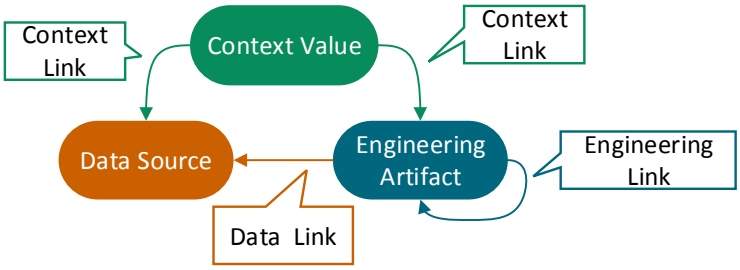


Figure 3. Context-aware engineering model

Figure 4 shows an excerpt of a CAEM instance according to our use case scenario. The semantics of the excerpt is as follows: (i) at the station *front-end assembly*, the part *front-end* is manufactured; (ii) the prototype engineer is interested in information from the data sources: BOM, PDM and reports, whereas the shop floor worker is only interested in BOM and PDM data; (iii) the front-end assembly station manufactures the right-hand-drive and the left-hand drive; (iv) at the pre-production test batch 2 (second test batch) only the right-hand-drive variants are manufactured.

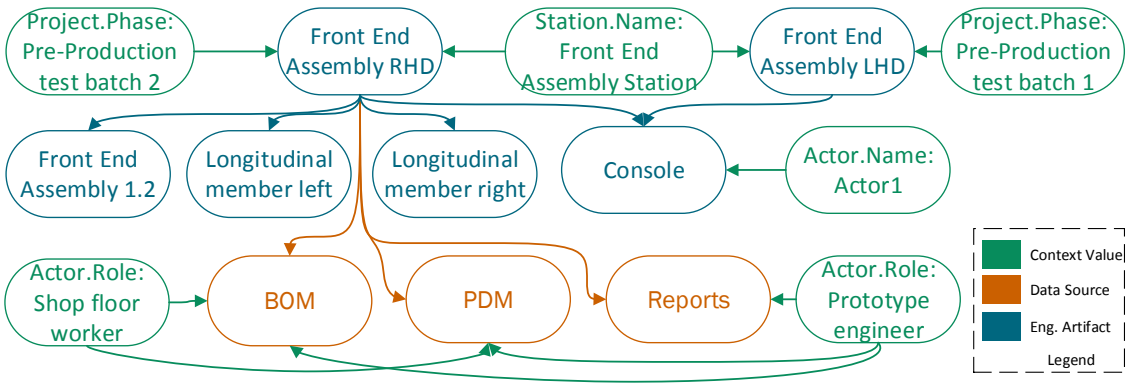


Figure 4. Excerpt of a CAEM instance for the use case scenario

3.2 Operators for Decision Information Package Composition

The composition of DIPs is supported by four operators which interact with the CAEM. In order to define operators, we interpret the CAEM as a set of items and define set-operators for it. According to our context-aware engineering model shown in Figure 3, we define three

sets:

$$EngineeringArtifacts : EA = \{ea_1, ea_2, ..., ea_n\}$$
$$ContextValues : CV = \{cv_1, cv_2, ..., cv_m\}$$
$$DataSources : DS = \{ds_1, ds_2, ..., ds_p\}$$

We model the links of the CAEM as relations between target and source artifacts.

$$ContextEALinks : CEA \subseteq \{(cv_i, ea_j) | cv_i \in CV, ea_j \in EA\}$$
$$ContextDSLinks : CDS \subseteq \{(cv_i, ds_j) | cv_i \in CV, ds_j \in DS\}$$
$$DataLinks : DL \subseteq \{(ea_i, ds_j) | ea_i \in EA \wedge ds_j \in DS\}$$
$$EngineeringLinks : EL \subseteq \{(ea_i, ea_j) | i \neq j \wedge ea_i, ea_j \in EA\}$$

We define a particular situation of a user X as $SIT_x = \{cv_{x1}, cv_{x2}, ...\}$.
Engineering Artifact Selection: The operator $SelectEA$ selects all engineering artifacts which are relevant for the given situation SIT_x:

$$SelectEA(SIT_x) = \bigcap_{cv_{xj} \in SIT_x} \{ea_i | (cv_{xj}, ea_i) \in CEA\} \quad (1)$$

Selection of relevant data sources: The operator $SelectDS$ filters the relevant data sources for a particular situation SIT_x:

$$SelectDS(SIT_x) = \{ds_i | (cv_{xj}, ds_i) \in CDS \wedge cv_{xj} \in SIT_x\} \quad (2)$$

Discovery of data sources: The operator $DiscoverDS$ identifies all data sources which provide information about the relevant engineering artifact ea:

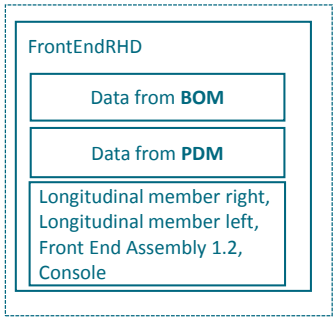
$$DiscoverDS(ea) = \{ds_i | (ea, ds_i) \in DL\} \quad (3)$$

Discovery of related engineering artifacts: The operator $DiscoverEA$ discovers engineering artifacts that are related to another particular engineering artifact:

$$DiscoverEA(ea_j) = \{ea_i | (ea_i, ea_j) \in EL \vee (ea_j, ea_i) \in EL\} \quad (4)$$

To compose a DIP for a given situation SIT_x, $SelectEA$ creates the list of engineering artifacts relevant for the situation. This operator uses an intersection to be more restrictive and reduces the number of engineering artifacts more than using union. Afterwards, the data sources are determined using $SelectDS$ or $DiscoverDS$. If $SelectDS$ has an empty set as the result, which means that no context value is assigned to a data source, $DiscoverDS$ explores the data sources. Finally, $DiscoverEA$ finds the related engineering artifacts for each relevant engineering artifact defined by $SelectEA$. Figure 5 visualizes a DIP and how to construct it using the operators. Furthermore, it shows the result for applying the operators to the context model.

Note that all these operators are set-oriented, because the result of an operation execution may end in a set of selected or discovered EAs and DSs. Furthermore, the operators let to hide from, e.g. connectivity, data source types and access mechanisms to the data sources. Hence, the realization approach of the operators has to cope with this abstract operator level and provide a transformation to the underlying IT environment, e.g. to the sources of the systems. All these issues are covered in the next section.



Given: Situation $SIT_1 = \{shopfloorWorker, preproductionTestBatch2, FrontEndAssemblyStation\}$

Construction:

(1) Relevant Engineering Artifacts:
$SelectEA = \{FrontEndRHD\}$

(2) Relevant Data Sources for $FrontEndRHD$:
$SelectDS = \{BOM, PDM\}$

(3) Related Engineering Artifacts:
$DiscoverEA = \{Longitudinal\ member\ right, Longitudinal\ member\ left, Front\ End\ Assembly\ 1.2, Console\}$

Figure 5. Construction of a DIP using the operators with respect to the use case scenario

4 Architecture to Provide Decision Information Packages

In order to realize our approach, we introduce the system architecture depicted in Figure 6. The architecture enables to integrate the concept of decision information packages into the engineering environment. The environment is characterized by the users and their applications and the engineering context, which comprises all data in the engineering domain such as data stored in IT systems and context data such as sensor data, machine data, and user data. The architecture consists of three components, namely *Context-Aware Provisioning*, *Resource Access Platform* and *Context Management*.

Environmental context is gathered by the *Context Management*, which processes the low-level engineering context into higher-level context, also known as situations. For instance, it derives given GPS coordinates of a user into the situation that the user "is in the manufacturing plant". Note that we do not focus on the Context Management and assume that there is an appropriate implementation available such as the ones introduced in [18] and [13].

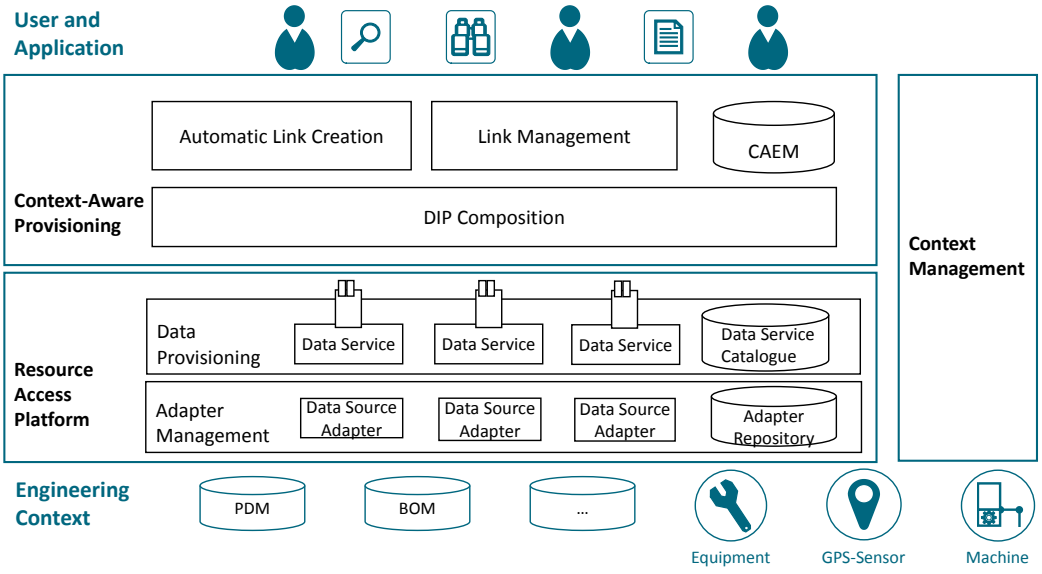


Figure 6. System Architecture to provision DIPs

4.1 Context-Aware Provisioning

The context-aware provisioning layer consists of a database managing context-aware engineering data model (CAEM) and of the subcomponents *DIP Composition*, *Link Management* and *Automatic Link Creation*.

In order to realize the CAEM, we use an entity-relationship model to implement it into a relational database. The *link management* provides an API to create and delete links as well as to store the engineering artifacts, context elements, and data sources in the CAEM. This simplifies the definition of context links and engineering links for domain experts. The *automatic link creation* facilitates the creation of context links, which is a high effort task when conducted manually. It enables to create a set-based definition of links, for instance, if the engineering artifacts have common attribute values. Second, it allows to import relations from other systems. For instance, sometimes the information about which part is manufactured, and on which station, is stored in another IT system. Engineering links could be also derived from a system model describing the relationships between different parts and manufacturing equipment. In addition, data links can be derived from this model, since it stores, e.g. the relationships between the engineering artifacts and its simulation model [19]. Finally, it is possible to integrate learning mechanisms so that the domain expert gets proposals for context links if relationships with similar engineering artifacts exist. The *DIP Composition* component gathers all the information required to compose a DIP as shown in Figure 5. It receives the situation from the context management. According to the situation determined by the context values, the DIPs are constructed using the operators.

4.2 Resource Access Platform

The Resource Access Platform (RAP) serves as a single entry point to access data sources through uniform interfaces. The RAP provisions data sources to the DIP Composition. The Resource Access Platform consists of two components: (i) the *adapter management component*, and (ii) the *data provisioning component*.

The *adapter management component* is responsible for binding data sources to the Resource Access Platform. This component consists of the Data Source Adapter Repository that stores adapters used for binding, and of a runtime environment to deploy and execute them. Adapters can be automatically and dynamically deployed using software provisioning technologies such as TOSCA [20]. To bind data sources, adapters for each data source are extracted from the Data Source Adapter Repository, are parameterized, and are then automatically deployed. The *data provisioning component* is accessed by the DIP Composition and contains the *Data Service Catalog* that provides meta-information about all available data sources. Furthermore, it contains *Data Services* that offer access to the actual data. First, the RAP is accessed by the DIP Composition in order to search for relevant data sources found in the CAEM in the Data Service Catalog. The RAP then provides references to corresponding Data Services that encapsulate access to underlying data using proper adapters.

The Resource Access Platform is based on the Resource Management Platform (RMP) as introduced by Hirmer et al. [21]. We extend this platform to support specific needs of the engineering domain. By doing so, we support data sources of the engineering domain, e.g. CAD models, sensor data, or simulation data. In addition, we automate the lookup in the Data Service Catalog, which was previously done manually. The interface to applications accessing the RAP (e.g. the DIP Composition) remains the same.

4.3 DIP Composition – Example Scenario

In order to clarify how DIPs can be created based on the introduced architecture, we describe an examplary scenario based on our case study (see Section 2.1). The corresponding procedure for this examplary scenario is depicted in Figure 7. For better clarity, unnecessary components have been omitted in this architecture. In this scenario, a shopfloor worker encounters an issue when assembling the car shell and requires a DIP in order to get all information necessary to solve this issue. In this case,

the part does not fit into the apparatus of the car shell. The situation of the worker is $SIT_1 = \{shopfloorWorker, FrontEndAssembly, FrontEndAssemblyStation\}$, which means that the *shopfloor worker* is working on the *front end assembly* of the car shell at station *FrontEndAssemblyStation*.

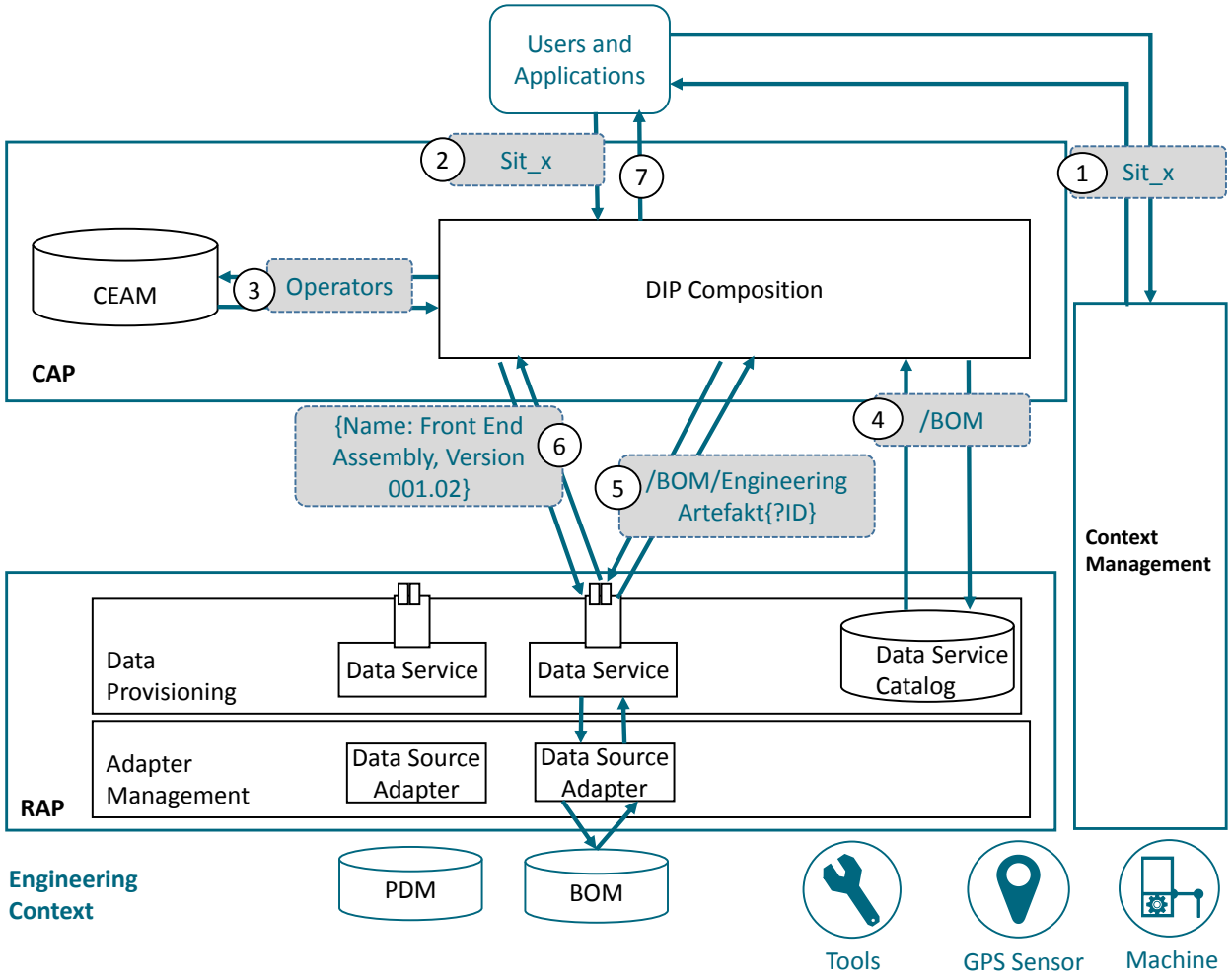


Figure 7. Example scenario to compose DIPs

In order to create the DIP for this scenario, different steps need to be conducted. These steps, also depicted in Figure 7, are:

1. The application for which a DIP is generated, in this case, for instance, a mobile application used by the shopfloor worker, picks up the situation $SIT_1 = \{shopfloorWorker, FrontEndAssembly, FrontEndAssemblyStation\}$ from the context management component.
2. This situation SIT_1 is passed to the DIP composition.
3. Taking the situation into account, the operators described in Section 3.2 are executed, identifying the relevant engineering artifacts, relevant data sources, and dependent engineering artifacts.
4. In the next step, relevant data is extracted from the data sources. By applying the operators, it is known that the BOM and PDM data sources are required to obtain information about the unfitting part. To realize this, the DIP composition component queries the data service catalog of the RAP for the appropriate data services of the BOM and PDM data sources and gets its REST URI back. The REST URI of the BOM data source for the case scenario is "/BOM".

5. Afterwards, information how the corresponding REST resource needs to be invoked in order to get the data of an engineering artifact is requested at the RAP. This is done by a "HTTP GET request" using the URI "/BOM". This request gets back the possible operations regarding the REST interface of the data service, which is "/BOM/EngineeringArtifact?ID" for the BOM system. This operation can be used to get information about a specific engineering artifact with the corresponding ID.
6. The REST call "GET/BOM/EngineeringArtifact?ID=EA1" now extracts the data via the data service from the BOM data source using the corresponding adapter. In this example, the DIP composition component receives the JSON object $\{name : FrontEndAssembly, version : 001.002\}$.
7. The DIP composition collects the data and passes it to the DIP application.

Note that the results of these steps are examplary for this scenario, e.g. the URI of the REST call; and differ in other application scenarios. Through the described steps, DIPs can be generated for this or any other scenario in the industrial domain.

5 Evaluation

We evaluate our approach according to the goal of DIPs which is to improve decision making on the shopfloor. With our approach, we enable to automatically compose and provide DIPs in the engineering domain. We aim to reduce the amount of information according to the context of the users in order to provide compact DIPs. This is important in order to relieve workers from meaningless information. Therefore, we apply our approach on a real data set of the introduced case scenario and investigate the reduction of information.

5.1 Case-oriented Evaluation of Information Reduction

In order to evaluate how well context can be used to filter relevant data, we apply our approach on a real data set and determine the information reduction. The data set originates from our case study introduced in Section 2. We look into data to assemble the bottom of the car shell. Table 1 shows the number of items in the CAEM. Engineering artifacts are the parts of the bottom of the car shell. The context values are belonging to the entities *station*, *actor*, and *project*. The data sources are BOM, PDM, and measurement reports. Engineering links describe the *part-of* relationships of the parts. In order to evaluate how well context can be used to filter data, we investigate the compactness of the DIPs. The compactness of DIPs is dependent on the *number of engineering artifacts* and *the data size of the information packages* of engineering artifacts. We assume, the smaller the DIPs are, the more effective the context-aware filtering.

Table 1. CAEM data

Data Type	Number
EngineeringArtifacts	785
ContextAttributes values	88
ContextLinks	5261
EngineeringLinks	964
DataLink	2355

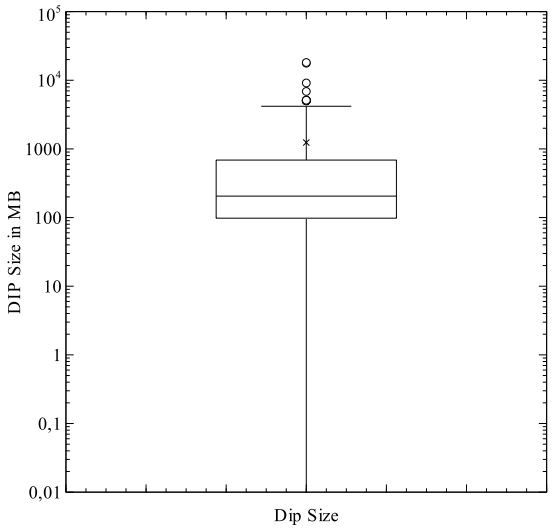
Reduction of Number of Engineering Artifacts We analyze the reduction number of engineering artifacts per DIPs using different kinds of situations. A situation consists of values from the names of actors and stations as well as from the phases of projects. We

have calculated 1348 DIPs and counted the number of engineering artifacts per DIP. The results are shown in Table 2. The first three columns indicate which context attributes are used to define a situation. We build every valid situation according to the context attributes. Column *number of DIPs* reflects only DIPs for valid situations. The next column reports the *average number of EAs per DIP*. Without our approach, in the worse case, a human worker would have to examine all 785 engineering artifacts. The results show that with two different kinds of context attributes, we get comparatively small DIPs, which contain between 1 and 6 engineering artifacts in average. We conclude that using at least two different kinds of context values reduces the number of engineering artifacts by a factor of 100.

Table 2. Reduction of engineering artifacts

Station	Phase	Actor	Number of DIPs	Average number of EA per DIP
yes	no	no	34	3.44
no	no	yes	41	9.41
no	yes	no	13	366
yes	no	yes	46	2.3
yes	yes	no	383	1.17
no	yes	yes	453	6.5
yes	no	yes	46	2.3
yes	yes	yes	332	1.17

Reduction of the DIP size We also investigate the DIP sizes resulting from the considered situations. Therefore, we calculate the DIP size for each possible situation based on two or three different kinds of context attributes. In our case, the critical file size is the one of the geometry model, which is required to visualize the 3D-Model of the car. We neglect data from the BOM since they are only key-value pairs and the measurement reports, since their size is constant. Figure 8 shows the results via a box-plot. Despite few huge outliers, such as a DIP with size of 18.12 GB, over half of the DIPs range between 97 MB and 687 MB.



Outlier	18129,58 17721.37; 9129,36; 6868,21;
Max	1,408.83
3.Quartile	687.87
Median	205.29
1.Quartile	97.95
Min	0.057

Figure 8. Boxplot of DIP size

The evaluation of the reduction shows that the size of DIPs can be reduced significantly by filtering them via the context. This enables a more efficient decision making process

because the worker does not have to browse meaningless data. Station, actor and phase are appropriate types of context, since they reduce the number of EAs in DIPs drastically and, thus, influence the size of DIPs.

6 DIPPing – A Mobile Application to Provide DIPs

This section describes a mobile application, named DIPPing (Decision Information Packages Provisioning App), that we developed to provide DIPs. In order to build the DIPPing application, we first defined a goal with respect to the process improvement and then conducted a requirements analysis. Based on this analysis, we defined the functions of the application and specified implementation aspects. As a result, we designed and implemented the application's user interface. This mobile application was prototypically implemented as part of a cooperation project with a German car manufacturer.

6.1 Goal

As a basis for the requirement analysis of the DIPPing application, we use the previously described scenario regarding the case study in the pre-production plant (see Section 2).

There are two major weaknesses in the case study. On the one hand, the manual creation of a DIP is time-consuming and knowledge-intensive, as the information has to be gathered from many heterogeneous data sources. On the other hand, the information can only be retrieved at the in-house engineer's office so that the engineer has to cycle back and forth between the workplace and the station where the issue occurs.

These weaknesses should be addressed by the DIPPing application. For this purpose, the DIPPing app was designed specifically to be available on mobile devices and to generate and display DIPs by using the previously introduced architecture. To evaluate if it is benefical to use mobile devices, we performed the ValueApping Method [22]. Consequently, the goal of the DIPPing app is composed of the following sub-goals:

- DIPs should be generated automatically based on the current situation. The implementation should follow the DIP approach and use the corresponding architecture (see Section 4.3).
- DIPs should be available on the go, so that the decision making can take place location-independently and the running distances are saved.
- DIPs should be displayed in a clear manner, considering reduced screen size of mobile applications.

6.2 Requirement Analysis

Based on the goal, the functions of the DIPPing app are defined. These functions are sequentially executed. An overview is given in Figure 9. In detail, the functions are:

1. *Capture Situation:* The current situation should automatically be recorded by the DIPPing app, i.e. which worker works at which station, handling which specific part and variant.
2. *Check Situation:* The detected situation should be shown to the user. The user can manually check if the situation is correct and adjust it as needed. This is required in order to handle the uncertainty of captured context values.
3. *Generate DIP*: The DIP is generated using the context-aware provisioning layer of our architecture as introduced in Figure 6.
4. *Display DIP*: The app displays the DIP produced by the DIP composition component with respect to the DIP structure introduced in Section 2. This is necessary in order to guarantee that the users will easily understand the given information.

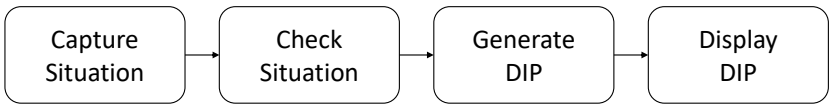


Figure 9. Functionality of DIPPing

6.3 DIPPing Implementation

In the following, we describe implementation aspects of the DIPPing app with respect to the required functions defined in the previous sub-section.

Capture Situation: We decided to use beacons [23] to realize this function. These beacons can be applied to automatically detect at which station a worker currently works. To do that, the stations are equipped with beacons. The DIPPing app can then detect the nearest beacon and then determine the current station of the worker. Finding out the role and expertise of this worker can be determined by an authorization function of the DIPPing application. Which variant is currently being produced or which part is involved can be found out using, e.g. QR-Codes.

Check Situation: After the situation was automatically determined, the DIPPing user can manually check the situation. By doing so, the user verifies that the detected situation provided by the DIPPing app is correct. Otherwise, the situation can be revised. For this purpose, two additional options for context capture were implemented: scanning QR codes and adding information manually.

Generate DIP: The function to generate DIPs is implemented using the introduced DIP architecture (see Figure 6) serving as the basis for this implementation. The implementation of the *context-aware provisioning* is as follows: The CAEM itself is implemented as a relational model used as schema for an SQLite[3] database, which can be accessed through a Java-based interface connecting to the database with the corresponding SQLite driver. The Java interface is hosted on an Apache Tomcat application server. For the implementation of the *Context Management*, we used the existing tool SitOPT [18], [13], which provides an interface to register on situations that are derived based on context data. The *Resource Access Platform* consists of two components. For the implementation of the adapter layer, we use MongoDB[4] as adapter repository as well as an Apache Tomcat Java runtime environment for the adapters. Hence, all adapters are implemented in Java, exclusively. For the deployment of adapters, we use the TOSCA runtime OpenTOSCA [24]. To implement the data provisioning layer, we use Java REST services as well as a Java-based implementation of the Data Service Catalog, which stores its data into a MySQL database. The DIP itself is generated based on this implementation and as described in Section 4.3.

Display DIP: In order to realize the display of the DIP, a mobile user interface was designed on the basis of the DIP structure (see Figure 1). The presentation is separated into two layers: On the first layer, all relevant engineering artifacts for the current situation are shown and, on the second layer, the corresponding data for each relevant engineering artifact are displayed. The details are described in the next section.

6.4 User Interface of DIPPing

We provide a user interface for the DIPPing app which is depicted in Figure 10. This user interface contains the following screens:

[3] https://sqlite.org/
[4] http://mongodb.com/

- Screen 1: This screen shows the automatically detected situation. The displayed situation is structured according to the context model using context entities and their attributes (see Figure 2). For a better understanding, context entity icons are used as specified in [17]. Furthermore, it contains the functionality of adding manual context values using the button *plus* and scanning QR-Code.
- Screen 2: This screen shows the first level of the DIP structure, i.e. the relevant engineering artifacts for a specific situation. For instance, this can be the parts involved in the problem.
- Screen 3: This screen shows a detailed view of an engineering artifact. It displays information from the BOM, the PDM system, and various measurement reports. The CAD geometries can be shown via the button *Visualize*. Measurement reports are listed and can be opened in the PDF format by clicking on them.

The user interface was implemented as a web application. For this purpose, the technologies HTML5, CSS, and JavaScript were used. The advantage of implementing a web application is that the implementation is independent of the operating system of the mobile application.

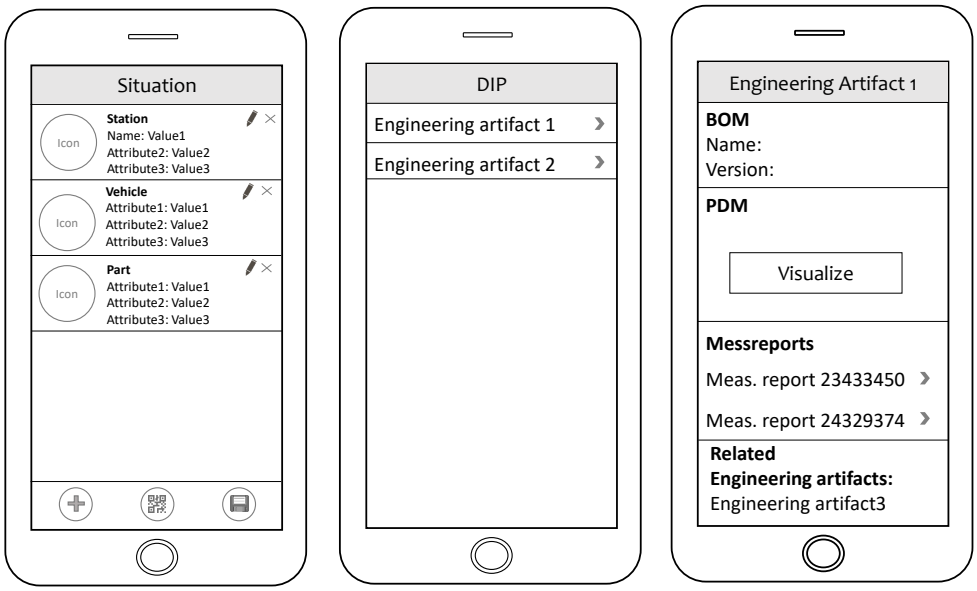


Figure 10. User interface of the DIPPing prototype

6.5 Evaluation of DIPPing

In order to evaluate the usability and the benefits of DIPPing, we conducted an expert evaluation in the pre-production plant. Therefore, six experts were interviewed based on the prototype demonstration of DIPPing. The experts differed in age, gender, responsibilities and process they work in. We identified the following benefits with respect to the process in the pre-production plant:

- *Reduction of traveling distance:* Normally the engineer has to go to his office in order to search for the information. With DIPPing he is able to search directly at the station and has not to travel anymore. All experts revealed that the app reduced the time of finding a solution extremely.
- *Reduction of information search time:* Due to the provision of DIPs, the engineers have all required information for their task at once and do not have to search in different IT-Systems. However, the interviews showed that this time reduction is dependent on the

knowledge about the IT-system. Persons who have good knowledge in the IT-Systems did not see a search time reduction in contrast to persons without that knowledge.

- *Availability of engineering data:* The interview participants pointed out that the quality of their decision were increased since they could directly compare the as-is state of the car with the should-be state defined in the engineering data.
- *Flexibility of information search:* The experts highlighted the fact, that with DIPPing they could perform the search anytime and anywhere they wanted. In the past, it was often the case, that they were unscheduled in the pre-production plant and were asked by the shopfloor workers for possible solutions of their current problems.

7 Related Work

The DIP approach can be seen as virtual integration. Yet, in contrast to federated database systems [25] and recent ontology-based integration approaches [26], [27], DIPs do not require a complex mediated schema nor schema matching process to include domain experts without IT knowledge into the integration process. With respect to related work, we differentiate four groups according to automatic context-aware provisioning of DIPs. The first group reviews approaches to abstract data sources for automatic data provisioning. The second group comprises approaches in the engineering domain that acquire engineering data in the client side without filtering relevant data. The third and fourth groups focus on filtering relevant data either on data level or on application level.

A lot of middleware platforms have been developed that abstract from accessing data sources [28], [29], [30], belonging to the first group. However, none of the existing approaches provides the functionality needed to provide DIPs. First of all, dynamic environments of the engineering domain need to be handled with frequently (dis-)appearing data sources. In this article, we introduce the RAP to bind these data sources dynamically through adapters that are deployed using TOSCA. Furthermore, the RAP is generic, i.e. it supports all kinds of sources assuming that a corresponding adapter exists for them. Consequently, the RAP fits our approach very well and was used instead of other approaches.

Approaches belonging to the second group, provide data acquisition from multiple engineering IT systems. Katzenbach et. al introduce a common engineering client, where data are provisioned by an engineering service bus [6]. Similarly, the authors of [8] suggest a system-level integration using standards and harmonized human interfaces. However, none of them consider filtering relevant data.

The third group integrates the context into data sources. Martinenghi and Torlone [31] and Roussous et al. [32] extend the relational data model with its context and define an appropriate query language. Stavrakas et al. [33] integrate the context into XML by developing a multidimensional semi-structured data model and a query language to process the context data [34]. Since all approaches integrate the context into an existing data model, there is no generally defined context model involved. Furthermore, the databases have to be adapted to integrate the context in contrast to our approach which integrate the context virtually. The fourth group addresses linking the context and data on the application level. Bobillo et al. [11] develop a model to manage context-relevant knowledge in ontologies. This is based on the domain ontology of the knowledge-based system and on a context ontology. Barkat et al. [10] define a context ontology to integrate context into the semantic databases, called OntoDB. Hence, their approach is restricted to applications using exactly this database. Bolchini et al. [35] introduce a context ontology based on a self-developed context model to define the portions of the ontology which are relevant. Similar to this, they provide a method to define context-aware views for relational databases [9]. Hence, many related approaches try to achieve similar goals regarding context-aware filtering of relevant data using ontology models. In our approach, we decided to omit the use of ontologies to reduce the complexity.

Most advantages of ontologies come with reasoning and linking to other ontologies. For our approach, a simple meta-model is sufficient.

8 Summary

In this article, we introduce an approach to compose and provide *decision information packages* (DIPs) to support problem resolving in the smart factory. We introduce DIPs on the basis of a real use case at a German car manufacturer. DIPs are constructed using a meta-model and corresponding operators. The meta-model links engineering data to context data. The operators process this model to find data relevant for specific situations, e.g. of a shop floor worker being faced with issues assembling a specific part. To realize this approach and to integrate it into the engineering domain, we introduce a system architecture. The approach is evaluated through a prototypical implementation to demonstrate its feasibility and a case-oriented evaluation using real data to highlight the compactness of DIPs. Our approach leads to a significant reduction of amount of information.

This article is an extended version of the paper "Context-Aware Decision Information Packages: An Approach to Human-Centric Smart Factories" presented at the 21st European Conference on Advances in Databases and Information Systems (ADBIS) 2017 [14].

Acknowledgments

This work is partially funded by the German Research Foundation (DFG) as part of the Graduate School of Excellence Advanced Manufacturing Engineering project GSaME-C2-008.

References

[1] N. Jazdi, "Cyber physical systems in the context of Industry 4.0," in *Automation, Quality and Testing, Robotics, 2014 IEEE International Conference on*, 2014. [Online]. Available: https://doi.org/10.1109/aqtr.2014.6857843

[2] C. Gröger, "Building an industry 4.0 analytics platform," *Datenbank-Spektrum*, vol. 109, no. 2, p. 5, 2018. [Online]. Available: https://doi.org/10.1007/s13222-018-0273-1

[3] O. Vermesan and P. Friess, *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. River Publishers, 2013.

[4] D. Lucke, C. Constantinescu, and E. Westkämper, "Smart factory - a step towards the next generation of manufacturing," in *Manufacturing Systems and Technologies for the New Frontier*, M. Mitsuishi, K. Ueda, and F. Kimura, Eds. London: Springer London, 2008, pp. 115–118. [Online]. Available: https://doi.org/10.1007/978-1-84800-267-8_23

[5] C. Gröger, L. Kassner, E. Hoos, J. Königsberger, C. Kiefer, S. Silcher, and B. Mitschang, "The Data-driven Factory - Leveraging Big Industrial Data for Agile, Learning and Human-centric Manufacturing," in *ICEIS 2016*, vol. 1, pp. 40–52, 2016. [Online]. Available: https://doi.org/10.5220/0005831500400052

[6] A. Katzenbach, "Automotive," in *Concurrent Engineering in the 21st Century*. Springer, pp. 607–638, 2015. [Online]. Available: https://doi.org/10.1007/978-3-319-13776-6_21

[7] R. Stark, H. Hayka, J. H. Israel, M. Kim, P. Müller, and U. Völlinger, "Virtuelle Produktentstehung in der Automobilindustrie," *Informatik-Spektrum*, 2011. [Online]. Available: https://doi.org/10.1007/s00287-010-0501-z

[8] D. Trippner, S. Rude, and A. Schreiber, "Challenges to Digital Product and Process Development Systems at BMW," in *Concurrent Engineering in the 21st Century*. Springer, pp. 555–569, 2015. [Online]. Available: https://doi.org/10.1007/978-3-319-13776-6_19

[9] C. Bolchini, E. Quintarelli, F. A. Schreiber, and M. T. Baldassarre, "Context-Aware Knowledge Querying in a Networked Enterprise," in *Methodologies and technologies for networked enterprises. Lecture Notes in Computer Science*. Vol. 7200, Springer, pp. 237–258, 2012. [Online]. Available: https://doi.org/10.1007/978-3-642-31739-2_12

[10] O. Barkat and L. Bellatreche, "Linking Context to Ontologies," in *2015 11th International Conference on Semantics, Knowledge and Grids (SKG)*, 2015. [Online]. Available: https://doi.org/10.1109/skg.2015.36

[11] F. Bobillo, M. Delgado, and J. Gómez-Romero, "Representation of context-dependant knowledge in ontologies: A model and an application," *Expert Systems with Applications*, 2008. [Online]. Available: https://doi.org/10.1016/j.eswa.2007.08.090

[12] E. Hoos, P. Hirmer, and B. Mitschang, "Improving Problem Resolving on the Shop Floor by Context-Aware Decision Information Packages," in *In Proceedings of the CAiSE 2017 Forum*, 2017.

[13] P. Hirmer, M. Wieland, H. Schwarz, B. Mitschang, U. Breitenbücher, S. G. Sáez, and F. Leymann, "Situation recognition and handling based on executing situation templates and situation-aware workflows," *Computing*, vol. 99(2), pp. 163–181, 2016. [Online]. Available: https://doi.org/10.1007/s00607-016-0522-9

[14] E. Hoos, P. Hirmer, and B. Mitschang, "Context-Aware Decision Information Packages: An Approach to Human-Centric Smart Factories," in *Proceedings of the 21st European Conference on Advances in Databases and Information Systems (ADBIS)*. Springer International Publishing AG 2017, August 2017, Konferenz-Beitrag, pp. 42–56. [Online]. Available: https://doi.org/10.1007/978-3-319-66917-5_4

[15] H. Hegge and J. C. Wortmann, "Generic bill-of-material: A new product model," *International Journal of Production Economics*, vol. 23(1–3), pp. 117–128, 1991. [Online]. Available: https://doi.org/10.1016/0925-5273(91)90055-x

[16] J. Stark, *Product Lifecycle Management*. Springer International Publishing, 2015.

[17] E. Hoos, M. Wieland, and B. Mitschang, "Analysis Method for Conceptual Context Modeling Applied in Production Environments," in *Proceeding of the 20th International Conference Business Information Systems, Lecture Notes in Business Information Processing* , vol. 288, pp. 313–325, 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-59336-4_22

[18] M. Wieland, H. Schwarz, U. Breitenbücher, and F. Leymann, "Towards Situation-Aware Adaptive Workflows," *PerCom*, 2015.

[19] M. Eigner, T. Dickopf, H. Apostolov, P. Schaefer, K.-G. Faißt, and A. Keßler, "System lifecycle management: Initial approach for a sustainable product development process based on methods of model based systems engineering," in *A Short Portable PLM Course*, ser. IFIP Advances in Information and Communication Technology, J. Sauza Bedolla, J. Martinez Gomez, and P. Chiabert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, vol. 442, pp. 287–300. [Online]. Available: https://doi.org/10.1007/978-3-662-45937-9_29

[20] OASIS, "Topology and orchestration specification for cloud applications."

[21] P. Hirmer, M. Wieland, U. Breitenbücher, and B. Mitschang, "Automated Sensor Registration, Binding and Sensor Data Provisioning," in *Proceedings of the CAiSE 2016 Forum*, 2016.

[22] E. Hoos, C. Gröger, S. Kramer, and B. Mitschang, "Valueapping: An analysis method to identify value-adding mobile enterprise apps in business processes," in *Enterprise information systems*, ser. Lecture Notes in Business Information Processing, J. Cordeiro, S. Hammoudi, L. Maciaszek, O. Camp, and J. Filipe, Eds. Cham: Springer, 2015, vol. 227, pp. 222–243. [Online]. Available: https://doi.org/10.1007/978-3-319-22348-3_13

[23] Y. Sung, J. Kwak, Y.-S. Jeong, and J. H. Park, "Beacon distance measurement method in indoor ubiquitous computing environment," in *Advances in Parallel and Distributed Computing and Ubiquitous Services*, J. J. J. H. Park, G. Yi, Y.-S. Jeong, and H. Shen, Eds. Singapore: Springer Singapore, 2016, pp. 125–130. [Online]. Available: https://doi.org/10.1007/978-981-10-0068-3_15

[24] T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, A. Nowak, and S. Wagner, "OpenTOSCA - A Runtime for TOSCA-based Cloud Applications," in *ICSOC*, 2013. [Online]. Available: https://doi.org/10.1007/978-3-642-45005-1_62

[25] A. P. Sheth and J. A. Larson, "Federated database systems for managing distributed, heterogeneous, and autonomous databases," *ACM Computing Surveys*, 1990. [Online]. Available: https://doi.org/10.1145/96602.96604

[26] N. F. Noy, "Semantic integration," *ACM SIGMOD Record*, 2004.

[27] M. Wauer, J. Meinecke, D. Schuster, A. Konzag, M. Aleksy, and T. Riedel, "Semantic Federation of Product Information from Structured and Unstructured Sources," in *Web-based multimedia advancements in data communications and networking technologies*. IGI Global, 2013. [Online]. Available: https://doi.org/10.4018/978-1-4666-2026-1.ch008

[28] A. Langegger, W. Wöß, and M. Blöchl, *A Semantic Web Middleware for Virtual Data Integration on the Web*. Springer Berlin Heidelberg, 2008, pp. 493–507. [Online]. Available: https://doi.org/10.1007/978-3-540-68234-9_37

[29] A. C. P. Barbosa, F. A. A. Porto, and R. N. Melo, "Configurable data integration middleware system," *Journal of the Brazilian Computer Society*, 2002. [Online]. Available: https://doi.org/10.1590/s0104-65002002000200002

[30] A. Grant, M. Antonioletti, A. C. Hume, A. Krause, B. Dobrzelecki, M. J. Jackson, M. Parsons, M. P. Atkinson, and E. Theocharopoulos, "Ogsa-dai: Middleware for data integration: Selected applications," in *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, 2008. [Online]. Available: https://doi.org/10.1109/escience.2008.131

[31] D. Martinenghi and R. Torlone, "Querying Context-Aware Databases," in *Flexible Query Answering Systems. FQAS 2009. Lecture Notes in Computer Science* . vol. 5822, Springer, 2009. [Online]. Available: https://doi.org/10.1007/978-3-642-04957-6_7

[32] Y. Roussos, Y. Stavrakas, and V. Pavlaki, "Towards a Context-Aware Relational Model," in *International Workshop on Context Representation and Reasoning*, 2005.

[33] Y. Stavrakas and M. Gergatsoulis, "Multidimensional Semistructured Data: Representing Context-Dependent Information on the Web," in *Advanced information systems engineering*. Springer, 2002. [Online]. Available: https://doi.org/10.1007/3-540-47961-9_15

[34] Y. Stavrakas, K. Pristouris, A. Efandis, and T. Sellis, "Implementing a Query Language for Context-Dependent Semistructured Data," in *Advances in databases and information systems*. Springer, 2004. [Online]. Available: https://doi.org/10.1007/978-3-540-30204-9_12

[35] C. Bolchini, C. Curino, F. A. Schreiber, and L. Tanca, "Context Integration for Mobile Data Tailoring," in *7th International Conference on Mobile Data Management (MDM'06)*, 2006. [Online]. Available: https://doi.org/10.1109/mdm.2006.52