Teamleader App – a Collaborative System Allowing Ad-Hoc Planning Decisions

Sönke Knoch, Pascal Lessel, Melanie Reiplinger, Marcel Köster, Vladimir Pavlov
German Research Center for Artificial Intelligence
Research Department Intelligent User Interfaces
Saarbrücken, Germany
soenke.knoch@dfki.de, pascal.lessel@dfki.de, melanie.reiplinger@dfki.de, marcel.koester@dfki.de, vladimir.pavlov@dfki.de

Leenhard Hörauf, Christoph Speicher
Centre for Mechatronics and Automatisation gGmbH (ZeMA)
Group of Assembly Systems and Automatisation Technology
Saarbrücken, Germany
leenhard.hoerauf@mechatronikzentrum.de, christoph.speicher@mechatronikzentrum.de

Rouven Vierfuß, Torben Joppien
Miele
imperial-Werke oHG
Bünde, Germany
rouven.vierfuss@imperial.de, torben.joppien@imperial.de

***Abstract*—This paper proposes a collaborative system that provides decision support for team leaders in an industrial production scenario where staff planning needs immediate adaptations. Requirements were gathered and led to use cases that specify the solution, a mobile application called Teamleader App. A system with an advanced communication protocol was developed, which integrates several sub-systems, such as a supporting domain model and production simulation supporting workers with forecasts of potential decisions. A case study was conducted, which provides first insights into the suitability of the presented solution and revealed further aspects that will improve the system.**

Keywords-Decision support; staff planning; domain model; interaction design; user experience.

I. INTRODUCTION

Products highly individualized to fit consumers' needs require flexible production strategies that allow small batch sizes. To deliver manufactured products on time, the person responsible for production has to ensure that the number of planned pieces will be achieved. During production, multiple events occur that may have an impact on the production target. Information technology can help to handle such events. First mobile applications were developed that support people in selected, problem-specific tasks, such as staff planning [1], issue tracking [2] and perfomance measuring [3].

This work is a part of activities towards Industry 4.0 (see [4] and [5]) that bridge the gap between real and virtual worlds and foster the utilization of the potential of the Internet of Things [6]. Industry 4.0 describes the future industrial production as a production of highly individualized products [7] and with an increasing flexibility of production processes. These developments involve a challenge for staff planning engineers who have to answer this flexibility in production with an adaptive staff plan strategy. Increasing quantities of data and complexity make this planning process even harder.

The coordination between planning engineers—in this scenario called team leaders—to find additional qualified workers within the factory consumes time and is a difficult task. To assist team leaders in this process, the concept of a planning support system is suggested. In a defined scenario, a mobile application will support team leaders by visualizing the current worker-to-production-line allocation and by interconnecting the team leaders to coordinate personnel allocation in an efficient way. Therefore, information from several distributed information sources is prepared and presented on a mobile device. Relevant information dependent on the user's role and specific context of use is shown. Human resource allocation can be edited and optimized directly using the user interface.

This article extends previous work (see [1]) concerning the concept for a mobile application to support team leaders in adapting staff plans as a reaction to unforeseen events. Since then, the application was implemented and a first prototype tested by team leaders. The steps during this interaction design [8] process form the subject of the work presented here.

In Section II, an overview of related work is given and the distinctions from the suggested approach are drawn. Section III describes the industrial scenario that is taken as a basis for the considerations made in the following sections. The requirements analysis process including the resulting use cases is outlined in Section IV. It leads to the system and interaction design in Section V. In Section VI, the domain model that, for example, gathers and stores worker profiles and supports the

system in its task is developed. Section VII describes how the production is simulated to receive production forecasts. The results of the case study follow in Section VIII. Finally, Section IX contains a discussion of the suggested approach and gives an outlook on future work.

II. Related Work

The suggested system has strong relations to the research field of computer-supported cooperative work (CSCW) [9], also known as groupware. According the definition by Wilson [10] enabling technologies in the Teamleader App case are wireless networks, mobile devices and the merging and aggregation of relevant information. Johansen's Time-Space Matrix [11] divides CSCW systems into four categories. According to this classification, the proposed system is a different time / different place system, processing asynchronous communication. Since systems increased in number and complexity, the classification in the 2x2 matrix became more difficult. Thus, Penichet et al. [12] suggested a more flexible classification method. According to this classification, the Teamleader App is a type C-5 system: C, as it coordinates a production company's internal processes and processes information that enables interaction between team leaders; 5, as it is an asynchronous application that is intended to appear in different spaces (distributed).

Current staff plan software is implemented as a standalone solution or integrated in centralized enterprise resource planning (ERP) or manufacturing execution systems (MES). Time tracking is commonly part of the MES, whereas time management can be part of both MES and ERP [13, 199-212]. Most staff plan or workforce management systems focus on the scheduling and optimal resource allocation task. User interfaces present timetables with a view over a planned period in a very functional way. Some software providers offer mobile applications that allow access to the staff plan systems. These systems lack adaptive context- and role-specific information processing and neglect the collaboration and coordination aspects tackled in this work. MES and ERP systems include high installation, application and maintenance costs. If already implemented, huge efforts and costs are required to adapt such software to a specific use case according to its size and complexity. A problem-specific solution that processes information from such systems to support stakeholders in an efficient way is the target that is focused on in this work.

In research, the project ENgAge4Pro [14] focuses on age-appropriate staff plan. To address the ergonomics topic, physical attributes, such as body weight and height, are considered. The research project EPIK [15] focuses on the optimal allocation of resources to enhance efficiency. A mobile application was developed that supports the worker with context-specific information. The research project KapaflexCy [16] covers short-term production scheduling. A mobile application allows employment requests to be sent to workers. After receiving these requests, the workers coordinate the takeover of the employment themselves. From a hierarchical perspective, this is a bottom-up approach. Unlike the project ENgAge4Pro, the main concern of the staff planning support system is not ergonomics. In contrast to the EPIK project, the system suggested is developed to support team leaders in their planning task. Therefore, more attention is paid to the appropriate presentation of relevant information on the device. Optimized resource allocation will be included in the form of an additional function (not the object of research). Compared to the KapaflexCy project, the mobile application developed here implements the allocation of employment in a top-down way. Nevertheless, worker-related information can be used to provide feedback for workers, e.g., when a mistake during manufacturing has occured [2].

Figure 1. Manufacturing steam ovens at imperial/Miele.

III. Scenario

The production of kitchen appliances in Germany is facing several challenges. Companies require the ability to produce their products under optimum cost and flexibility due to rising variants and a competitive market. Therefore, it is necessary that the manufacturing industry makes efficient use of resources and energy in order to keep the high-cost country of Germany a competitive production location. Manufacturing of steam ovens in imperial/Miele plant floors (see Figure 1) follows the "Miele Value Creation System". Multiple U-shaped production lines for diverse product classes allow for highly flexible handling of varying production programs. Each steam oven is assembled by a single worker in a one-piece-flow setting, which entails high responsibility and a complex work content for all employees.

Once per week, the plant's foremen and team leaders plan the production on the shop floor level, assigning resources and capacities to production orders. Detailed planning is done on a daily basis, considering the production program and availability of workers. In case of unexpected staff shortage or modified production volumes at short notice, the team leaders re-assign available workers to production orders and assembly stations, and also across assembly lines. The process of staff planning is demand-oriented and flexible, and quickly becomes complex and time consuming when trying to meet the demands of multi-variant production scenarios with varying production programs, small lot sizes on multiple lines and customer-individual products. Furthermore, team leaders want to foster a broad skill set in all employees by organizing a rotating assignment of workers to varying tasks while, at the same time, the high quality standards of Miele need to be guaranteed by intense training on each particular product class.

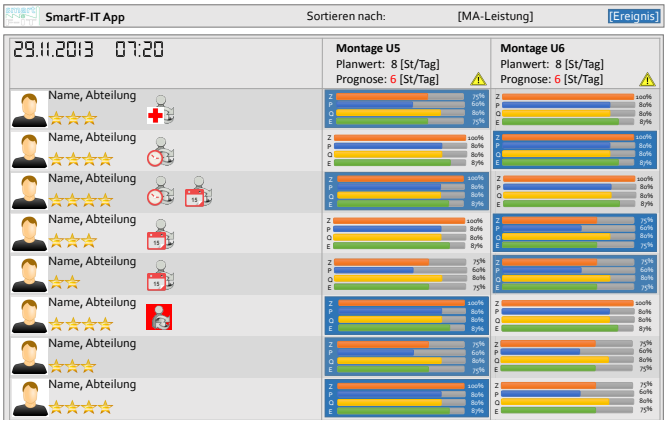

Figure 2. First Mockup of the User Interface.

A great deal of experience is needed in order to make the right decisions. Not all information that is necessary to make the right decision might be modeled and computed. Besides worker profiles that can be automatically generated, soft factors also have to be considered. Thus, an adaptive assistant system that transparently combines data from production orders, human resource management and the plant floor could significantly facilitate and speed up the daily staff planning process in order to support decision-finding.

IV. From Requirements to Use Cases

This section starts with a brief description of the process of interaction design. The most important requirements collected during this process are described and concentrated in five use cases.

A. Process

When the idea of a collaborative tool to support team leaders arose, motivated by the scenario formulated in Section III, the phase of gathering and analyzing requirements [17] started. Discussions between software engineers and experts from the problem domain resulted in a set of use cases. These use cases were formalized according to [18] and described from the user point of view, as suggested by [19] as an essential step in the software engineering process. The most important use cases are described in Section IV-C.

Afterwards, the use cases were discussed and essential data sources to enable the desired application were identified. It turned out that the integration of all these data sources is one of the most challenging parts when implementing such a system in a company. Formats to exchange data were defined and test data was generated. In parallel, first protoypes of the graphical user interface, in the form of paper and PowerPoint prototypes (see Figure 2) as suggested by [20], were designed and improved over several iterations. The resulting prototypes were discussed with the partners at imperial/Miele and the team leaders in the factory to involve users early in the design cycle [21]. The collected feedback flowed into the development of a first prototype that formed the basis for the case study, which is presented in Section VIII.

B. Requirements

In a factory hall up to five team leaders supervise multiple production lines, respectively. To tackle bottlenecks in production, team leaders can request workers from other team leaders (Req. 1). A mobile application could support this process by communicating these requests to all team leaders. Inversely, the same application can provide an overview of the current staff plan, worker profiles and workers' presence/absence (Req. 2). This view has to be exclusive and secured, as staff information is sensitive (Req. 3). The staff plan overview intends to help the team leaders in indicating workers he or she might suggest to change to another team leader when answering a worker request. During this answering process, feedback from the system about the effect of providing a certain worker to a team leader colleague would be helpful (Req. 4). The overview additionally allows team leaders to follow a worker's development, keeping him on track and taking care that he develops knowledge on manufacturing different product variants by switching in defined time intervals between product variants (Req. 5).

To enable these functional requirements, a lot of data has to be gathered and aggregated. Worker and product identification numbers have to be linked to maintain worker profiles. Worker presence/absence has to be taken into account to indicate a bottleneck just in time when it occurs. Numbers about production targets must be made available from the ERP system, allocated to team leaders and production lines. To compute the effect a change in the staff plan might have, information about production lines and product variants has to be modeled. A model that stores skills, relations, and roles is necessary to check access rights to the system (team leaders) and to store and maintain worker profiles.

C. Use Cases

The most important use cases (UC) derived from the requirements analysis are the following five:

UC1 Staff shift (Req. 1)
UC2 Staff presence (Req. 2)
UC3 Team leader authentication (Req. 3)
UC4 Production simulation (Req. 4)
UC5 Staff iteration (Req. 5)

The staff shift (UC1) is the core of the Teamleader App and closely intertwined with the production simulation use case (UC4). To request new personnel for a certain production line, the team leader presses a button. All other team leaders are notified and asked to suggest suitable workers. When a worker is suggested, the production simulation calculates the estimated number of products that will be manufactured if the suggested worker is removed from the suggesting site and added to the requesting site. To deliver fast results to the requesting team leader, the other team leaders are forced to answer within five minutes, otherwise the application is locked. If a team leader has no capacities he can send a "no" or "only in emergency cases" reply instead of suggesting a worker mandatorily. The requesting site receives worker suggestions or rejections together with calculated production values and is able to accept or refuse these suggestions. If accepted, the suggesting team leader is informed and finally requested to confirm the staff shift as his situation might have changed in

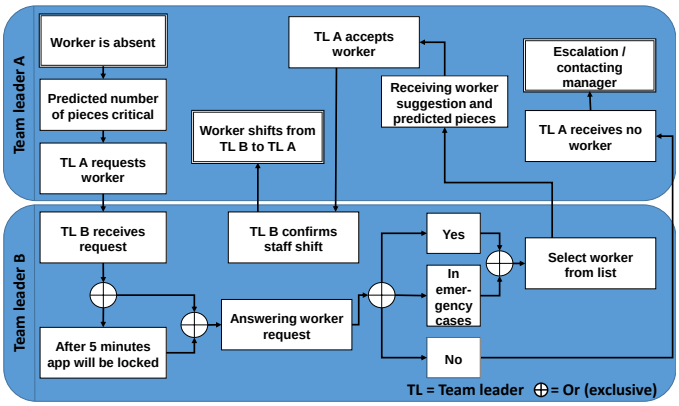

Figure 3. Staff shift use case (UC1) between two team leaders as a process.

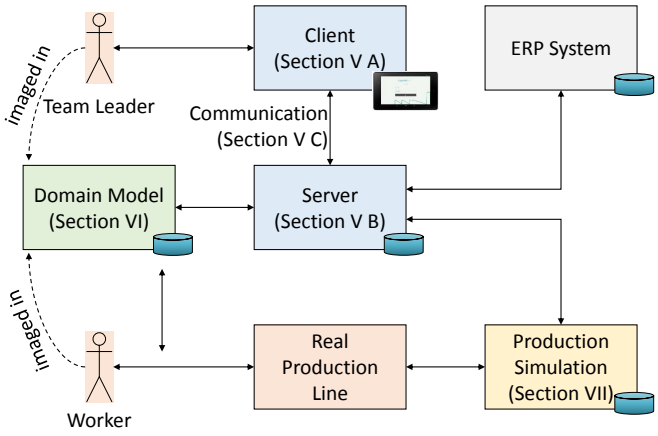

Figure 4. IT-Architecture.

Figure 5. Detailed overview of a team leaders' staff plan.

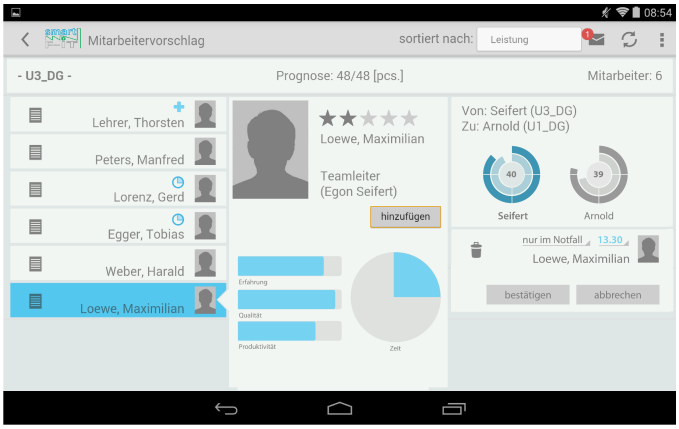

Figure 6. Answering a worker request.

the meantime. Figure 3 shows a staff shifting process between two team leaders.

The staff presence (UC2) enables team leaders to get an up-to-date overview about the status of their workers. The system displays workers that have already arrived at the production line and workers that are absent, e.g., in the case of sickness, unpunctuality, or vacation.

The authentication mechanism (UC3) provides secure access to the information in the system and leads to a view with information relevant to the current user.

A staff iteration (UC5) is desired when a worker has worked for a defined period of time manufacturing a certain product. In this manner, workers stay trained in manufacturing a certain set of products. Therefore, they switch between products in defined time intervals. The system reminds the team leader when a worker should switch to another product.

V. System Design

To tackle the challenges described in Section III and to support the use cases described in Section IV, we developed an application that supports the team leader in adjusting the daily routing when specific events occur. Figure 4 shows the coherences between all components of the suggested system that supports this kind of ad hoc planning. On the *client*-side described in Section V-A a suitable user interface for team leaders that supports all use cases was developed. On the *server*-side described in Section V-B, heterogeneous information sources are prepared and consulted in order to support reasonable decision making on the client-side. The *communication* necessary to support the Teamleader App's core feature (UC1) is described in Section V-C. The server connects to several heterogeneous information sources. An external source is the *ERP System* that delivers information on production planning. The *domain model* reflects all aspects in the application's context, such as workers, team leaders, products, and production lines. It is described in Section VI. The *production simulation* predicts how many products will be manufactured for a given staff plan. It uses its own model of a production line and is described in Section VII.

A. Client-side

1) Screens: The authentication (UC3) forms the entry point to the application. The application loads role-specific profiles for each user. As each team leader manages different manufacturing lines, the respective lines are loaded and currently allocated workers are presented on the home screen shown in Figure 5. This screen has a navigation bar on top, which lets the user switch between detail and overview, sort workers, and contains a link to the message box, a reload function and a context menu. The screen in Figure 5 shows two production

lines that are arranged in the two columns on the left. Workers that are allocated to a line have a blue-colored profile in the respective line's column. A profile shows experience, quality, productivity and time values for each worker (in the detail view aggregated on a 5-star scale). The time value, represented by a pie chart, indicates the amount of the recommended time a worker has spent manufacturing a certain product (UC5). The workers are arranged on the right. If a worker is absent (UC2) a symbol signals the reason: a cross indicates sickness, a suitcase vacation, and a clock lateness. In the left column of the home screen in Figure 5, the system shows a deviation from the planned schedule and visualizes a warning by highlighting the discrepancy from the production forecast (UC4) in red and encourages the team leader to act in an ad-hoc manner.

One common reaction to meet the planned quantities at the end of the day is the search for suitable workers. If the team leader decides to request a worker, his colleagues see the screen presented in Figure 6. A red one in the message box symbol at the top of the screen indicates an important message from a requesting team leader. If the receiving team leader switches to the message box and decides to suggest a worker, he will see the screen in Figure 6. On the left of the screen a list with all workers is shown. The middle of the screen shows the profile of the selected worker. A button allows adding an available (not absent) worker to the suggestion message. If that happens, the production is simulated under the new condition that the selected worker will not work on his current line and instead work on the line requested by the other team leader. The charts on the right of the screen show two values: the inner and outer circle represent the number of manufactured products before and after editing the staff plan respectively. The left chart provides numbers for the suggesting team leader, the right chart for the receiving team leader. This visualization is intended to support the team leader in making his decision. Different workers lead to different predictions. The list of suggested workers can be extended to contain more workers. If the suggestion is complete, the teamleader can send his response by pressing a button. When his colleague accepts his suggestion, the worker can shift between the production lines (UC1).

2) Worker profiles: The detail view shown in Figure 5 shows the qualifications of workers. It is possible to switch between detail view and overview. The latter shows a five star ranking for each worker. To compute the qualification of a worker (w) for a respective manufacturing line (l) and product (p), three parameters were defined:

$experience(w, l, p)$
: Total time spent on this line by a worker.

$quality(w, l, p)$
: Defective pieces per shift by a worker.

$productivity(w, l, p)$
: Pieces per shift by a worker.

These parameters allow the team leader a rough estimation of the worker's skills. The ranking function $rank(w, l, p)$ forms a weighted aggregation of these parameters and represents the skill level of each worker on a scale between 0 and 5. These weights are dynamic and adapted to the specific use case.

3) Realization: An eight inch tablet was identified to be most suitable for the daily deployment in a factory environment. It is small enough to fit in a team leader's pocket and offers enough space on the screen to present relevant information.

The implementation is based on a model-view-controller (MVC) framework. The separation into view and controller allows a fast integration of new user interfaces and the reuse of components. Both view and controller access the model, which holds the dynamic data that is presented in the accessing view. In a particular case there exists a bidirectional data binding mechanism. A controller is aware of changes that are made by the user in a view. The view is automatically updated if changes on the model-side occur. In the Teamleader App these bidirectional data bindings are important for the dynamic updates of the user profiles (experience, quality, and productivity visualizations), which change continuously during a working day.

Furthermore, the implementation is separated into different modules. These modules can be included or excluded according to the the specific requirements of an individual factory. This kind of flexibility allows the generation of lightweight and full-featured versions of the Teamleader App, which adapts itself in this manner to its environment and context of use. The content on the application site can be used in a multilingual environment. An integrated translation engine allows one to dynamically change the language of the Teamleader App. This feature enables the company to introduce new languages by defining translation pairs of vocabulary.

B. Server-side

The information that is necessary to support the planning process of a team leader as described in Section V-A is located at three different points that are sketched in Figure 4. In the present case, the information system can be described as a multi-computer, partitioned, distributed, shared-nothing system. Thus, a suitable strategy to integrate the information has to be selected. To preserve the autonomy of the sources, a virtual integration strategy was selected, which leaves the data at the sources. This kind of on-demand integration in a decentralized manner enables us to keep the system design easy to extend and to transfer data only when needed from solely relevant sources. A mediator-based approach [22] was chosen to realize the virtual integration system. The mediator provides an interface implemented as a Web service and communicates with the application. It is the responsibility of the mediator to provide a structural and semantic data integration. Wrappers are implemented for each information source to overcome the heterogeneity on the data level and to enable the data flow between mediator and sources.

Focusing on the data sources, the ERP System is the only existing system that is already available in a common factory. An ERP system in the considered scenario provides—in collaboration or separately—access to time tracking and production planning data. The domain model encodes the workers' skill matrix and working history. It is described in detail in Section VI-B. The production simulation described in Section VI-A contains a model of production lines containing information about process steps and involved manufacturing

equipment and allows estimations about quantities that can be achieved when a specific worker is rescheduled on that line. In combination with processing times, it could also be used to calculate optimal resource allocations with algorithms from the operations research field. The factory worker model constitutes new input data that allows new forms of optimization based on the user model and corresponding profiles.

The server was implemented using the inversion of control design concept and dependency injection. This technique allows replacing parts of the server without writing any gluing code lines through reflection. An extension or adaption of functionalities benefits strongly from this approach. The server's authentication process maps every client instance request to the client itself. This approach ensures that only authorized clients can access a specific data slice.

C. Client-Server Communication

The actual process of transferring workers between team leaders involves several steps and states. Figure 7 shows the high-level connection between a team leader *TL* 1 and his colleagues *TL* 2 to *TL* $n-1$. We have chosen a client-server architecture over a peer-to-peer one for two main reasons. First, a lot of different aggregation procedures (like worker-profile based information) require information from a central big-data factory storage that is influenced by many machines and sensors inside the factory. Second, communication to another team leader might be impossible for several seconds due to an insufficient network connection or a crash of a client device. In such situations a communication proxy (like a server) can handle those special cases by realizing a separation of concerns.

Figure 8 shows the possible internal states and transitions of a single worker transfer. The solid black lines, on the one hand, describe possible actions that can be performed by the team leader who initiated the request. The dotted lines, on the other hand, indicate actions that can be triggered by all other team leaders *TL* 2 to *TL* $n-1$.

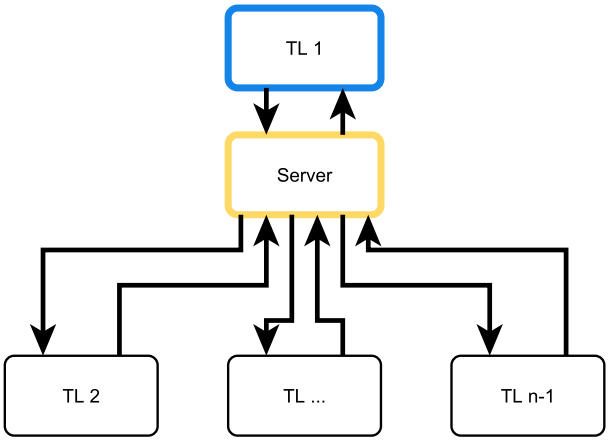

Figure 7. High-level team leader (TL) and server connection

As soon as *TL* 1 initiates a worker transfer by requesting new workers in the client application, the request is sent to the server and the actual transfer process starts. The initialization on the server side causes the newly created request to switch its state from *Idle* to *Requested* (see Figure 8). Information about the request (the production line, the desired skills, etc.) is now broadcasted to the other team leaders. Each team leader can now decide on his or her own to offer possible workers for a transfer or to decline the request. Note that it is not possible to offer the same worker twice, as long as the request, containing the worker is not in the *Closed* state.

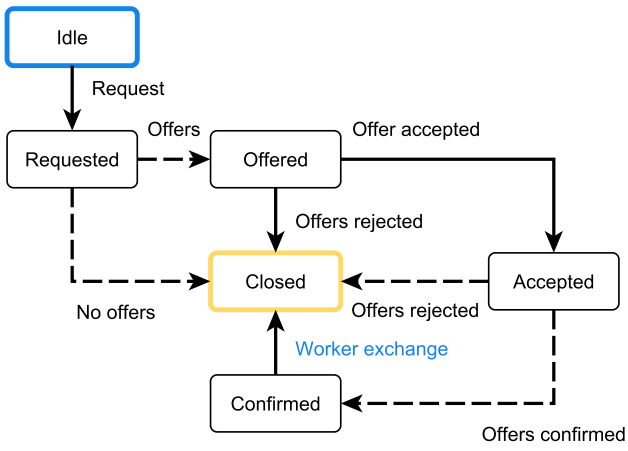

Figure 8. States of a single worker transfer

If no offered workers can be received, the global request will switch to the *Closed* state and *TL* 1 will be notified that no workers are currently available for a transfer. If at least one other team leader offers some workers, the state will switch to *Offered*. In this case, *TL* 1 can review every received offer and can decide whether to accept the proposed workers or not. Similar to the previous steps, a rejection of all proposals switches the whole request to *Closed*. Accepting at least one proposal will trigger a state change to *Accepted*.

Once an offer is accepted, the responsible team leader is notified and can review the proposed workers once again. This additional review phase allows the team leader to reevaluate his or her current situation, which could have changed since the workers were offered. If the transfer is approved, the overall state will be set to *Confirmed* and the proposed workers will be transfered to *TL* 1, since the global request was already successful. Other available offers, which are not confirmed yet, can still be confirmed or rejected afterwards. However, as soon as all responses from all team leaders are available, the global request will switch to *Closed* and the request can be removed from the server.

A frequent exchange of information between clients requires a stable, fast, and reliable transfer of data. To tackle this key issue, a channel-based communication protocol from Section V-C is used. Therefore, incoming and outgoing channels were implemented to support the staff shift procedure. The outgoing channels are divided into broadcast and individual message channels. A client instance subscribes to a channel and will be notified by the server in a given situation. This kind of communication was realized using the WebSocket protocol [23] that provides full-duplex communication channels. The internal information-exchange format uses JSON as intermediate data representation.

VI. Domain Model and Database

In order to directly link the Teamleader App to the actual production processes, namely the concrete current situation in the factory, we integrated a domain model as part of the application's backend. This model consists of a graph, representing a semantic network that flexibly interconnects information from different areas of the plant floor, on different layers of granularity. The network combines a topology of manufactured products with a model of the factory floor and the staff profiles, and is used as a shared domain model by different SmartF-IT applications (such as [1] and [2]). Apart from the product hierarchy, it represents the structure of the industrial facilities, consisting of production lines and their work stations. A third type of data contained in the model is the employee model, providing the application with detailed structural and individual information on the factory staff. In order to make these representations of products, lines and employees a realtime model of the plant floor, several pieces of up-to-date information have to be fed into the domain model. Team leaders and workers are connected to their respective responsibilities and work places. Planning figures, taken from the ERP system, link manufacturing lines to the product hierarchy, enabling the Teamleader App to check on production targets and trigger the simulation of goal achievement w.r.t. the respective staff configuration. Employees are interconnected via hierarchical relationships that depict their areas of responsibility.

A. Product Hierarchy and Facilities

Our approach of representing the domain as a semi-structured data network (cf. Section VI-C) allows for flexible modeling of products and plant floor facilities, customized for the needs of individual enterprises. Depending on the enterprise's requirements of production planning w.r.t. flexibility, minuteness, lot size and breadth of product range, the product hierarchy can be specified to an arbitrary level of detail, starting from product groups or categories (e.g., steamer, cooking chamber), ranging over devices (steamer, combi-steamer) and device types down to a fine-grained distinction between device variants, e.g., country- and market-specific versions of products.

The granularity of modeling the plant floor facilities depends on a factory's individual philosophy and implementation of the production process, and of course on the respective product's inherent properties. If the production consists mainly of one-piece-flow processes with most of the assembly taking place on the same spot, and with the help of a material shuttle, the structural facility model might consist merely of floor areas or work places. In cases of conveyer belt production with clocked processes, or when a more fine-grained modeling of responsibilities and production planning is required, we need to represent individual production lines, work stations, and maybe even process steps (the single operations or work steps, executed automatically or by employees), which are in turn connected to the product hierarchy, namely, to those product variants that require these steps as part of their production schedule.

B. Model of a Factory Worker

The information about workers' skills and experience needed by the Teamleader App in order to generate useful recommendations is taken from the part of the domain model that represents the factory staff. This employee model represents semantic relationships between the horizontal and vertical roles of an employee in the factory (i.e., his tasks, work content, and his position within the staff hierarchy) on the one hand, and his skills (e.g., work experience) and individual requirements on the other hand. Examples for the latter are an employee's handedness, language skills, allergies to specific materials, or an inability to lift heavy weights or to distinguish colors. A small excerpt of the worker model is visualized in Figure 9. The qualification of workers is encoded within the *History* nodes of a semantic network between products, assembly lines, and employees. The application can, for instance, query the model for workers that are experienced assemblers of product p at assembly line l, and rank them using weighted aggregation as described in Section V-A. The depicted subgraph shows how information about a specific worker, *Williams*, is encoded in the model. *Williams* himself is an assembly worker (shown by the *hasRole* relationship), whose native language is English and who currently works at work slot *U6_S05*, which belongs to the slot group *U6*. *Williams* is associated to the product hierarchy indirectly: his work place links via the *produces* relation to the product group *DGC* (combi steam oven), that is currently being produced at *U6*. In order to simulate production processes, the Teamleader App will query, e.g., *Williams*'s working experience w.r.t. the production of *DGC* (via the *hasExperience* relationship).

Of course, such working experience has to be fed into the model before it can be provided to applications such as the Teamleader App. An automatic logging of production data is most convenient for realtime model updates, and many modern factories already implement the foundations of such logging in the form of sensors that are part of the production lines and that monitor, e.g., the execution of single process steps. In enterprises that have no automated logging of process steps, the staff's working experience can be updated, e.g., on a daily basis, using the production data from the preceding workday. Similar to the degrees of freedom in modeling product hierarchies and facilities (cf. Section VI-A), the recording of data can vary in its level of granularity. Simple counting of assembled pieces can be refined to an informed logging about the number of correctly vs. incorrectly assembled devices, with results from the test rig incorporated in the data recording, or even to logging of individual work steps.

Data logging is partially decoupled from an analysis of the recorded data, because evaluation can aggregate over various levels of logging. The assessment of a worker's qualification is done on aggregates of the original logs, e.g., by summing over the number of devices of a specific type, assembled by this worker within a certain interval, or by computing the ratio of correctly executed work steps across devices. This is an important factor in a scenario where not all desired use cases of evaluation, i.e., modes of aggregation, are known at the time of database modeling, or even at the time of data recording. Different applications will have different interests w.r.t. the modeled and recorded data, and therefore compute individual aggregation functions on the graph.

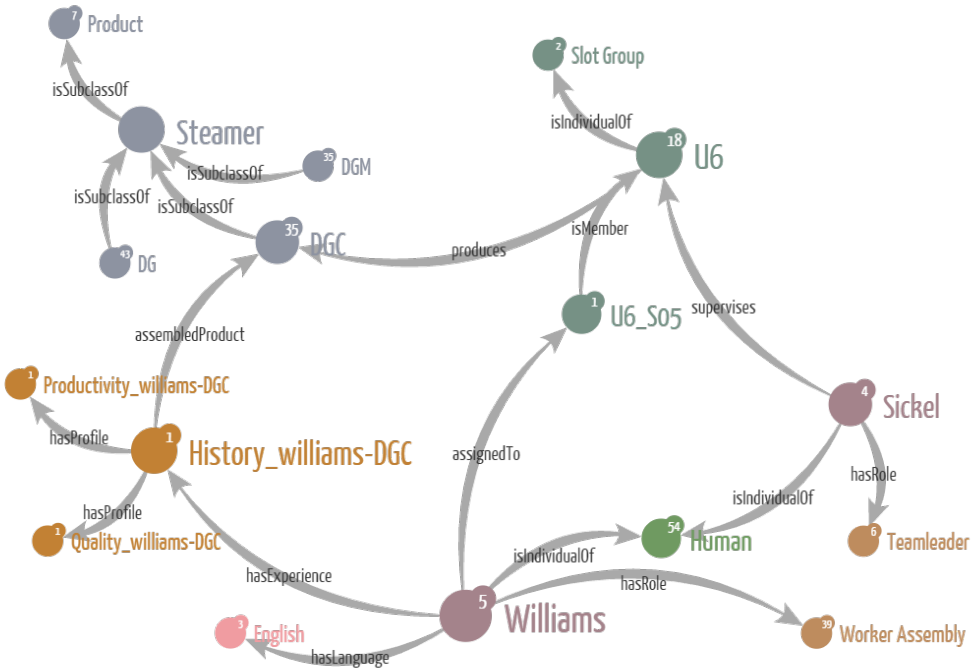

Figure 9. Excerpt of the domain model. Node colors denote the semantic groups of information encoded in the graph (blue: product hierarchy, dark green: plant floor facilities, orange: worker experience as part of the employee model).

In the imperial/Miele scenario, the Teamleader App presents worker experience on the level of product types or variants, meaning that a relatively coarse data logging in the form of daily updated production numbers is sufficient. However, in order to ensure a fair and realistic assessment of worker qualification, we plan to realize a feedback mechanism using signals from the repair stations, in order to integrate the knowledge about whether a device that failed testing was incorrectly manufactured due to a mistake made by the worker, or rather due to other factors, e.g., material deficiency or tool abrasion. This goal entails another prerequisite, namely that each single device that was produced relates back to the workers involved in its production process. In modern factories, this might be realized by barcode stickers or RFID chips attached to the product, possibly even in the form of a digital product memory [24].

C. Implementation of the Domain Model as Database Backend

Our domain model is implemented using the open-source, Java-based graph database Neo4j [25]. Querying of the model from within the application is realized by accessing Neo4j's RESTful interface. The model's contents are derived from a domain ontology (built using the Web Ontology Language OWL, [26]) by automatically mapping the ontology's concepts and relations into the Neo4j graph database format (T-Box). Once created, the graph can be populated and updated (A-Box) at runtime with dynamically-changing data like a worker's history, retrieved by logging of assembly operations.

There are several reasons for choosing a graph database over conventional storage formats like, e.g., SQL databases. Regarding performance for path operations in highly-connected data such as our domain model, relational databases quickly become overburdened by queries of increasing complexity due to joins and index lookups; whereas in graph databases, which use index-free adjacency in traversing from node to node, query latency is relatively independent of the database size and the number of connections (see [27], chapter 2). Note that denormalization for relational databases is not an alternative here, since the data model is not tailored exclusively to the needs of the staff planning app, but is instead meant to provide a flexible, multi-purpose source of semantic information, with diverse applications reading from and writing to the model. This implies that a) we cannot anticipate what relations will be queried most frequently at runtime, and b) we need to prepare for easy model update (e.g., based on sensor data), in order to keep the graph a realtime model of the staff data. Consequently, there is no use in optimizing read access for specific relations at the cost of slower write access.

The most crucial benefit of graph databases in the context of Industry 4.0 is that they allow for an explicit, intuitive, and easily-expandable modeling of the complex semantic dependencies that exist in modern factories, and that the semantics of such graphs can easily be understood even by users unfamiliar with conventional modeling languages like the unified modeling language (UML).

VII. Production Simulation

When scheduling workers for different assembly lines, a team leader is supplied with various information and key figures via the APP. Part of this information are the expected quantity forecasts considering planned workers, production program and assembly times of individual product variants. For this forecast, a simulation is run in the background of the Teamleader App, in which a defined part of the value stream of the steam oven assembly is mapped. The dynamic simulation model, which is prepared with the software Plant Simulation [28], is connected with the Teamleader App via database and is

started by a hypertext transfer protocol (HTTP). The data exchange between the simulation model and the database occurs via an ODBC module. In the database, the Teamleader App deposits information such as worker assignment to the lines, their qualifications and production program as parameters for the model initialization.

If the HTTP is called in the Teamleader App, the simulation model will be started. This happens during the rescheduling of the worker to the assembly lines. Thereby, among other things, the new worker allocation is deposited in the database and the model is filled with these parameters. Afterwards, the simulation occurs, taking nearly three seconds for the simulation of an entire shift (7.5 hours). At the end of the simulation, the results of the simulation are stored in a database. The Teamleader App recognizes the new result, reads it, and visualizes it in the front end.

The value stream includes the individual areas of assembly and test stands at the end of the assembly line. In the factory, two organizational forms of assembly occur, which are mapped in the simulation model. On the one hand, steam ovens are assembled in one-piece-flow, on the other hand, fixed position assembly is used at another assembly area (Figure 10 shows an extract of the simulation model). During one-piece-flow, the worker moves with the product to be assembled along different assembly stations. At these stations, the material is provided and the assembly process steps can be performed. Whereas in the areas of the fixed position assembly, the whole assembly process is executed at one station except for the functional test.

The one-piece-flow lines are displayed in the simulation model as assembly rows and consist of five stations and workplaces, which are spatially (edges) and temporally (intermittent, asynchronous) interlinked. In fixed position assembly, separate stations and workplaces are modeled. In both cases, the assembly happens according to the object principle. Figure 10 represents an extract of each of the two assembly areas of the simulation. Each assembly station consists of a station and the workplaces for the worker. In the case of one-piece-flow the worker will move from station to station, following the product until it is fully assembled. For each station the process times (depending on product variants) are stored in a table. A source will provide the assembly station or assembly lines with various products (orders), which are listed in the production program table.

As in reality, the test stands are positioned at the end of the assembly lines, whereas an assembly line is followed by one test stand. The test stands can be fed with complete assembled products by every line. Depending on the inspection scope, assembled products are outsourced to adjacent test stands, which is also considered in the model. Since long distances need to be walked to faraway test stands, which has an impact on the produced quantity, the workers are sent with their products to the adjacent test stands.

The qualification of the worker also has an impact on the quantity. In the model this aspect is considered, due to the factors assembly area, one-piece-flow, fixed position assembly and inspection. This is also reflected by the deposited worker model.

In the first version of the simulation model, no optimization was performed regarding the worker scheduling. The model

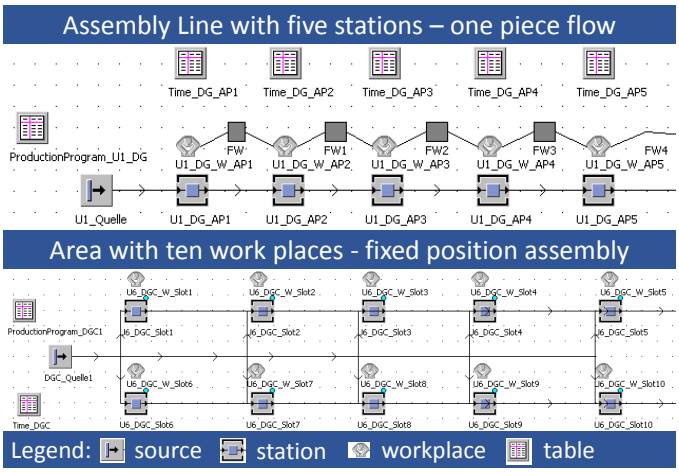

Figure 10. Model extractions representing one-piece-flow and fixed position assembly.

shows possible consequences of a team leader's decision and supports him regarding his selection. The team leader has the final say because many soft framework conditions need to be considered, such as the education of new workers or the performance of a team, which are hard to simulate.

VIII. Case Study

To receive further insights into the requirements and needs of team leaders, we decided to conduct a user study with the additional goal in mind to evaluate the usability of the current prototype (a typical step in the user-centered design process [29]). Specifically, we wanted to confirm the following hypotheses:

H1 Team leaders will in general be able to use the current prototype without instructions.

H2 Team leaders will appreciate the functions offered and can imagine using them in their daily work.

We also expected the team leaders to provide us with more ideas for functions that could ease their work further.

A. Method

The study was conducted during the working hours at the work place of the team leaders. We conducted a single session with each participant and closed the study with a group discussion session in which their supervisor, who had also a general notion of how the app works, was also present. Every session took approximately 30 minutes. The mobile application was shown on an 8-inch tablet.

In respect to the goals of the user study and the hypotheses to test, we decided to let the team leaders explore the application on their own without any explanation beforehand. The participants were encouraged to think aloud while using the system, as suggested by [30] and [31], to reveal positive and negative design decisions and observe their interactions. To ensure that every team leader would be exposed to all implemented features, we guided them to certain functions, if observation indicated that they would fail to find them otherwise. As stated before, some of the team leaders were

involved in the earlier requirements analysis and thus might have remembered which functions the system should offer in principle, but no team leader saw the current prototype state in advance. Prior to any interaction, the team leaders answered a pre-session questionnaire collecting general information (e.g., how much experience they have with tablets) and questions about problems in their daily work that could be overcome with technical aids. After the exploratory interaction phase a System Usability Scale (SUS) [32] was provided as well as a post-session questionnaire with quantitative questions (e.g., "I think I can ease my daily work with the team leader app") to be answered on 5-point Likert scales and qualitative questions (e.g., "Which of the features seen needs improvement?").

The group discussion took place three hours after the first single session and focused mainly on the impression the participants had of the application. We thus followed an unstructured interview technique.

B. Results

The study was conducted with five team leaders. Even though the number of participants looks small, we refer the reader to the work of Nielsen [33] in which three to five participants were reported to be sufficient to find nearly all usability problems in a system. The average age of the participants was 36 years (SD=6.78, Mdn=36) and they all reported having intermediate experience (M=3, SD=0, Mdn=3) with computers. Each of them owns a smartphone (reported experience level: M=3.6, SD=0.55, Mdn=4) while only two own a tablet (reported experience level: M=1.8, SD=0.84, Mdn=2). In general, the team leaders reported to be open to technical innovations (M=3.8, SD=0.45, Mdn=4).

1) Potential Technical Improvements: The team leaders are convinced that technical aids can support them in their daily work (M=4.2, SD=0.84, Mdn=4). We asked them to state up to three aspects (without providing answering options) that could be supported and how they could imagine being supported by a system. Interestingly, three of the five participants stated that currently communication seems to be problematic and assistance applications for this might be helpful. Twice an improved access to specific data (e.g., which malfunctions are currently active, or a direct overview of how many products were created on a given day and tested without error) was mentioned. One team leader stated that better mobility of mobile applications would also be beneficial, as access to specific subsystems is currently only available from stationary work stations.

2) Perception of Features and App in General: Concerning the quantitative and qualitative questions in the post-session questionnaire, we learned that the team leaders think that the team leader app is a reasonable innovation (M=4.6, SD=0.55, Mdn=5) and that it eases their work (M=4.8, SD=0.45, Mdn=5). This serves as evidence for **H2**. We asked them (again, without providing any options to select from), which features were perceived as most useful. Table I shows an overview of them. Concerning the results reported so far, the mentioned features are easily explainable. The capability to exchange an employee and the option to receive an overview of workers at assembly lines together with system-derived values helps to make expert knowledge (which every team leader has for his own assembly line) available to others and thus eases the communication among the team leaders. The prediction system, on the other hand, also seems useful as it provides a simple-to-understand number to find out what consequences the exchange will have. This again serves as a formalization of expert knowledge. These aspects were also confirmed in the group discussion.

TABLE I. OVERVIEW OF PERCEIVED MOST HELPFUL FEATURES

Feature	Times mentioned
Exchange of an employee	4
View of worker attributes (experience, quality, productivity)	3
Prediction system	3
Overview of workers at an assembly line	3

3) Usability: The observation of the participants has shown that the participants in general were able to interact with the application without instructions. The core problem observed though was that they were not so sure which areas were clickable (and thus lead to further information) without trying it. Even though this testing helped the team leaders to explore the functionality of the app, we are reluctant to accept this as evidence for **H2** and will test this specifically in the next iteration of our user-centered design cycle. For this, we will also fix the usability problems revealed in this study (by observation and discussion, cf. Table II). The System Usability Score (SUS) [32] supports this further as an average score of 62 (min=50, max=70) indicates issues (as the maximal achievable score is 100). Following the work of Lewis and Sauro [34], we can distinguish between the usability part and the learnability part of the SUS; we still see these issues (usability score of 60.63) but the learnability is slightly better (67.5). The quantitative questions concerning the usability indicated that team leaders think that the usage of an 8 inch tablet is not the worst possibility (M=1.4, SD=0.89, M=1), but also not the best option (M=3.9, SD=1.1, Mdn=3). Also when actively asked, the navigation was perceived as improvable (M=3.6, SD=1.14, Mdn=4) and the graphics seem not to be completely understandable (M=2.8, SD=1.48, Mdn=3). The text parts of the app seamed to be better understandable (M=4.2 SD=1.1, Mdn=5) and the selection of color could be improved (M=3.2, SD=1.3, Mdn=3).

TABLE II. OVERVIEW OF FOUND USABILITY PROBLEMS

Usability issues
• Unclear navigation in terms of which areas are clickable and lead to further information
• The association between a worker and his corresponding line (when a team leader supervises more than one assembly line) is unclear
• Buttons and font sizes are sometimes too small
• The assembly line overview is not completely clear at a first glance (prediction of assembly line outcome was interpreted as prediction of worker performance, which is given only indirectly)
• More explanations on the way the application works are needed (e.g., how stars are derived)
• The exchange of an employee view was not intuitive enough. The different statistics were unclear and the team leaders had the feeling that to make a good decision they need further information that is not yet accessible directly in this view. Additionally, more information on the exchange status should be integrated, as the team leaders were not sure what happens after they have sent a request.
• The arrival of in-app e-mails should be made more obvious.

4) Requested Features: Several features could be elicited that could potentially ease the work of team leaders and

would in consequence further improve the perception of the Teamleader App.

- It should also be possible to rate specific workers, to establish a second measure besides the system-derived measurement. In consequence, the team leaders also want to make notes on the different workers.
- More information about the workers should be accessible (e.g., how many products they have built).
- The application should offer the option to exchange workers only for a specific amount of time.
- The application should directly integrate other different sub-systems (e.g., ERP systems).
- There should be more options in the exchange; i.e., it should be possible to directly request specific workers (because experienced team leaders have certain knowledge about many workers) or request workers that fulfill other specific attributes (e.g., can drive a specific vehicle).
- Team leaders want to see the state of assembly lines of other team leaders. Here, they do not want to see every detail, but rather an overview of the line.

As these features focus on making expert knowledge digitally available, ease communication further and improve mobility, it seems reasonable to implement them in the next iteration.

C. Discussion

It has to be noted that the team leaders currently only investigated the app exploratorily, and a study in which we observe how they really use the app in their daily work will provide a clearer picture and will be the next step. Nevertheless, the results taken from this study are valuable and important in the user-centered design process. They revealed certain usability issues that need to be targeted first, before an unsupervised study can provide reliable results. On the other hand, we learned that the integrated features are perceived as valuable and that the team leaders believe that these can ease their work. The implementation of requested features will improve the mobile application and increase the chance that it will become a helpful tool in everyday work. A further analysis of the corresponding old and new interactions with the Teamleader App will lead to deeper scientific investigations.

IX. Conclusion and Future Work

In this work, a mobile application for team leaders, to support them in making ad-hoc planning decisions, was proposed. Therefore, the interaction design process started with a requirements analysis, which led to five use cases. Addressing these use cases, a system was developed that follows a classical client-server architecture. The server provides access to data stored in a graph data base that contains the domain model of the suggested solution. Further information is fetched from existing systems, such as ERP systems and a production simulation component that computes predictions about the daily production for a given staff plan. The conducted case study showed that team leaders see a benefit in the current prototype. It also showed that there is still room for improvement to make the application usable without explanation. A set of additional features was identified which could significantly improve the use of the Teamleader App in everyday work.

In future work, it is planned to improve the interaction concept of the mobile application. The usability issues identified in the case study form the entry point for this work. Further on, a set of new features that were suggested by team leaders will be included. After these improvements are realized, a discussion with the team leaders will follow to check if the improvements match their intention. Then, an evaluation is planned where team leaders use the application over a period of time during their work. It will show the benefit of the application in the real production context. In parallel, it is planned to evaluate the quality of production simulation. Historic data has been recorded that will be used to measure the correctness of values simulated by the system.

Acknowledgment

This research was funded in part by the German Federal Ministry of Education and Research (BMBF) under grant number 01IS13015 (project SmartF-IT). The responsibility for this publication lies with the authors.

References

[1] S. Knoch, M. Reiplinger, and R. Vierfuß, "Mobile Staff Planning Support for Team Leaders in an Industrial Production Scenario," in *Proceedings of the Eighth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2014), Rome, Italy*. International Academy, Research and Industry Association (IARIA), 2014, pp. 44–47.

[2] P. Lessel, M. Müller, and A. Krüger, "Towards a Novel Issue Tracking System for "Industry 4.0" Environments," in *CHI '15 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '15. New York, NY, USA: ACM, 2015.

[3] V. Pavlov, S. Knoch, and M. Deru, "CEPBoard Collaborative Electronic Performance Board and Editor for Production Environments in Industry 4.0," in *Proceedings of the International Symposium on Pervasive Displays. International Symposium on Pervasive Displays (PerDis-15), June 10-12, Saarbrcken, Germany*. ACM, 2015.

[4] H. Kagermann, W. Wahlster, and J. Helbig, Eds., *Securing the Future of German Manufacturing Industry: Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0, Final Report of the Industrie 4.0 Working Group*. Berlin: Forschungsunion im Stifterverband für die Deutsche Wirtschaft e.V., Apr. 2013. [Online]. Available: http://forschungsunion.de/pdf/industrie_4_0_final_report.pdf

[5] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & Information Systems Engineering*, vol. 6, no. 4, pp. 239–242, Aug. 2014.

[6] F. Mattern and C. Floerkemeier, "From the Internet of Computers to the Internet of Things," in *From Active Data Management to Event-Based Systems and More*, ser. Lecture Notes in Computer Science, K. Sachs, I. Petrov, and P. Guerrero, Eds. Berlin: Springer, 2010, vol. 6462, pp. 242–259.

[7] W. Wahlster, "Semantic Technologies for Mass Customization," in *Towards the Internet of Services: The THESEUS Research Program*, W. Wahlster, H.-J. Grallert, S. Wess, H. Friedrich, and T. Widenka, Eds. Berlin: Springer, 2014, pp. 3–13.

[8] J. Preece, Y. Rogers, and H. Sharp, *Interaction Design: Beyond Human-Computer Interaction*, 4th ed. Hoboken, NJ: Wiley, 2015.

[9] J. Grudin, "Computer-supported Cooperative Work: History and Focus," *Computer*, vol. 27, no. 5, pp. 19–26, May 1994.

[10] P. Wilson, G. B. T. C. Computer, and T. A. A. C. Branch, *Computer Supported Cooperative Work: An Introduction*. Springer, 1991. [Online]. Available: http://books.google.de/books?id=-EMqvaITnE0C

[11] R. Johansen, *GroupWare: Computer Support for Business Teams*. New York, NY, USA: The Free Press, 1988.

[12] V. M. R. Penichet, I. Marin, J. A. Gallud, M. D. Lozano, and R. Tesoriero, "A Classification Method for CSCW Systems," *Electron. Notes Theor. Comput. Sci.*, vol. 168, pp. 237–247, Feb. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.entcs.2006.12.007

[13] J. Kletti, *Manufacturing Execution Systems (MES)*. Berlin; London: Springer, 2007.

[14] RWTH Aachen University IAW, "ENgAge4Pro Project," 2014, URL: http://www.iaw.rwth-aachen.de/index.php?article_id=20&projid=89&proalias=ENgAge4Pro [accessed: 2014-04-10].

[15] Fraunhofer IAO, "EPIK Project," 2014, URL: http://www.epik-projekt.de/ [accessed: 2014-04-10].

[16] ——, "KapaflexCy Project," 2014, URL: http://www.kapaflexcy.de/ [accessed: 2014-04-10].

[17] E. Hull, K. Jackson, and J. Dick, Eds., *Requirements Engineering*. London: Springer, 2011.

[18] A. Cockburn, *Writing Effective Use Cases*. Boston, MA: Addison-Wesley, 2001.

[19] R. Pressman, *Software Engineering: A Practitioner's Approach*, 6th ed. McGraw-Hill, 2005.

[20] A. Spaulding and J. S. Weber, "Usability Engineering Methods for Interactive Intelligent Systems," *AI Magazine*, vol. 30, no. 3, pp. 41–47, 2009.

[21] C. Abras, D. Maloney-Krichmar, and J. Preece, "User-Centered Design," in *Berkshire Encyclopedia of Human-Computer Interaction, Volume 2*, W. S. Brainbridge, Ed. Great Barrington, MA: Berkshire, 2004, pp. 763–768.

[22] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[23] I. Fette and A. Melnikov, "The WebSocket Protocol," http://tools.ietf.org/html/rfc6455, 2011, [Accessed 27-February-2015].

[24] J. Haupert, "DOMeMan: A Framework for Representation, Management, and Utilization of Digital Object Memories," in *9th International Conference on Intelligent Environments (IE) 2013. International Conference on Intelligent Environments (IE-13), 9th, July 18-19, Athen, Greece*, J. C. Augusto, V. Bourdakis, D. Braga, S. Egerton, K. Fujinami, G. Hunter, F. Kawsar, A. Lotfi, D. Preuveneers, A. W. bin Abdul Rahman, and V. Zamudio, Eds. IEEE, 7 2013, pp. 84–91.

[25] Neo Technology, Inc., "Neo4j," http://www.neo4j.org/, [Accessed 23-June-2014].

[26] D. L. McGuinness and F. van Harmelen, "OWL Web Ontology Language Overview," W3C, Tech. Rep., 2004. [Online]. Available: http://www.w3.org/TR/owl-features/

[27] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases*. Beijing: O'Reilly, 2013. [Online]. Available: http://my.safaribooksonline.com/9781449356262

[28] Siemens Product Lifecycle Management Software Inc., "Plant Simulation," 2015, URL: http://www.plm.automation.siemens.com/en_us/products/tecnomatix/manufacturing-simulation/material-flow/plant-simulation.shtml [accessed: 2015-02-24].

[29] C. Abras, D. Maloney-Krichmar, and J. Preece, "User-Centered Design," in *Encyclopedia of Human-Computer Interaction*, W. S. Bainbridge, Ed. Berkshire Publishing Group, 2004, pp. 763–767.

[30] C. Lewis and J. Rieman, "Task-Centered User Interface Design: A Practical Introduction," Online, 1994, http://hcibib.org/tcuid, last accessed on March 1, 2015.

[31] S. Oh and B. M. Wildemuth, "Think-aloud Protocols," in *Applications of Social Research Methods to Questions in Information and Library Science*, B. M. Wildemuth, Ed. Libraries Unlimited, 2009, ch. 19, pp. 178–188.

[32] J. Brooke, "SUS: A Quick and Dirty Usability Scale," in *Usability Evaluation in Industry*, P. W. Jordan, B. Weerdmeester, A. Thomas, and I. L. Mclelland, Eds. Taylor and Francis, 1996, pp. 189–194.

[33] J. Nielsen and R. Molich, "Heuristic Evaluation of User Interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '90. New York, NY, USA: ACM, 1990, pp. 249–256. [Online]. Available: http://doi.acm.org/10.1145/97243.97281

[34] J. R. Lewis and J. Sauro, "The Factor Structure of the System Usability Scale," in *Proceedings of the 1st International Conference on Human Centered Design: Held as Part of HCI International 2009*, ser. HCD '09, San Diego, USA, 2009, pp. 94–103. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02806-9_12