Towards the Shop Floor App Ecosystem: Using the Semantic Web for Gluing Together Apps into Mashups

Andrei Miclaus
Karlsruhe Institute of Technology
Vincenz-Prießnitz-Str. 1
D-76131 Karlsruhe
+49-721-608-417-08
andrei.miclaus@kit.edu

Wolfgang Clauss
Ondics GmbH
Neckarstraße 66/1a
D-73728 Esslingen
+49-711-310093100
wolfgang.clauss@ondics.de

Eugen Schwert
Atos Bull GmbH
Zettachring 2
D-70567 Stuttgart
+49-211-39934092
eugen.schwert@atos.net

Martin Alexander Neumann
Karlsruhe Institute of Technology
+49-721-608-417-07
martin.neumann@kit.edu

Ferdinand Mütsch
Karlsruhe Institute of Technology
ferdinand.muetsch@student.kit.edu

Till Riedel
Karlsruhe Institute of Technology
+49-721-608-417-06
till.riedel@kit.edu

Fabian Schmidt
SICK AG
Erwin-Sick-Str. 1
D-79183 Waldkirch
+49-7681-202-4172
fabian.schmidt@sick.de

Michael Beigl
Karlsruhe Institute of Technology
+49-721-608-417-00
michael.beigl@kit.edu

ABSTRACT

Any upcoming industrial revolution will rely on the ability to harness software as the nervous system of future production environments. This paper proposes an app ecosystem as key enabler of industry digitization and argues for the need of semantic web technologies as primary enablers for their interoperability. We shortly discuss how we envision the emergence of semantically annotated apps on the manufacturing shop floor. Subsequently, we demonstrate how a loosely coupled mashup of apps can easily form a full stack internet of things solution that covers sensor data from its origin toward its visualization in a web browser.

CCS Concepts

• Information systems→Mashups • Information systems→Semantic web description languages • Software and its engineering→Distributed systems organizing principles • Information systems→Web services

Keywords

App; App Ecosystem; Mashups; Shop Floor; Manufacturing Industry; Web; Semantic Web; Linked-Data;

1. INTRODUCTION

Modern manufacturing relies on continuous productivity improvements to maintain profitability. With an increasingly important role, software is a key enabler for efficiency.

However, interoperability between existing systems and the addition of new software prove to be challenging and expensive tasks due to technical and sometimes human factors. These impediments turn the shop floor IT into a rigid siloed landscape.

Software needs to become more flexible in order to adapt to business needs and more transparent in order to increase a feeling of control for the software operators. Ideally, development cycles should be shorter and integration of new functionality into existing heterogeneous systems should be frictionless. Our industry partners agree that software enabled process monitoring, analysis and quality management, increased situational awareness and predictive maintenance are improvements needed on the shop floor to cater for new products and decreased batch sizes.

We aim at bringing flexible, scalable IT to the shop floor through the introduction of apps and an app ecosystem. Apps share only a standardized runtime platform and a lightweight standard for intercommunication that couples them loosely on the application layer. This design features seamless simultaneous development of many app serving different features and allows their frictionless integration on the shop floor.

However, wiring and interfacing these apps together in a meaningful way poses challenges to the communication architecture. Due to heterogeneous data sources and multiple layers of abstraction (from machine to ERP data), it is difficult to find common ground when creating a functional ecosystem. We believe that standardization in such a domain is a very difficult endeavor that is destined to fail or to require unnecessary amounts of effort and resources. Semantic technologies however, have proven to more easily create static mashups with generic components [10] or even allow the creation of dynamic mashups [11]. Therefore, we opt for the use of web and semantic web technologies that allow a high degree of flexibility for the purpose of interconnection.

2. The App Ecosystem on the Shop Floor

The app store model, as employed by Apple and Google with great success, is a centralized single-stop location that allows many application downloads to a common target platform. It combines payment models (free, freemium, paid, and paymium) with software distribution and feedback systems as well as ecosystem governance and authentication. In unison with the software and hardware platform (Android, iOS etc.) they have formed rich application ecosystems where users can easily download and combine apps.

As the manufacturing industry is very different from the consumer market (B2C), a shop floor app ecosystem (B2B) needs to deal with additional requirements regarding the app store model (offline deployment and licensing), execution environment (offline, secure and safe updates), underlying business models (e.g. fractionalized ownership or performance based payments) and most importantly the much higher degree of interconnection between heterogeneous systems and data. The latter has a high impact on the potential for innovation and creation of new goods and services.

In our view, the app ecosystem on the shop floor should use self-contained click-to-install apps as users will already have had some experience from to their personal smart phone usage.

In the following, we will describe the central parts of the envisioned app ecosystem.

The App – Apps are defined as web applications running in minimal, lightweight Docker containers. To achieve isolation we chose docker in favor of VMs due to better performance and increased ease of use and more available open source tooling. In order to remain open and interoperable on the syntactical level apps are able to communicate over HTTP via REST level 3 [2]. On the semantical level, an app should be able to describe its APIs and data using some form of semantic web technologies such as Hydra, RDF, JSON-LD. In our model, an app may encapsulate fully-fledged applications by exposing them via a web interface. apps therefore can proxy the interaction to an entire ERP system. On the other end of the spectrum, an app may be just a stand-alone HTML browser application. An app should not have external dependencies on any other infrastructure resources such as databases used to store data specific to the app in order to attain high modularity involving low coupling and high cohesion,

The App Store – The app store has a central role in the eco-system, as it is the starting point of transactions and core authentication hub. Additional responsibilities include app provisioning, deployment, and user management. The aspects of authentication and user management can be tackled by employing WebID [9], OpenID[1] or other decentralized solutions used on the web. App provisioning and deployment are mostly delegated to the app execution system after proper authorization. The issues of firewalls and isolated networks are still open. However a combination of port 80 and offline duplication of resources can mitigate them.

The App Execution System (AES) – is the minimal layer of software and hardware necessary for running apps. It can be understood as an operating system for apps. For example, the Cloudfoundry technology (used in IBM Bluemix and GE Predix) is a thick execution system providing database or other infrastructure services. We believe that the app execution system should be minimal, in that it should only provide the minimum functionality such as app isolation, app life cycle management (deployment, update etc.) and allow a restricted set of API calls to the underlying system and hardware. Because of these reasons, we restricted ourselves to encapsulating apps in Docker[2] containers and running them on the Docker daemon while requiring apps themselves to include all needed dependencies. With a bare minimum of centralized management, each app provides meta information (semantics, endpoints, heartbeat, ram usage etc.) about itself and the payload.

Implementation of a shop floor enhancing use case entails the creation of a mashup consisting of a suite of light-weight, fine-grained, loosely coupled services (the apps), each running in a containerized environment and having its own technological stack while communicating mainly over HTTP endpoints based on semantic query subscription patterns.

While surveying the literature we mostly found analyses of app ecosystems in the public or enterprise domain [4,5,6,7,8]. However, no prior work has adapted the app concept to the shop floor and its requirements on intercommunication.

3. Manufacturing Use Case

Electronic manufacturing is a sensitive process and humidity and temperature for example, play an important role. To ensure quality, optical inspection steps are performed on every printed circuit board (PCB). However, under certain conditions issues with the PCBs may still arise. Therefore, to improve the quality and throughput, the shop floor has to be continually upgraded with new capabilities that allow employees to promptly react and reduce errors.

First, employees need to visualize the quality inspection data, both on large screens and their smartphone. This in turn allows them make timely and more informed decisions. Second, temperature and humidity sensors need to be installed in order to monitor the production environment. In case some sensors use different units (˚C vs. ˚F), they still need to present meaningful data to workers or other machines. Third, inspection, temperature and humidity data need to be aggregated for both workers and data analysts. The latter need to be able to use the data in order to train machine learning models for predictive maintenance. The fourth and final step, is the deployment of the machine learning algorithm into production and associating the predictions with machine location and PCB meta-data.

This scenario requires physical devices and software from different vendors that need to interoperate to achieve the desired outcome. While the predictive analytics step is still in development, the mashup is explained in greater detail in the following section.

4. Proof of Concept Mashup

The proof of concept mashup shown in Figure 1, consists of apps that are implemented as stand-alone Docker containers with functionality restricted to a small business value generating subset. Every app is uniquely identifiable and addressable within the web by its URL and RESTful interactions are possible. The apps have been developed independently and we have relied on common semantics to allow their interoperation.

[1] OpenID, http://openid.net

[2] Docker, https://www.docker.com/

The App Hub can be thought of as a very simple app store. It manages each app's life cycle by generating Docker compose files according to which apps are part of requested mashups.

Machine 1 and 2 are automatic optical inspection (AOI) machines. Inspection data is available via publish/subscribe (using server sent events) through the data app. Adding another machine into the mashup requires instantiating the data app container and deploying it. Inspection data is visualized inside a web app that uses responsive design. The visualization app can also associate between the different input data from the inspection machines and the aggregated temperature data from MOSCITO by leveraging the semantic annotations in the payload.

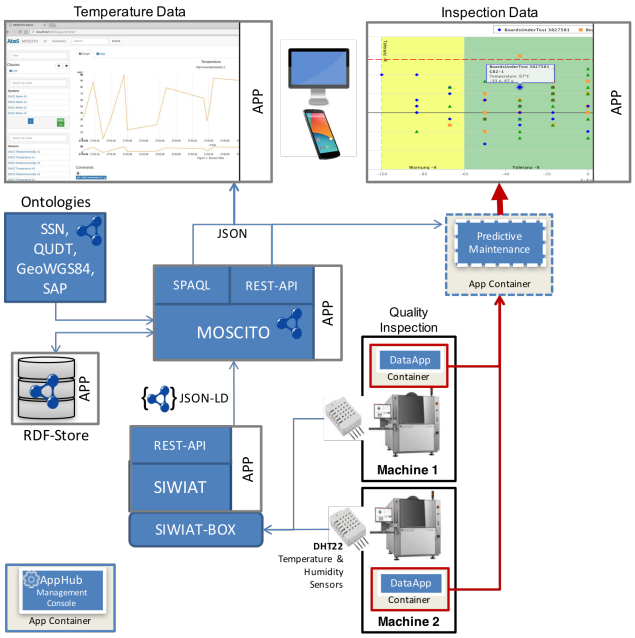

Figure 1 Prototype semantic app mashup

SIWIAT[5] is an embedded gateway-as-a-box with internal apps developed by the authors and targeted at the industrial shop floor. Its role is to wrap legacy or non web-capable sensors, annotate the measured values and make them available via a REST interface.

MOSCITO[8] is a semantic Middleware developed by the authors based on OSGI, Apache Jena and the RDF4J framework. It provides a set of functionality for the management of semantically linked data accessible through RESTful Web Services, such as ontology and rule management, SPARQL-Endpoints as well as triple store connections and data integration from heterogeneous data sources.

MOSCITO acts as shop floor semantics engine and visualizing tool that collects, connects related data form heterogeneous data sources and manages ontologies and rules. Figure 2 exemplifies the visualization of annotated data by using knowledge from ontologies.

Both SIWIAT and MOSCITO do not need to rely on a common predefined data model for exploring, visualizing or processing data. Therefore, integration of shop floor data into business processes can be performed using semantics-driven apps even as

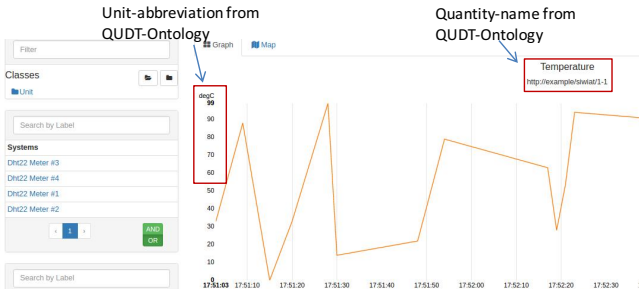

Figure 2 Temperature observation rendered in MOSCITO with help of the semantic information

proxies to enterprise resource planning (ERP) or manufacturing execution system (MES) systems.

4.1 Payload Semantics

Due to the lack of generally agreed upon shop floor ontology standards, we annotate the data by reusing existing ontologies, such as the Semantic Sensor Network (SSN) ontology[9]. Additionally, custom ontologies can be used to supplement the semantics by representing domain specific information. Listing 1 shows semantically enriched sensor value readings using the JSON-LD syntax. The Ontologies used in this sample are SSN, QUDT[10] and its extensions.

```
"@context": {
    "qudt": "http://qudt.org/schema/qudt#",
    "qudt-unit": "http://data.nasa.gov/qudt/owl/unit#",
    "qudt-quantity": "http://qudt.org/vocab/quantity#",
    "ssn": "http://purl.oclc.org/NET/ssnx/ssn#",
    "xsd": "http://www.w3.org/2001/XMLSchema#"
},
"ssn:observedProperty": {
    "@id": "qudt-quantity:ThermodynamicTemperature"
},
"@graph": [{
    "@id": "http://example.com/siwiat/17#83-resultvalue",
    "@type": "qudt:QuantityValue",
    "qudt:quantityValue": {
        "@type": "xsd:float",
        "@value": "67.0"
```

Listing 1 JSON-LD snippet exemplifying semantic sensor data annotation

This approach is independent of a specific all-encompassing ontology and can therefore be used in any domain. Moreover, a high degree of domain specificity can be attained, if custom ontologies are added to the mix in order to cover additional aspects. This way, migrations to upcoming shop floor ontologies are supported as well.

As seen in Figure 3, the embedded IoT-Gateway technology SIWIAT was extended by a prototypical Linked-Data app for interactive semantic annotation of measurement values coming from sensors and machines on the shop floor.

Each measured channel can be annotated from a list of ontology endpoints (Figure 3). Subsequent processing of sensor data is thus enriched with the first-hand knowledge about the sensor, the sensor application and the context of the measurements. The JSON-LD output is JSON-LD 1.0/W3C compliant and was

[5] Ondics SIWIAT, http://siwiat.com/de/app-box

[8] Bull, http://www.bull.com

[9] SSN Ontology, https://www.w3.org/2005/Incubator/ssn/ssnx/ssn

[10] QUDT Ontology, http://www.qudt.org/

syntactically tested with a structured data validator tool.

Selecting predefined annotations enables easy introduction of semantics on the shop floor by workers and domain experts alike. None of whom need to be experts in semantic technology. Semantic knowledge is pushed in the background and helps payloads carry the necessary information in order to allow data interpretation on demand and on various levels of abstraction.

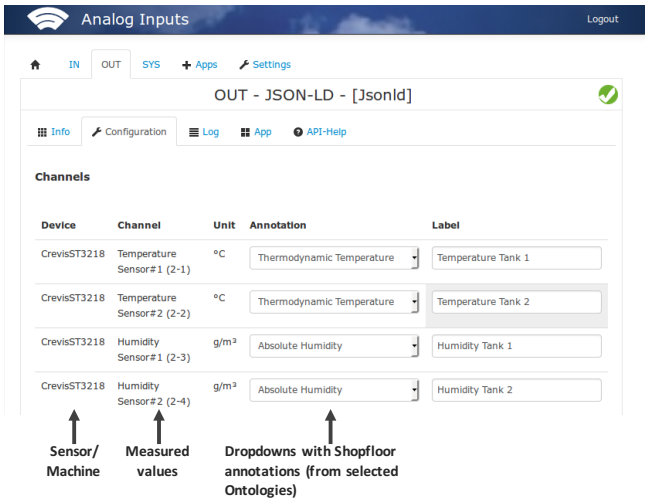

Figure 3 Semantic sensor annotation GUI within SIWIAT IoT gateway

The sensor data integration flow that we have identified as being necessary to fulfill business needs is presented in the following:

- Acquire sensor data from any sensor/machine on shop floor
- Use semantics to annotate sensor data at earliest stage
- Fuse sensor data from heterogeneous sources
- Integrate sensor data in the business process & visualize data
- Obtain and fuse other higher level information

4.2 API Semantics

Following the principles of Unix pipes, the complexity of the underlying execution environment is kept as low as possible. Apps are therefore responsible to annotate their own data and build up communications over HTTP with other apps. Therefore, every app has a Hydra described REST API. Semantically annotated APIs are a prerequisite for automatic exploration and reconfiguration or search for specific data.

Hypermedia controls can be used for self-description and is an integral part of the REST architectural style and their proper application in web APIs corresponds to maturity level 3 (highest) in the Richardson Maturity Model (RMM) [2]. All information necessary to process a resource and its related resources are attached to the resource's representation. Listing 2 shows an example from the data app GET endpoint that returns the SSN sensing device type of one of the AOI machines.

In our proof of concept, we use Hydra annotations to enable apps to crawl APIs for the specific resource they need and the protocol they can speak (server sent events, web hooks, request/reply). Demand-driven design is a topic that currently gains popularity with recent emerging technologies such as GraphQL[11] and Falcor[12]. Hydra was the technology of choice in this instance because of the syntactical and semantical compatibilities to the web. However, this may change in the future if shop floor relevant scenarios can be implemented with less effort and more simplicity when using the above mentioned techonlogies.

Leveraging this information, future apps should be able to autonomously create connections between themselves guided by consumer/producer strategies.

```
"@id": "vocab:EntryPoint",
"@type": "hydra:Class",
"subClassOf": null,
"label": "EntryPoint",
"description": "The main entry point of the API.",
"supportedProperty": [
  {
    "property": {
      "@id": "vocab:EntryPoint/device",
      "@type": "hydra:Link",
      "hydra:title": "device",
      "hydra:description": "Info about the device",
      "domain": "vocab:EntryPoint",
      "range": "ssn:SensingDevice",
      "supportedOperation": [
        {
          "@id": "_:info_retrieve",
          "@type": "hydra:Operation",
          "method": "GET",
          "label": "Retrieves the info.",
          "description": null,
          "expects": null,
          "returns": "ssn:SensingDevice",
          "statusCodes": [200, 400, 404, 500]
        }
```

Listing 2 Hydra vocabulary excerpt form the data app

5. Lessons learned

The majority of shop floors lack a flexible IT and semantic technologies are far from being the norm. While working with our industry partners, we found that prerequisites such as formalized knowledge and data engineering are rare. Furthermore, the IT environment is not flexible enough to allow efficient and effective implementation and deployment of software.

The App-Store was the most familiar and accepted deployment concept and apps let domain experts organize and bundle their functionality in a modular fashion.

While semantics allow low overhead integration of app mashups for shop floor business processes, further semantics tools are required on the shop floor in order to provide proper acquisition, visualization and processing. Nowadays however, PLC and CSV file formats still dominate and only few tools are available for testing, deploying, and monitoring semantic data streams. The transition from structured API endpoints of ERP, MES systems to semantic web endpoints require wrapping that can be realized as proxy apps.

App technology caters for the diversity of shop floor requirements and supports the evolution of semantic technologies and their usage. Payload and API semantics allow crawling API and data in a uniform manner and gluing them together using domain concepts and the web. Additionally, shop floor domain experts

[11] Facebook GraphQL, http://graphql.org/

[12] Netflix Falcor, https://netflix.github.io/falcor/

could better understand the API. This is an important first step to allow the shop floor to create its own mashups without the need for software experts.

We believe that an overall richer ecosystem will drive adoption by setting examples and proving the methods with running systems.

We found that shop floor experts understand their own domain better during the semantic formalization process. Semantic technologies should therefore be abstracted away from the end user and he should experience semantic exploration at a click and drill-down level. This can foster "semantic" thinking without the technological barrier.

Semantic annotation should take place along the entire sensor data value chain, as near as possible to the data source - the sensor - and maintained in separable abstraction levels. The different abstraction levels carry specific semantics that are needed by employees with different responsibilities. This can make domain design less complex.

We used SSN and QUDT as design aids for sensor networks and their values. The knowledge encoded in these ontologies has proven useful in formalizing, structuring and organizing our application domain. However, scenarios that are more complex will be increasingly difficult to implement due to the required number of ad-hoc intermixed ontologies.

A minimal ontology capturing the necessary shop floor concepts is an important next step towards proper semantic usage. Unfortunately, we could not find a usable ontology for the shop floor in the literature. Consequently, existing ontologies have to be mixed for a comprehensive mapping of data and infrastructure on the shop floor. Therefore, we recommend the definition of industry specific ontology for the shop floor. To avoid long standardization processes, it should be treated as a living document that reuses already existing ontologies by reference.

Large-scale deployment of web and semantic technologies requires a strategy for retrofitting machines and shop floor equipment. Already deployed technologies include old standards such as MODBUS, Hart, ProfiBus, SPI, RS-485 or even analog signals. We found that the most unobtrusive, minimal and more accepted way is to use gateways and proxy apps that bridge these signals to the layers of the web.

Exploring the possibility of using web and semantic web technologies on the manufacturing shop floor is an ongoing research effort. However, the technologies we selected are sufficient for functioning prototypes and our experiences were positive and promising up to this point.

6. Conclusion

In this paper we have proposed modularizing the application development for the shop floor and adopting an app ecosystem style of IT governance.

Certainly, apps for the shop floor have greater complexity and require increased security and a higher degree of interconnection in order to generate the necessary business value[13]. This of course poses the extra challenge of supporting multiple vendors and the heterogeneous environments typically found in IoT scenarios. In this regard, semantics are no decoration but essential requirements to make apps interoperable.

We create app interoperability by bridging the syntactical barriers via web technologies such as HTTP and JSON-LD and the semantic barriers by using semantic technologies as API glue and exchanging semantically enriched payloads using standardized web ontologies.

We sketched a technological platform based on the app paradigm that allows the rapid introduction of new IT particularly in small to medium industrial enterprises and shared our lessons learned.

7. ACKNOWLEDGMENTS

This work was funded by the German Federal Ministry of Education and Research (BMBF) as part of the ScaleIT[14] project (grant nr. 02P14B189). We wish to thank our project partners for the valuable insights they have provided us.

8. REFERENCES

[1] Dominique Guinard and Vlad Trifa. 2016. Building the Web of Things: With examples in Node.js and Raspberry Pi. Manning Publications.

[2] Martin Fowler. 2010. Richardson Maturity Model @ martinfowler.com. Retrieved June 23, 2016 from http://martinfowler.com/articles/richardsonMaturityModel.html

[3] Markus Lanthaler and Christian Gütl. 2013. Hydra: A vocabulary for hypermedia-driven web APIs. In *CEUR Workshop Proceedings*. Retrieved March 27, 2016 from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.362.4758

[4] S. Leminen, M. Westerlund, M. Rajahonka, and R. Siuruainen, "Towards IOT ecosystems and business models," Lect. Notes Comput. Sci, vol. 7469 LNCS, 2012.

[5] K Kimbler. 2010. App store strategies for service providers. 14th International Conference on Intelligence in Next Generation Networks (ICIN), 2010. doi:10.1109/ICIN.2010.5640947

[6] Jieun Kim, Yongtae Park, Chulhyun Kim, and Hakyeon Lee. 2014. Mobile application service networks: Apple's App Store. doi:10.1007/s11628-013-0184-z

[7] Stefan Wenzel. 2014. App Store Models for Enterprise Software: A Comparative Case Study of Public versus Internal Enterprise App Stores. Software Business. Towards Continuous Value Delivery, 2014, doi:10.1007/978-3-319-08738-2_16

[8] Roland M Müller, Bjorn Kijl, and Josef K J Martens. 2011. A comparison of inter-organizational business models of mobile app stores. Journal of theoretical and applied electronic commerce research 6

[9] Sambra Andrei, Henry Story, and Tim Berners-Lee. 2014. WebID 1.0: Web Identity and Discovery. Retrieved from https://www.w3.org/2005/Incubator/webid/spec/identity/

[10] Robert Battle and Edward Benson. 2008. Bridging the semantic Web and Web 2.0 with representational state transfer (REST). Web Semantics: Science, Services and Agents on the World Wide Web 6.

[11] Simon Mayer, Nadine Inhelder, Ruben Verborgh, Rik de Walle, and Friedemann Mattern. 2014. Configuration of smart environments made simple: Combining visual modeling with semantic metadata and reasoning. International Conference on the Internet of Things (IOT), 2014

[13] Metcalfe's Law, https://en.wikipedia.org/wiki/Metcalfe's_law

[14] ScaleIT Industry 4.0 project, https://scale-it.org