The Social Factory: Connecting People, Machines and Data in Manufacturing for Context-Aware Exception Escalation

Laura Kassner, Pascal Hirmer, Matthias Wieland, Frank Steimle, Jan Königsberger, Bernhard Mitschang
Institute for Parallel and Distributed Systems, University of Stuttgart, Stuttgart, Germany
Email: firstname.lastname@ipvs.uni-stuttgart.de

***Abstract*—Manufacturing environments are socio-technical systems where people have to interact with machines to achieve a common goal. The goal of the fourth industrial revolution is to improve their flexibility for mass customization and rapidly changing production conditions. As a contribution towards this goal, we introduce the Social Factory: a social network with a powerful analytics backend to improve the connection between the persons working in the production environment, the manufacturing machines, and the data that is created in the process. We represent machines, people and chatbots for information provisioning as abstract users in the social network. We enable natural language based communication between them and provide a rich knowledge base and automated problem solution suggestions. Access to complex production environments thus becomes intuitive, cooperation among users improves and problems are resolved more easily.**

1. Introduction

The application of new technologies to the manufacturing world, such as the Internet of Things, Cloud Computing and Big Data analytics, has led to a development known internationally as either the Industrial Internet or the fourth industrial revolution [1]. The vision is a smart, connected and analytics- or data-driven factory [2] which combines a high degree of automatization with enormous flexibility on the basis of data-derived insights. Unlike the 1990s vision of Computer-integrated Manufacturing (CIM) [3], the smart factories of the fourth industrial revolution do not exclude the human worker. On the contrary, human workers are more important than ever because they remain the best at making quick decisions and figuring out problem solutions based on possibly incomplete or conflicting data. One important goal of the fourth industrial revolution is therefore the integration of human workers into a very complex and data-rich environment [2].

The function of workers in the smart factory transcends the tasks of a factory worker today: Instead of repetitive manual labor, which can more easily be automated, humans in the factories of the future will move into the role of flexible task-switching overseers with decision power for problem solving. This requires access to the right information at the right time and place.

In this paper, we explore the use of Social Media tools for connecting human workers to data sources and analytics results on the shop floor of a smart factory. Social Media has become ubiquitous in the last decade but is rarely used in a professional context today. In particular, no manufacturing-specific approach for integrating Social Media into the shop floor workspace currently exists. However, we believe that especially the manufacturing environment can benefit from the integration of Social Media as a means of communication and information distribution, in particular as the first generation of digital natives is entering the workforce. We present the complex environment of manufacturing in a medium which is intuitively accessible, supports natural human communication and is fun to use. This will help human workers navigate their tasks, improve decision-making and make factory workplaces more attractive.

This paper makes several important contributions towards these goals: We present the *Social Factory*, an analytics-backed social network for communication and error escalation in a manufacturing context, both as a concept and as an implemented prototype. This social network includes the functionality for text-based natural language interaction with both humans and machines, which the system equally represents as co-workers, backed by text analytics, for instance for starting, stopping and supervising machines. It also integrates sensor data from Internet of Things components. Exceptions in the manufacturing process are automatically recognized on the basis of sensor data and on the basis of written natural language statements within the social network's communication channels. Support for resolving the exception is automatically given via the social network on the basis of analytics on historical exception and documentation data.

The remainder of this paper is structured as follows: In Section 2, we present background work for the development of the *Social Factory*. In Section 3, we develop the concept of the *Social Factory* and define requirements. Section 4 gives an overview of the *Social Factory* architecture and a detailed discussion of its components and implementation. In Section 5, we describe an application scenario and how it is handled by our prototype. In Section 6, we evaluate our concept and implementation and conclude with Section 7.

2. Foundation and Related Work

In this section, we first present previous work which forms the foundation for the *Social Factory* as described in

URI: http://hdl.handle.net/10125/41355
ISBN: 978-0-9981331-0-2

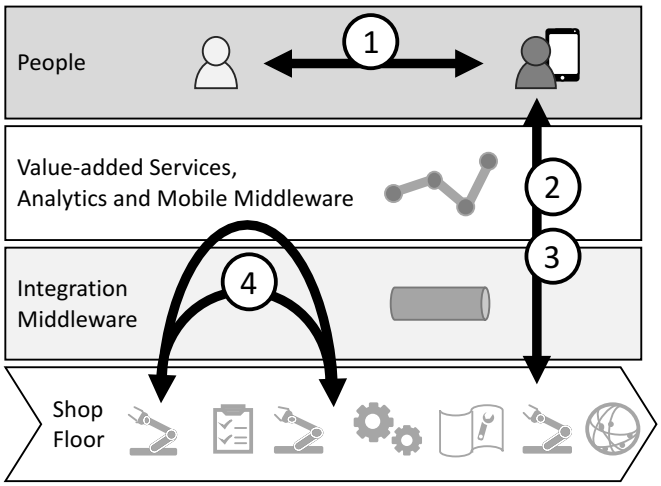

Figure 1. Simplified SITAM architecture with highlighted communication paths (adapted from [2])

this paper. Then we give an overview of related work.

An important prerequisite for the *Social Factory* is a *data-driven factory* which provides flexible access to all IT systems and data along the product lifecycle [2]. It exploits this data to keep human workers in the loop. The *Stuttgart IT Architecture for Manufacturing (SITAM)* defines such an agile, learning and human-centric architecture, overcoming the limitations of traditional manufacturing IT architectures. A simplified view of the architecture is shown in Fig. 1. The SITAM spans the entire product life cycle and comprises all processes, IT systems as well as physical machines and other data sources [2]. Its *Integration Middleware* provides service-oriented access to resources via a hierarchy of lifecycle-phase-specific enterprise service buses. It also provides data exchange formats, mediation and orchestration functionalities as well as a SOA Governance Repository to manage all services [4]. On top of this integration layer, the *Analytics Middleware* and *Mobile Middleware* provide analytics functionality for both structured and unstructured data as well as mobile data handling. These two layers build the basis for value-added services offered as apps via a marketplace to users. The SITAM also addresses cross-architectural topics such as data quality, security, privacy and governance. This architecture as depicted in Fig. 1 is ideally suited to facilitate the communications paths necessary for a Social Factory: *Human-to-Human* (1), *Human-to-Machine* (2), *Machine-to-Human* (3) and *Machine-to-Machine* (4).

Another important foundation for the introduction of a social network in a production environment is a context model [5] as the basis for transferring the factory environment into the digital world. The context model represents a *digital twin* of the real production environment, which includes the states of all objects, e.g., machines, tools, production processes, material, and others. Such a context model was defined in a previous research project and was used for the detection of failures and the management of the repair process [5]. Here the workers were integrated using *human tasks* in workflows. However, this approach is not flexible enough for a truly smart factory, since it uses pre-modeled workflows that can only handle a specific task. In contrast, a social network as proposed in this paper allows for free communication and collaboration between humans and machines.

Building on the factory context model developed in [5], [6] defines a reference architecture and detailed data collection and analytics processes for automated exception escalation in the smart factory. This includes the integration of structured data, e.g., sensor data, with unstructured data, e.g., text data. Exceptions are recognized by a wide range of cues such as explicit exception codes sent by machines, sensor data outputs, data patterns, and warnings or requests for help sent by human workers through text-based communication channels. Exceptions are then further analyzed in order to assign potential solutions or find experts who can develop new solutions. This includes a decision logic for ranking exceptions by urgency and resolving conflicting information, as well as data mining capabilities to automatically discover unusual situations from process and sensor data. The *Social Factory* concept is derived from the concept and architecture in [6]. It deliberately excludes some of its functionalities, focusing instead on the development of a concrete social-media-like communication infrastructure for shop floor management and problem resolution.

In recent years, the *Internet of Things* (IoT) has become a very active research topic thanks to the availability of cheap hardware devices and new approaches in network technology [7]. The IoT connects these devices, equipped with *sensors* and *actuators*, to enable what is called smart environments. Smart environments are monitored through interconnected sensors and can be adapted through actuators. However, monitoring such a large amount of sensor data distributed over several sensors and a timely reaction through actuators is challenging, especially in time-critical scenarios.

To cope with these challenges, [8] introduce a cloud-based situation recognition service, which is part of *SitOPT* [9] – an approach for situation-aware adaptive workflows. This service enables the derivation of high-level situations based on a large amount of raw sensor data by introducing a method and system architecture for efficient situation recognition. In the context of this paper, a *situation* or *exception* is defined as *a changing state in the production environment* that may lead to errors such as machine failures. After registering the sensors of the environment which is to be monitored, domain users can create Situation Templates. A Situation Template is a domain-specific model that allows users to define situations without needing to have deep technical knowledge of sensors. After modeling, the Situation Templates are transformed into an executable representation using, e.g., complex event processing technologies. This model is continuously executed, monitoring the sensors of the environment. Situation-aware applications can register on specific situations and are notified on their occurrence in a timely fashion. In the context of this paper, the situation recognition of SitOPT is used to detect situations in a production environment, e.g., machine failures.

Social Media comprises various platforms for creating

and exchanging user-generated content by using Web 2.0 technologies [10]. Besides social software such as blogs, wikis and micro blogging services, social networks are the most popular form of Social Media. The most common features of social networks are user profiles, friend-lists, private chat channels and walls where users can post public messages. Most interactions between users in social networks are text-based, although users are able to share audio or video files. Social Media has become ubiquitous today, but it is not that widespread in the professional context. Some companies use public social networks to connect to their customers and some companies use internal social networks to foster the communication between their employees [11]. But so far, companies do not connect their social networks to tools or machines within the production process.

Use case tailored social networks try to increase the usefulness by embedding additional features into the system. Wang et al. built a social network for efficient household water use management, which encourages the users to save water through gamification [12]. Magoutis et al. presented a social network for cloud deployment specialists with embedded tool support for monitoring and deploying cloud applications from inside their social network [13]. They also added a knowledge base which is filled from external repositories and which should help users share their knowledge. Although there are some examples for domain-specific social networks, there is no manufacturing social network.

While social networks connect humans with humans and the Internet of Things connects things with things, a concept called *Social Internet of Things* tries to close a gap by connecting humans and things [14]. In the social internet of things humans and things participate equally in a social network system. It also enables interactions between things and between humans and things. Zhang et al. propose an architecture for a smart home based on the social web of things [15]. Based on this architecture they have implemented a smart home environment, which uses a microblogging service to enable humans to interact with each other and to notify humans of sensor-based situations, but there is no direct text-based communication between humans and devices.

The text-based nature of interactions in Social Media enables the use of *Chatbots* for representing non-human entities and services. The idea of chatbots, artificial intelligences which mimic human conversation and with whom one interacts through written messages, dates back to the very beginnings of computer science [16] and was first fully fleshed out in [17] with ELIZA, a simple rule-based chatbot following the principles of conversational psychotherapy. A wide range of chatbots has been developed throughout the years for a variety of purposes, e.g., e-commerce or knowledge management. The interaction with a chatbot or artificial intelligence assistant provides an intuitive interface to services and information. So far, no domain-specific manufacturing chatbot exists. In the *Social Factory*, we employ chatbot functionalities for human-machine interaction as well as for handling exception reports and requests for help made by human workers.

3. The Social Factory

This section presents the concept of the *Social Factory*. We first discuss its goals and the general vision, then derive requirements for the *Social Factory* on the basis of which we develop our architecture and implementation in Section 4.

3.1. Goals and Vision

The fundamental goal of the *Social Factory* is to connect human workers, machines and data in the smart factory in a meaningful way which allows the workers to better do their jobs. It goes beyond prior approaches to shop floor information systems in that it fully integrates human workers and their implicit expert knowledge as well as machines and their sensor data. It then provides humans with the means to naturally communicate with machines and access knowledge sources via textual messages. By presenting machines as communicative partners that can be spoken to and give answers, the *Social Factory* motivates human workers to engage with their workplace.

In order to achieve this, the *Social Factory* uses communication channels modeled on those developed for Social Media, which are well-known and user-friendly. A social network structure is used for integrating humans and machines as well as for integrating, managing, and filtering data. Access to the Social Media interfaces is given within the factory shop floor setting.

Obviously, our technical concept is distinct from the socio-political use of the term 'social factory' for example in [18].

One important focus of the *Social Factory* is exception detection and handling as described in [6]. This is best achieved by creating and managing a knowledge base about historical exceptions or solutions drawn from manuals. This knowledge base can then also be used to document knowledge and exception events. Strong analytics tools are also needed to process data from the knowledge base and the real-time social network environment in order to recognize exceptions and support workers in problem-solving tasks. Thus, the *Social Factory* can become a self-learning system, which logs both sensor data and human knowledge and provides the analytics capabilities to draw relevant information from them.

3.2. Requirements

From this vision of the *Social Factory*, we can derive the following requirements:

1) The *Social Factory* needs to provide the right information at the right time and place (anywhere & anytime).
2) The *Social Factory* needs to optimally support the human users even under busy working conditions.
3) The *Social Factory* needs to engage human users to contribute their implicit knowledge about re-occurring manufacturing situations.

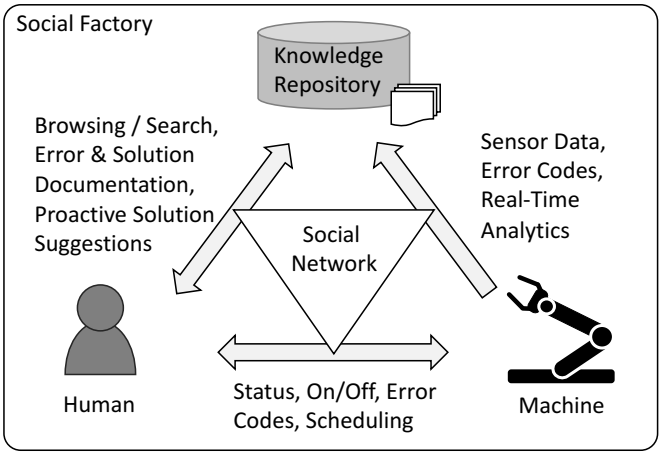

Figure 2. Information flows between humans, machines and knowledge repository in the Social Factory

4) Both humans and machines need to be represented as first-class citizens in the *Social Factory*.
5) Exceptions need to be recognized by the system quickly and reliably on the basis of available data, both from structured data (e.g., sensor data) and from unstructured data (e.g., human input).
6) The *Social Factory* needs to be a learning factory in the spirit of [2].

Figure 2 shows the information flows that are enabled by a *Social Factory* fulfilling these requirements between three main participants: human users, machines, and the knowledge repository. Humans are able to interact with machines, e.g., to obtain status and error information, switch them on or off and schedule production tasks. Machines send sensor data and error codes to the knowledge repository, where they are stored alongside real-time analytics results derived by an analytics middleware. Human users can document events in the knowledge repository as well as browse and search it deliberately and manually. They are also provided with proactive exception solution suggestions from the knowledge repository via the analytics middleware when they ask for help via the social network.

4. Architecture and Implementation

The architectural and structural components of the *Social Factory* are shown in Fig. 3. It includes entities from both the physical world and the digital world. Physical entities are represented in the digital world via their digital twin, which in this case means their *Social Network* user profiles.

The digital system components consist of multiple subcomponents: The *Social Network*, an analytics middleware and a knowledge repository. The middleware is comprised of three components: Two *Handler* components for managing interaction with machines and exception escalation and a *Text Analytics* component to analyze natural-language input. All three middleware components work together to enable *Chatbot functionalities* which allow human workers to speak directly to machines and to the handler components. The *Knowledge Repository* is a knowledge base containing machine documentation as well as historic error and solution data. Further, the situation recognition service from *SitOPT* is integrated into the *Social Factory* as an external digital component.

In the physical world, the *Social Factory* includes human users and machines. Human users communicate with machines and with each other through the *Social Network* frontend. Machines use the external *SitOPT* component to report situations based on sensor values to the users. Both human users and machines are represented in the *Social Network* by worker and machine profiles, respectively. Similarly, a chatbot named *Exception Bot* is represented using a special user profile – the *Exception Bot profile*. The *Expection Bot* is the communicative interface to exception handling and machine interaction handling tasks.

We chose the technologies and software components for our implementation carefully after thorough comparison of different alternatives. For example, we compared several open source social networks to find the most suitable one for our approach based on selection criteria such as extensibility and accessible interfaces. In a similar fashion, the databases, programming languages and communication protocols to be used have been compared and selected. As our data for the prototypical implementation, we use documentation as well as error data from the *Application Center Industrie 4.0* [19] and *Raspberry Pi* mini computers to simulate additional machines and sensors. The following sections present the digital system components of the *Social Factory* in greater detail.

4.1. Social Network

The *Social Network* component is the interface for users using the *Social Factory*. It provides classical features of social networking services like user profiles, friendlists, private chats or walls, where users can post public messages. As communication channels for text-based interaction we use both the wall, which presents a chronological list of publicly posted messages on a profile's home page, and the chat, which constitutes a direct and private one-to-one or group communication channel (cf. Fig. 6 for a depiction of a profile home page with wall and chat window). Besides that, we have added abstract roles and user profiles in order to integrate machines into the *Social Network*. Each machine has its own user profile, which has the user role *machine*. Human worker users can use those machine profiles to chat to the machine or to post to its wall.

Every user generated chat or wall message is passed to the *Text Analytics*, which processes it and then forwards it either to the *Exception Handler* or the *Machine Interaction Handler* for further processing (cf. Section 4.3). The following actions are currently supported by the system:

1) Status requests: Users can ask for the status of a machine and the machine will answer by providing its current status, e.g., current values of its sensors.

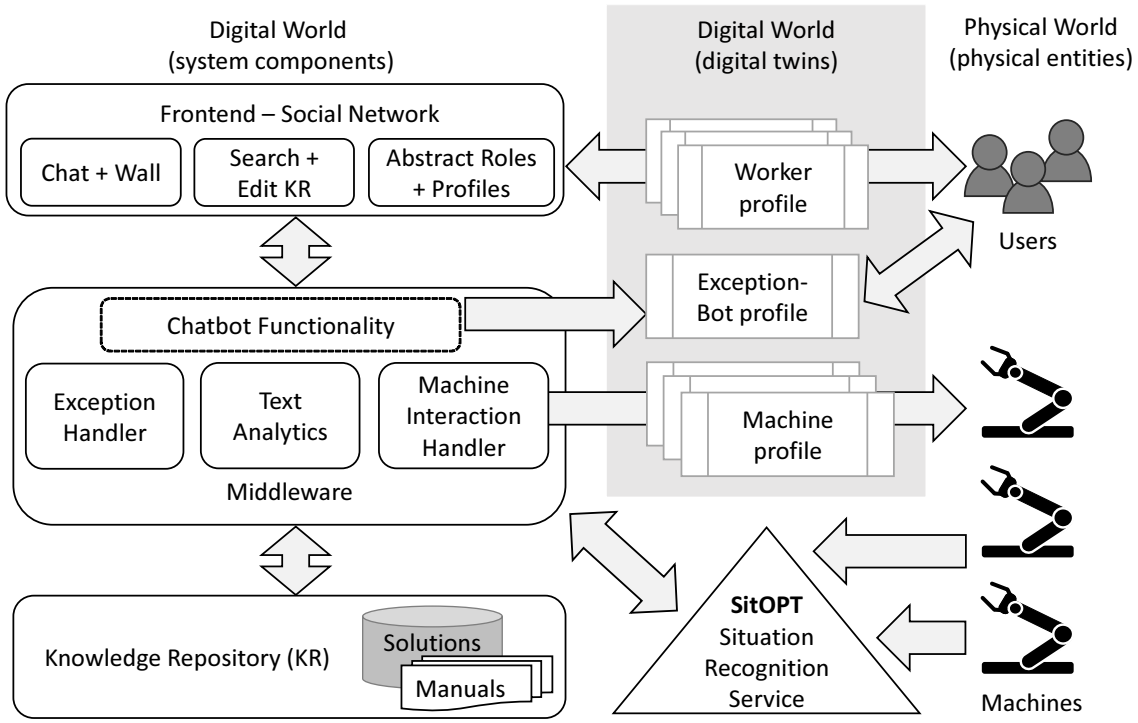

Figure 3. Architectural Components and Participating Entities in the Social Factory

2) Switching machines on or off: Users can switch the machine on or off via the *Social Network*. Since turning a machine on or off is an important decision, we classified this function as a *critical operation* and introduced a special user role called *maintenance engineer*. Each machine can have one or more *maintenance engineers* who are allowed to perform *critical operations* on this particular machine.
3) Call for help: If a user has a problem with a certain machine, they can report it via the chat or the wall of the machine or directly to the Exception Bot. The *Social Factory* will try to match it to a known problem in the *Knowledge Repository*. If the problem can be identified and has clearly defined solutions in the *Knowledge Repository*, the *Social Network* will present those solutions through the currently active communication channel. Otherwise, additional documents such as handbooks and documentation, which are also accessible through the *Knowledge Repository*, will be searched for similar issues. If a related paragraph is found in such a document, the *Social Network* will offer a hyperlink to the corresponding document.
4) Expert search: Users can also search for experts of a given machine. The *Social Factory* will then provide a list of the machine's maintenance engineers. Additionally, this mechanism could be extended to search the *Knowledge Repository* for users who provided a lot of solutions or troubleshooting documentation for this particular machine.

Besides interacting directly with a particular machine, users can also post messages to their own public wall. If one of the actions is recognized, the *Exception Bot* user profile will answer their inquiry. For this to work, the *Exception Bot* user needs a unique identifier of the machine to interact with, e.g., the user name of the machine in the *Social Network*. The *Exception Bot* user can also be contacted directly through chat communication (cf. Fig. 5). Additionally, it can join in to suggest possible solutions as two human users discuss a machine problem on the chat channel.

Users of the *Social Network* can also directly search the *Knowledge Repository*. They can access the list of known errors and their known solutions. They can also view the history of error incidents, which comprises all errors reported by the users of the *Social Network* and all errors recognized by the situation recognition service of *SitOPT*. The *Social Network* also provides an interface for users to upload documents into the *Knowledge Repository*, e.g., manuals, and an interface to manage the existing documents.

The *Social Network* component is implemented using the open source social networking engine elgg [20], which offers a desktop and a mobile interface. To implement the features described above, we extended elgg with several plugins. We implemented plugins to introduce our new roles (*maintenance engineer*, *machine* and *worker*) and their responsibilities. Additionally, we implemented plugins for interfacing other components of the system.

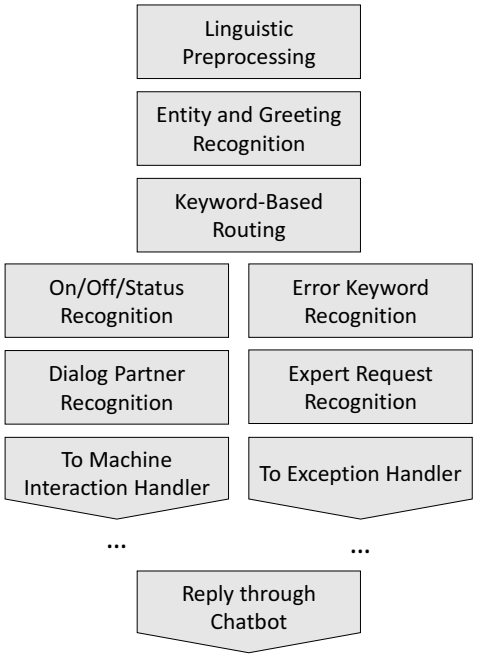

Figure 4. Text analytics pipelines

4.2. Knowledge Repository

The *Knowledge Repository* is used to store machine and error documentation to be used for proactive solution suggestions. It is a federated repository which stores structured, semi-structured and unstructured information. There are three categories of text-based semi-structured reports: (1) error descriptions for distinct error types, (2) solution descriptions for solutions which were found to apply to an error type, and (3) error incident reports for individual occurrences of errors. These are stored in relational database tables alongside structured metadata. Unstructured documents, for instance machine documentation PDFs, are stored in a fully indexed document store. Error descriptions and solution descriptions are tagged with keywords extracted from the text for easy retrieval.

The *Knowledge Repository* can be actively accessed for documentation and search through the *Social Network*. Further, the *Knowledge Repository* is accessed by the *Exception Handler* with search keywords extracted from chat and wall posts to retrieve potential solutions for current errors on the shop floor and to automatically create error incident reports based on chat and wall messages or on situations recognized by *SitOPT*.

We have implemented the *Knowledge Repository* as a combination of a MySQL database [21] and a Java component using Apache Solr [22] and Apache Tika [23], which manages PDF files.

4.3. Text Analytics

The *Text Analytics* component is part of the middleware and responsible for understanding natural language input into the *Social Network* via the chat or wall channels. It thus forms the backbone for the chatbot-like functionalities of the *Exception Bot* system user and the machine user profiles. It also processes error incident reports to derive keyword tags before they are uploaded into the *Knowledge Repository*. The *Text Analytics* component consists of a complex and modular processing pipeline, which is shown in Fig. 4. This pipeline analyzes the content of a text message and determines its type (cf. Section 4.1), then routes it to either the *Machine Interaction Handler* (cf. Section 4.5) or the *Exception Handler* (cf. Section 4.4). The *Machine Interaction Handler* proceeds to carry out the recognized command, the *Exception Handler* initiates searches for experts and solutions via the *Knowledge Repository*. A generated response is then given through the appropriate text-based communication channel. The *Text Analytics* component is implemented in Java in compliance with the Apache UIMA Framework [24] and using the uimaFit libraries [25]. It analyzes input through a mix of simple keyword and pattern recognition and more sophisticated syntactic analysis. The keywords for different situation categories are stored in a relational database. They can be modified through an elgg-plugin in the administrative area of the *Social Network*.

4.4. Exception Handler

The *Exception Handler* is one of the core components of the introduced architecture and is represented in the *Social Network* through a unique user profile, the *Exception Bot*, in order to interact with the users and machines. Its main goal is the reaction to occurring situations, either detected by users of the *Social Network* or by *SitOPT* (cf. Section 2). The situations detected by users are extracted through the *Text Analytics* component described in Section 4.3. Through this component, a machine-readable situation can be derived based on textual user input entered through the chat function of the *Social Network* or through the *Exception Bot*'s wall. The situations detected by *SitOPT* are already in the right machine-readable format. Once a situation is detected, it is handed to the *Exception Handler*, which is responsible for initiating steps to react properly. Based on this information, the *Exception Handler* finds responsible *maintenance engineers* in the *Social Network* and provides them with information about the situation and, if possible, with a list of potential solutions. These solutions are extracted from the *Knowledge Repository* as described in Section 4.2. Furthermore, an *Error Incident* entry is created automatically in the *Knowledge Repository*, so that situations which have already occurred are documented. This allows for analysis based on the *Knowledge Repository*, e.g., to predict situations. In summary, the *Exception Handler* represents the interface between recognized exceptions (situations) and the persons that are able solve them, e.g., *maintenance engineers*.

The *Exception Handler* component is implemented in Java exclusively. The main functionality of this component comprises the communication with the *Social Network*, the *Text Analytics*, the *Machine Interaction Handler*, and the *Knowledge Repository*.

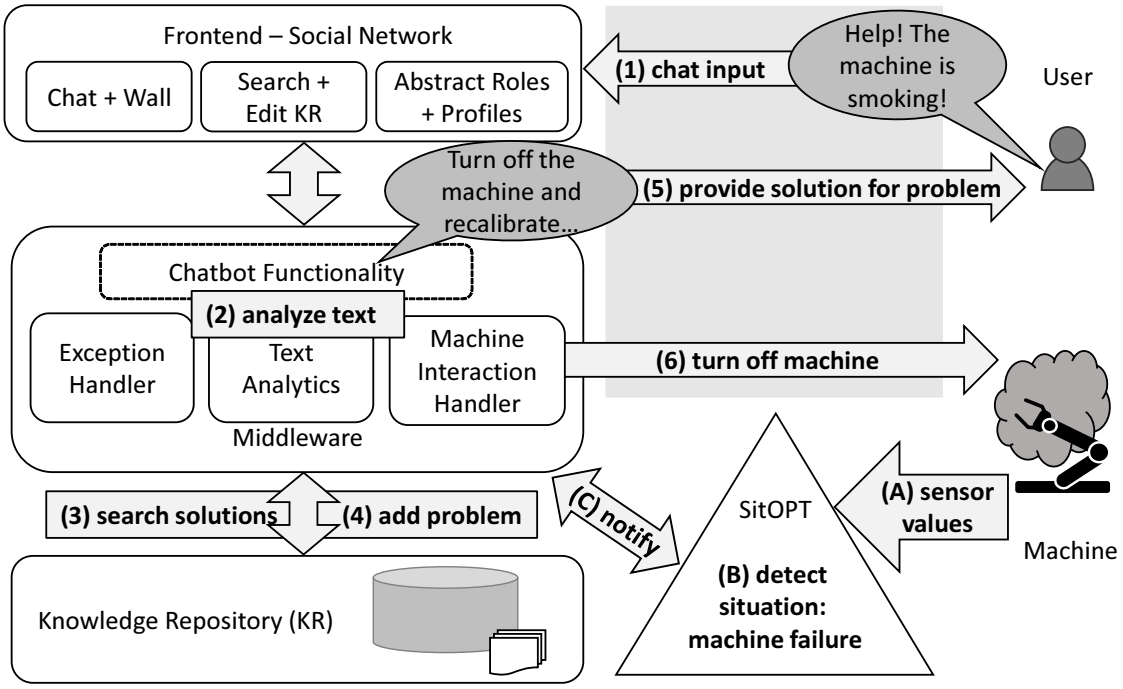

Figure 5. Social Factory Application Scenario and Use Case

4.5. Machine Interaction Handler

We developed the *Machine Interaction Handler* (MIH) as an interface between machines/sensors, *SitOPT*, and the *Social Network*. Every machine in the production environment is represented both in the *SitOPT* database and in the *Social Network* by a machine profile. This is necessary to enable exception recognition through *SitOPT* for each machine in the *Social Network*. The main task of the *MIH* is the synchronization of these components. That is, the *MIH* connects the *Social Network* and the *machine and sensor registry* component of *SitOPT* [26]. This registry offers a means for notification on the registration of new machines, on changes of existing machines or on removal of machines in *SitOPT*. Thus, *SitOPT* already offers a worthwhile abstraction from physical machines and sensors.

The *MIH* implements bidirectional synchronization of machine profiles between *SitOPT* and the *Social Network*: If a new machine is added or changed in *SitOPT* through its registry component, the *MIH* adds a corresponding new machine profile to the *Social Network*, which contains all information about the associated sensors, the geo location, and other metadata. If the machine is edited or deleted from *SitOPT*, the *MIH* accordingly edits the machine's profile information, or removes the machine profile from the *Social Network*. Also, if a new machine profile is registered in the *Social Network*, the *MIH* sends a message to the *SitOPT* registry components with all necessary machine and sensor information and *SitOPT* registers it. In this manner, the *MIH* provides a full decoupling of machines/sensors, *SitOPT*, and the *Social Network*.

Furthermore, the *MIH* offers a means to switch machines on or off through the *Social Network*. Authorized users, i.e., *maintenance engineers* that are responsible for certain machines, can switch them off through the messaging functionality of the *Social Network* or through the machines' walls using textual input, e.g., "turn off machine immediately".

The interfaces of the MIH are realized through HTTP due to the requirements of *SitOPT*, also assuming that the machines offer a corresponding interface for switching them on or off. In the future, we plan to support more interfaces to machines based on established Machine-to-Machine standards such as OPC-UA [27].

4.6. Interfaces and Communication

For communication between the components we have implemented REST-based messaging interfaces. To enable stateless communication we have defined shared message types which are used by every component. In order to ensure statelessness, each component in turn adds its specific information to the message while it is processed by the system.

For example, in the case that a user asks for help handling an exception, the *Social Network* creates a message containing the unique user name of the user, the communication channel (chat or wall) and the text written by the user. In the next step the *Text Analytics* processes the text and enriches the message by adding keywords which will be used by the *Knowledge Repository* to search for solutions. It also recognizes the call for help scenario and sets the type of the message to *errorIncident*. Based on the type, the message

is forwarded to the *Exception Handler*, which sends the message to the *Knowledge Repository*. After the possible solutions have been found, the *Knowledge Repository* adds the solutions to the message and sends it back to the *Social Network*, which can inject the message into the channel of the correct user.

5. Application Scenario and Prototype

In our approach, there are two ways in which exceptions on the shop floor can be detected: (1) by factory workers that detect occurring exceptions, or (2) by the automatic situation recognition system *SitOPT* based on context data. Both options are described in the following using a typical use case scenario. The individual steps and data flows for this use case can be seen in Fig. 5.

We assume that a worker on the shop floor detects an erroneous behavior of a machine, e.g., that there is smoke coming out of it (cf. Fig. 5). The worker reports this problem through a chat message that can be either addressed to the affected machine or to the *Exception Bot* via its abstract user profile (step 1). Alternatively, the message may also be posted to the wall of either the machine or the Exception Bot. Either way, the message is forwarded to the *Text Analytics* component in the middleware. The *Text Analytics* extracts important information about the occurring error from the text. In this case, e.g., it extracts a signal word for a problem (”Help!”) and a keyword for the error that occurs (”smoking”). The affected machine is derived from the addressee of the chat message or from a machine identifier mentioned in the text if the chat was directed at the Exception Bot (step 2). This information is handed to the *Exception Handler* and can be used to find a suitable solution for the problem. We assume that such a solution is provided inside the *Knowledge Repository*, which offers a means to search for solutions using the extracted keywords. Furthermore, information about the problem incident is added to the *Knowledge Repository* (steps 3 and 4). In case of the smoking machine, the solution could be switching it off immediately and recalibrating its working temperature. After the solution is retrieved from the *Knowledge Repository*, the user receives a chat message stating that they need to turn off the machine immediately (step 5). Thus, the user still has full control about whether the machine will be switched off or not. If the user decides to act on the suggestion, they need to send the machine a chat message, e.g., “Turn off!”. Note that the worker needs to have permissions to turn off the machine, i.e. be a *maintenance engineer* for this machine. Otherwise, the user needs to contact the responsible *maintenance engineer* through the *Social Network*. After the chat message is sent to a machine, it is forwarded to the *Text Analytics* component, which analyzes the message and forwards it to the *Machine Interaction Handler* that switches it off (step 6). The user then receives feedback through the *Social Network*.

The same problem situation, i.e. the smoking machine, could also be detected by an automatic situation recognition system such as *SitOPT*. In this case, exceptions are recognized automatically based on raw sensor data from the shop floor – in our example, the output of a smoke sensor (steps A and B). The detected exception is handed directly to the *Exception Handler* in a machine-readable format (step C). The *Exception Handler* then searches for a solution in the *Knowledge Repository*. Assuming, as before, that such a solution can be found, it contacts the *maintenance engineer* responsible for the specific machine through the chat channel of the *Social Network* and documents the exception incident in the Knowledge Repository (steps 3, 4, and 5). In addition, the occurred exception is posted to the machine’s profile in order to publicly notify everyone who may be affected by it. In our example, the solution can be applied as described before.

Figure 6 shows the user interface of the *Social Factory* as implemented in our prototype. In the center, we see the wall of the user Busy Bill where he can post public messages, like status messages, problems, or questions, to the *Social Network*. At the right of the page there is the chat interface where a user can chat with other users or machines. Fig. 6 shows a similar case as Fig. 5. The *maintenance engineer* Busy Bill is responsible for a machine called Lasermachine#42, which just posted to him (and the other responsible *maintenance engineers*) that the situation ”smoke” occurred (step 1). Since Busy Bill cannot discover the cause for smoke at this machine, he posts his question to his wall. The *Social Factory* analyzes his questions and posts the found solution via the *Exception Bot* to his wall again (step 2). Since the solution suggests that the machine could have overheated, Busy Bill verifies that by asking Lasermachine#42 for its status. The Lasermachine replies with the current sensor data of its attached sensors (step 3). Busy Bill now sees that the temperature sensor reports 2453 °C and recognizes that the machine must be overheated. According to the solution proposed by the *Social Factory* he decides to turn off the machine and recalibrate the laser (step 4). Since this is a *critical operation* the *Social Factory* demands that Busy Bill confirms his decision by clicking a provided link. If Busy Bill wasn’t a *maintenance engineer* for this machine, the *Social Factory* would decline his request. After Busy Bill has confirmed his request, the machine updates its status by sending it to all responsible *maintenance engineers*. Finally, the machine is turned off (step 5) and Busy Bill can begin to recalibrate the laser.

In summary, our system handles exception recognition from unstructured text and from structured sensor data. In both cases, corresponding solutions can be provided to the workers and also executed via the *Social Network*.

6. Evaluation

We can now evaluate our architecture and implementation of the *Social Factory* with respect to the requirements defined in Section 3.2.

The *Social Factory* provides a mobile and desktop variant of its user interface, which allows human workers to access it anywhere and anytime while moving around doing their work on the shop floor. It also includes real-time analytics for instantaneous replies to text-based requests, machine status

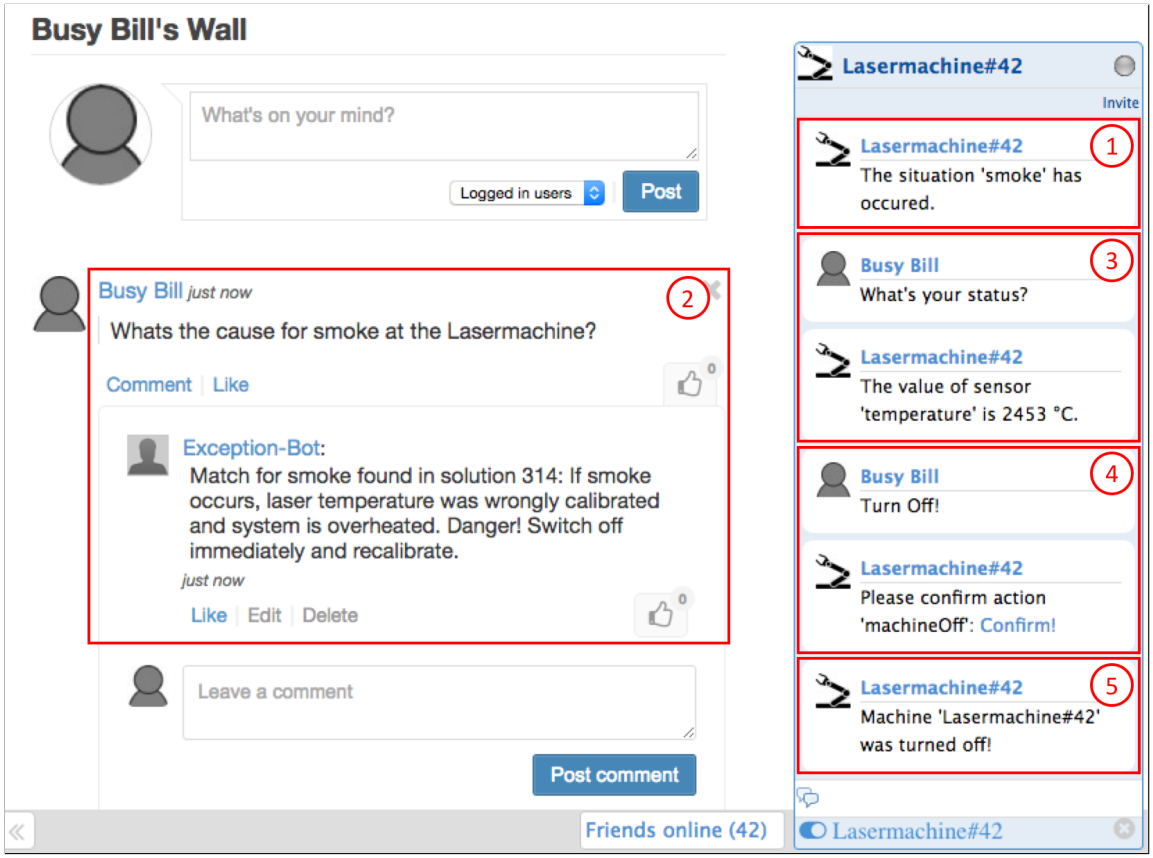

Figure 6. User Interface of the Social Factory prototype

queries or orders to switch machines on and off. Thereby it fulfills requirement 1, providing the right information at the right time and place. The easy-to-use interface with a familiar social network structure and the mobile / desktop access as well as the text-based interaction (which could be extended to speech-based interaction) also allow for accessibility under busy working conditions, thus fulfilling requirement 2. Fulfillment of requirement 3, engaging human users to contribute implicit knowledge, is achieved by the interactive social network character, which turns machines into quasi-animate conversation partners. The social network also provides a straightforward interface for documenting incidents, sharing knowledge and uploading resources as well as using them. The representation of machines as partners with Social Media profiles also contributes towards fulfillment of requirement 4, the appropriate representation of humans and machines: every physical entity has a *digital twin* in the *Social Factory*, with a flexible role system with different permissions and features. Since the activities of machines in the *Social Network* are handled by the *Machine Interaction Handler* and the situation recognition service of *SitOPT*, machines do not need to be ”smart” per se. Even older machines can be integrated into the *Social Factory* provided they are equipped with sensors and some degree of automation. The *SitOPT* component and the *Text Analytics* component work together to provide quick and reliable, data-based exception recognition in fulfillment of requirement 5. Finally, requirement 6 of the learning factory is fulfilled by the extensive documentation and analytics capabilities supported by *SitOPT*, *Text Analytics* and the *Knowledge Repository*.

We have implemented a fully functional, highly modular and easily extendable version of the *Social Factory*. Concept, development and testing were carried out in the course of a year-long project with 11 students, amounting to 2.5 man-years. The *Social Factory* will be provided as an open source software and can thus be easily applied in a number of real-world and research settings.

7. Conclusions and Future Work

In this paper, we have presented the concept of the *Social Factory* as a data-rich and data-driven manufacturing environment enabled by natural language communication channels within a social network. We have implemented a prototype for the *Social Factory* using state-of-the-art open source technologies in the areas of text analytics, data integration, Social Media, web and messaging as well as cutting-edge research in the area of situation recognition. The system is modular, easily configurable and easy to set up.

We have provided a realistic use case scenario to illustrate the benefits of the *Social Factory* and have evaluated our concept and implementation against a set of requirements developed from the data and physical situation in a smart factory environment.

In our future work, we intend to improve the sophistication of the text analytics component and implement push notifications for better mobile information provisioning. The solution suggestion process will be enhanced by allowing users to rate the quality of a provided solution and learning from this feedback. We are also looking into calendar integration for easier scheduling and automated expert availability judgments, as well as a more refined role system for designating informal experts in addition to official maintenance engineers. Finally, we want to test the *Social Factory* in a real-world shop floor setting.

Acknowledgments

The authors would like to thank the students of the project "Connecting Users & data in Production for Context-Aware esKalation of Exceptions (CUPCAKE)" for implementation work on the *Social Factory*. We also thank the German Research Foundation (DFG) and Daimler AG for financial support of this project as part of the Graduate School of Excellence advanced Manufacturing Engineering (GSaME) at the University of Stuttgart, and the DFG project *SitOPT* (Grant 610872).

References

[1] H. Kagermann, J. Helbig, and W. Wahlster, *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0: Securing the Future of German Manufacturing Industry; Final Report of the Industrie 4.0 Working Group*. Forschungsunion, 2013.

[2] C. Gröger, L. B. 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 *Proceedings of the 18th International Conference on Enterprise Information Systems*, 2016.

[3] J.-B. Waldner, *CIM, principles of computer-integrated manufacturing*. Chichester, West Susssex, England and New York: Wiley, 1992.

[4] J. Königsberger and B. Mitschang, "A Semantically-enabled SOA Governance Repository," in *Proceedings of the IEEE 17th International Conference on Information Reuse and Integration*. IEEE, 2016.

[5] M. Wieland, F. Leymann, M. Schäfer, D. Lucke, C. Constantinescu, and E. Westkämper, "Using Context-aware Workflows for Failure Management in a Smart Factory," in *Proceedings of the 4th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM)*. Florence, Italy: IARIA, 2010.

[6] L. B. Kassner and B. Mitschang, "MaXCept – Decision Support in Exception Handling through Unstructured Data Integration in the Production Context: An Integral Part of the Smart Factory," in *48th Hawaii International Conference on System Sciences (HICSS)*, 2015.

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

[8] P. Hirmer, M. Wieland, H. Schwarz, B. Mitschang, U. Breitenbücher, and F. Leymann, "SitRS - A Situation Recognition Service Based on Modeling and Executing Situation Templates," in *Proceedings of the 9th Symposium and Summer School On Service-Oriented Computing (SummerSOC)*, 2015.

[9] M. Wieland, H. Schwarz, U. Breitenbücher, and F. Leymann, "Towards Situation-Aware Adaptive Workflows," in *Proceedings of the 13th Annual IEEE Intl. Conference on Pervasive Computing and Communications Workshops: 11th Workshop on Context and Activity Modeling and Recognition*. St. Louis, Missouri, USA: IEEE, 2015, Workshop Paper.

[10] A. M. Kaplan and M. Haenlein, "Users of the world, unite! the challenges and opportunities of social media," *Business Horizons*, vol. 53, no. 1, 2010.

[11] J. DiMicco, D. R. Millen, W. Geyer, C. Dugan, B. Brownholtz, and M. Muller, "Motivations for social networking at work," in *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '08. New York, NY, USA: ACM, 2008.

[12] Z. Wang and A. Capiluppi, "A specialised social network software architecture for efficient household water use management," in *Proceedings of the 9th European Conference Software Architecture, ECSA 2015*. Cham: Springer International Publishing, 2015.

[13] K. Magoutis, C. Papoulas, A. Papaioannou, F. Karniavoura, D.-G. Akestoridis, N. Parotsidis, M. Korozi, A. Leonidis, S. Ntoa, and C. Stephanidis, "Design and implementation of a social networking platform for cloud deployment specialists," *Journal of Internet Services and Applications*, vol. 6, no. 1, 2015.

[14] A. M. Ortiz, D. Hussein, S. Park, S. N. Han, and N. Crespi, "The cluster between internet of things and social networks: Review and research challenges," *IEEE Internet of Things Journal*, vol. 1, no. 3, 2014.

[15] C. Zhang, C. Cheng, and Y. Ji, "Architecture design for social web of things," in *Proceedings of the 1st International Workshop on Context Discovery and Data Mining*, ser. ContextDD '12. New York, NY, USA: ACM, 2012.

[16] A. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 59, no. 236, 1950.

[17] J. Weizenbaum, "ELIZA - A Computer Program For the Study of Natural Language Communication Between Man And Machine," *Communications of the ACM*, vol. 9, no. 1, 1966.

[18] C. Fuchs and M. Sandoval, *Critique, Social Media and the Information Society*, ser. Routledge Studies in Science, Technology and Society. Taylor & Francis, 2013.

[19] M. Landherr, U. Schneider, and T. Bauernhansl, "The Application Center Industrie 4.0 - Industry-driven manufacturing, research and development," in *Proceedings of the 49th CIRP Conference on Manufacturing Systems (CIRP-CMS)*, 2016.

[20] Elgg, "Open source social networking engine," https://elgg.org/.

[21] MySQL, "MySQL," https://www.mysql.com/.

[22] The Apache Software Foundation, "Apache Solr," https://lucene.apache.org/solr/, 2015.

[23] ——, "Apache Tika," https://tika.apache.org/, 2015.

[24] D. Ferrucci and A. Lally, "UIMA: an architectural approach to unstructured information processing in the corporate research environment." *Natural Language Engineering*, vol. 10, no. 3-4, 2004.

[25] P. V. Ogren and S. J. Bethard, "Building test suites for UIMA components," *SETQA-NLP '09 Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, no. June, 2009.

[26] P. Hirmer, M. Wieland, U. Breitenbücher, and B. Mitschang, "Automated Sensor Registration, Binding and Sensor Data Provisioning," in *Proceedings of the CAiSE 2016 Forum at the 28th International Conference on Advanced Information Systems Engineering*, 2016.

[27] OPC, "OPC Foundation Unified Architecture," https://opcfoundation.org/about/opc-technologies/opc-ua/.

All links were last followed on 2016-06-09.