Dynamically optimized production planning using cross-layer SOA

Domnic Savio [1], Stamatis Karnouskos [1], Daniel Wuwer [2] and Thomas Bangemann [2]

[1] SAP Research
Vincenz-Priessnitz-Strasse 1, D-76131, Karlsruhe, Germany
{domnic.savio, stamatis.karnouskos} @sap.com

[2] ifak - Institut f. Automation und Kommunikation e.V. Magdeburg
Werner-Heisenberg-Strasse 1, D-39106, Magdeburg, Germany
{daniel.wuwer, thomas.bangemann} @ifak.eu

Abstract

Responding to the dynamic requirements of the changing market and optimising costs are two challenges faced by corporate manufacturing plants globally. In order to be agile, modern manufacturing plants employ optimized supply chain mechanisms to reduce the response time of the market needs. However bringing changes to the shop floor after a production is planned is costly. In this paper we demonstrate how Service Oriented Architecture (SOA) driven approaches offer the flexibility to adapt manufacturing plants based on a dynamic production plan in close cooperation with a backend system. The methods discussed were deployed on a prototype test rig and integrated with SAP ERP. The results introduce dynamic behaviour of the production plan by adapting to the changing nature of the shop floor, and at the same time, providing the real time status of the machines to the enterprise services which optimize further the production plan dynamically.

1. Motivation

In any industry, major manufacturing giants take effective steps in breaking up and outsourcing production activities in the effort to be more agile and achieve redundancy and performance [2]. To respond quickly to the market demands, there is a need to be able to dynamically rearrange product lines. Supply Chain Management offers a variety of strategies to minimize cost and optimize production planning. However most of the approaches followed are static based on pre-calculated global production planning and have no control on the changes that happen on the shop floor. These changes could range from the breakdown of a machine to the production excess of a particular product. By having real-time information provided to the back end system and in parallel be able to partially control the production line could provide significant advantages.

SOA is a well known approach for enterprise systems, which is actually investigated to be expanded towards the lower levels within the production systems hierarchy like the control or even field level [1]. Web services are suitable and capable of running natively on embedded devices, providing an interoperability layer and easy coupling with other components in highly heterogeneous shop-floors. Device Profile for Web Services (DPWS [5]) and OPC UA [6] are emerging technologies for realizing web service enabled controllers and devices.

As the initial DPWS consortium had partners from the printer industry, DPWS was considered mostly for the home automation industry. However, as microcontrollers form the basis of any automation unit, the extension to equipment suitable to other automation domains is to be investigated, while considering those specific characteristics, among others, like real-time aspects, availability, security or safety. Several projects such as SIRENA (www.sirena-itea.org), SODA (www.soda-itea.org) and SOCRADES (www.socrades.eu) provide a platform to develop a DPWS stack targeting the industrial automation devices on the shop floor [3]. The goal is to provide the same interoperability and easiness of integration of devices (from now on in this paper the term is used to refer mainly to different automation equipment like field devices, controllers and similar), focusing mostly on the functionality they offer at the shop floor. Therefore any functionality could be represented as a host of services offered by the device itself.

Integration of the devices on the functional level allows us to focus on orchestrating services based on their role in a process, and not the device per se. Devices can host a variety of services needed at different level of manufacturing units. Collaboration is enhanced among the entities on the shop floor i.e. among the devices, as well as among the devices and any other services offered or consumed at different production system level. This leads to the minimization of isolation islands of heterogeneous devices and boosts interoperability [7]. However the whole approach poses some significant challenges as well. Understanding the semantics as well as evaluating them against a specific context is needed while communicating vertical across different layers as well as horizontal on the same level. For example temperature control in a coffee maker and that of an industrial boiler is characterized by different operational and application dependent characteristics and consequently would need different severity of attention.

The devices that host web services might have their functionalities implemented using their proprietary tools in different languages. In case control and field level get populated with web services, enterprise services could easily get critical information that might be very valuable for reporting and performing manufacturing analytics. Another interesting issue is the support of the eventing in all web service enabled layers ranging from device up to enterprise services [4]. By supporting an event instead of a pull infrastructure, we are able to minimize traffic, while tailored fine grained solutions can be realized. First at all this reduces the effort and network traffic related to management applications like production or asset management. Influences from using eventing for distributed control are currently investigated.

While introducing new concepts it is essential to consider the integration of components already existing or being based on alternative and well established technologies. For several reasons like investments, skills of persons, or others it is the usual case not to replace complete installations. These are extended using the same or improved technologies. The integration of non web service enabled automation components is done using gateways or mediator, as done in the case described hereafter.

We demonstrate the above ideas in a real time production prototype, a test rig, which checks for red and black tokens from a magazine, is considered. The rig performs four different operations before the final product is stored in its respective magazine. The entire rig is controlled by a Siemens S7-300 Programmable Logic Controller (PLC). On top of this controller is an OPC based interface that provides access to PLC related data (I/O and internal data). This interface is accessed from a Mediator (through an OPC client). This Mediator currently supports DPWS to offer production site related services to the SAP ERP System. It is intended to enhance the Mediator to provide an OPC UA conformant interface.

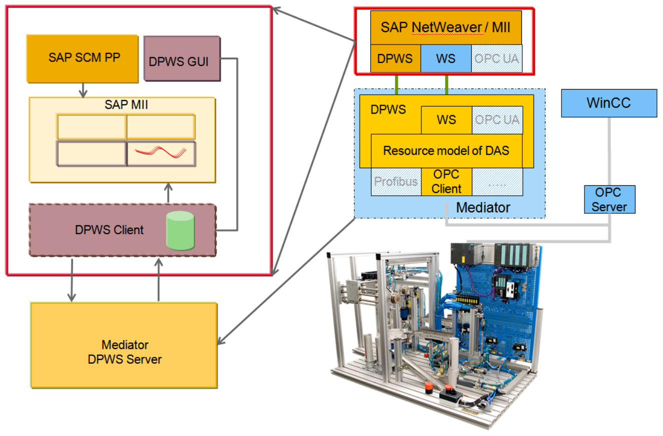


Figure 1. Architecture Overview

2. Architecture

The SAP Supply Chain Management (SCM) Production Planner (PP) system supports manufacturing by optimizing throughput times and bottleneck capacities using new scheduling processes (supply optimization) and by achieving online integration of production planning and control activities. This provides a transparent overview of the entire order network. To have an optimized schedule in the production plan, the SAP SCM PP needs data about the available resources from the shop floor to cross boundary locations, in a globally located manufacturing plant. The quality and validity of the information is far more important to reach the real time simulation of the shop floor resources. To achieve this, SAP has added Manufacturing Intelligence and Integration (MII [8]) as an integration middleware between the Manufacturing Execution Systems (MES) on the Shop Floor and the SAP ERP landscape. MII also offers bidirectional connectivity between the control and field level devices and the ERP system. As a result, the manager could have an overview of which current orders are being executed and what is the status of the executing machine. Moreover intelligent information can be extracted from manufacturing analytics and performance factors provided by the SAP MII.

As depicted in Figure 1, the SAP MII is connected to the Mediator software over a shared database. The database is wrapped via web services that can handle DPWS invocations and subscribe to its events. Those services are provided by the Mediator. An alternative access path using OPC UA is currently being developed. Supervision and control of the PLC is done using SIEMENS SIMATIC WinCC [9], that is connected to the PLC via an OPC interface. The PLC controls the five main modules of the test rig: The loader, distributor, inspector, processor and the repository. Each module has a distinct task to be executed

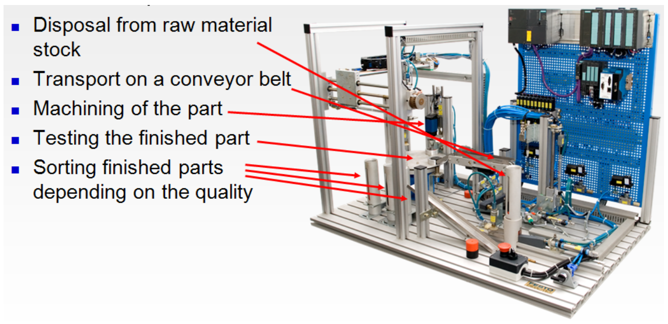

Disposal from raw material stock
Transport on a conveyor belt
Machining of the part
Testing the finished part
Sorting finished parts depending on the quality

Figure 2. Main Operations

and is interconnected to the process flow of the other modules. The Mediator software monitors the process status and hosts routines that control the flow of execution on the PLC. It also exposes three main functions as a web service i.e.:

1. starting the production
2. stopping the production
3. subscribe to events occurring at the test bed

3. The testbed

The test rig consists of a suite of pneumatic based hardware that does operations like picking, placing, moving, drilling, proximity sensing and stocking. The overview of the hardware is pictured in Figure 2. The test rig is supported by a 10 bar compressor and operates in 24 V DC from the PLC power supply unit. It is assumed that it produces tokens which are drilled with a hole in the centre. The tokens are supplied from a magazine which drills the hole, checks its colour and material, and sorts it to the corresponding storage magazine.

The main parts of the test rig (also seen at Figure 3) are:

- Input magazine and distributor: This module consists of a shoving out cylinder operated pneumatically, a pile magazine which stocks the red and black tokens and a swivel arm. As the tokens come down, the shoving out cylinder separates them from the pile magazine and pushes it against a control switch (micro switch). When vacuum is applied, the swivel arm sucks the token and swings it from the position 'distribution' to position 'proofing'. Then the vacuum is switched off so that the token moves to the next module over the conveyor.
- Proofer: The proofer module consists of the proof station, a rejection cylinder and a conveyor. As the token reaches the proof station, a capacitive proximity sensor senses it. This sensor then activates the PLC to read the data that comes from a colour sensor. The PLC registers the colour of the token. The material is also checked by an inductive sensor which energizes at the presence of metals. The third measurement is dedicated to the size of the token. In case the size is not within the limits, the rejection cylinder is activated and it pushes the token into a rejection bin. In case the size is ok, the conveyor is activated and it moves the token to the rotating work table.
- Rotating work table: This module consists of a turn table, a drill press with clamping cylinder and a piston that checks for a hole in the token called the proof cylinder. The rotating table stops at four positions: conduct, boring, checker and dispatcher. The token delivered from the proofer passes through a narrow conduct to the rotating table which moves the token to the drilling press. A hole is then drilled and then the token is passed on to the checker. The piston in the checker moves itself to the centre of the token to prove if there is a hole in it. If there is a hole, then the dispatcher moves the token to the rejection bin. If the token passed this test, then it is rotated to the dispatcher position.
- Dispatcher: The dispatcher reads the colour information from the PLC and calculates the trajectory of the vertical and horizontal motion of the linear actuators. It then picks up the component and moves the actuators towards the storage magazine where the tokens are stored respective to their colour. The dispatcher indicates to the PLC that the token had finished its passage through the production street. This is noticed by the Mediator who updates this information to the SAP MII over web service events along with colour information.

At the end of each operation, there are events generated by the Mediator to SAP MII. Either the successful completion of the operation or the failure of a task is reported.

4. The prototype integration

Our main effort was to integrate via web services the shop floor and the test rig. To that extent, we have used DPWS to connect the enterprise software and test rig via the mediator.

Enterprise level software: The SAP MII software performs the basic interconnection between the shop floor and the ERP landscape. It reads the data sent by the DPWS interface layer over the common database. It also connects to the SAP SCM PP system over the datasource connectors which are provided using BAPI interfaces. SAP MII also hosts a GUI which is a web interface running on NetWeaver Web Application Server (NW WAS) 6.0. The web interface

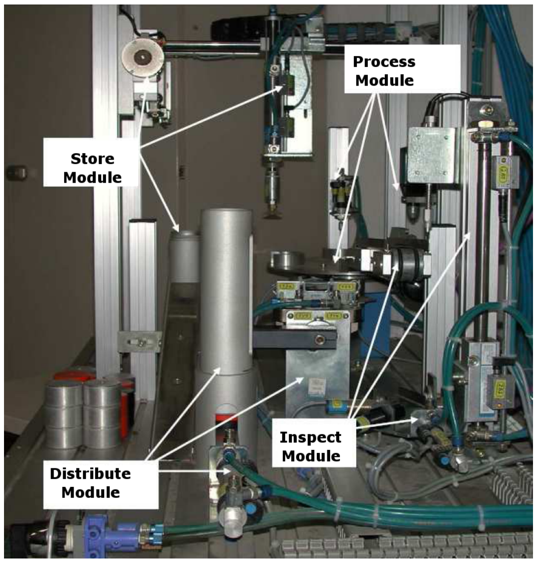
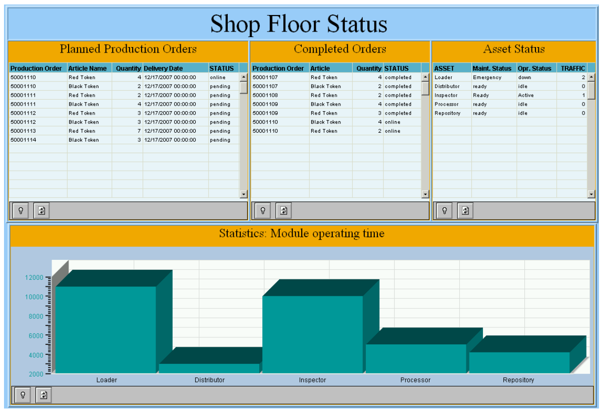


Figure 3. Material flow



Planned Production Orders

Production Order	Article Name	Quantity	Delivery Date	STATUS
50001110	Red Token	4	12/17/2007 00:00:00	online
50001110	Black Token	2	12/17/2007 00:00:00	pending
50001111	Red Token	2	12/17/2007 00:00:00	pending
50001111	Black Token	4	12/17/2007 00:00:00	pending
50001112	Red Token	3	12/17/2007 00:00:00	pending
50001112	Black Token	3	12/17/2007 00:00:00	pending
50001113	Red Token	7	12/17/2007 00:00:00	pending
50001114	Black Token	3	12/17/2007 00:00:00	pending

Completed Orders

Production Order	Article	Quantity	STATUS
50001107	Red Token	4	completed
50001107	Black Token	2	completed
50001108	Red Token	2	completed
50001109	Black Token	4	completed
50001109	Red Token	3	completed
50001110	Black Token	4	online
50001110	Red Token	2	online

Asset Status

ASSET	Maint. Status	Opr. Status	TRAFFIC
Loader	Emergency	down	2
Distributor	ready	idle	0
Inspector	Ready	Active	1
Processor	ready	idle	0
Repository	ready	idle	0

Figure 4. Plant manager's MII view

runs a suite of applets to display the status of the modules in the test rig. The applets wrap SAP MII transactions which run queries in the database. The transactions also analyze the result using a complex business rule which is developed using the MII business logic editor. The rules check which token (red or black) is produced and check against the corresponding production order. If there are more red tokens produced than the ones required in the production order, they are stored and then logically deducted in the next production plan which is already designed and ready to be executed and on the pipeline. This does not increase the production time, rather neutralizes the delay. The MII also shows the status of the test rig modules. Figure 4 presents the snapshot of the SAP MII GUI that is visible to the plant floor manager.

Mediator : The Mediator is used to aggregate various services in SOAs. As such, a Mediator can be seen as a gateway except that it hides (or surrogates) many devices and not just one. However, Mediators go beyond gateways since they introduce semantics in the composition. They aggregate, manage and eventually represent services based on some semantics; in our case used to aggregate various non WS-enabled devices. This way, higher level application could communicate with the Mediator via DPWS, instead of communicating to devices with proprietary or standard fieldbus interfaces. Using a Mediator introduces another level of abstraction and aggregation between the clients and devices. A service mediator is understood to be a device that controls a set of lower-level non-web-service-enabled devices, which it uses to implement a process of which it exposes a service interface. Thus the individual lower-level devices are invisible outside of the mediator. Thus, seen from the outside, there is no essential difference between a service mediator and a composite service that relies on a set of service-enabled devices. The Mediator is build based on a layered approach starting on the base level with device proxies representing individual non-web-service enabled devices. Upon this level, object instances represent compositions of devices like a machine or a part of a plant exposing functionality through a web service interface (in this cases based on DPWS). Below the device proxy level interfaces to access different standard or propriatory communication channels are supported. In our case OPC is used to access the test rig. The Mediator monitors data within the Siemens S7-300 PLC using the OPC interface. It also controls the start and stop operations of the PLC. The Mediator exposes the start and stop operations as a web service. These operations can be invoked from the SAP MII software. The mediator retrieves the status information from the SIMATIC software. It propagates this information as web service events through DPWS interface which then write it to the database finally visible to the SAP MII software. The mediator hosts a DPWS server to expose the start, stop and the events as a web service representing operations of the plant. In parallel the WinCC hosts a GUI to display the current status of the operations that are carried out on the PLC.

DPWS Integration: The DPWS integration layer consists of a client that discovers and subscribes to the server hosted on the mediator gateway. This layer of the software builds the SOAP messages to invoke the start and the stop operations of the mediator. The Mediator, upon receiving these SOAP requests, issues commands to the OPC interface of the PLC to start the pneumatic pusher that moves a token from the input magazine to the conveyor belt. The DPWS layer also de-constructs the SOAP messages from the status event and writes the data to the database over the JDBC interface. The status event is pre defined to a set of error codes which represent the health of individual modules and the results of the operations done by them. SAP MII reads these events from the database and executes a transaction that verifies the event and decides what could be the next operation that the test rig has to execute. The DPWS layer reads data from the database periodically. This is done to execute the commands of the SAP MII which writes the results of the business intelligence into the database.

5. Final thoughs and conclusions

Deploying web services on devices (Mediator represents devices close to the process) in the shop floor e.g. a PLC operation and connecting it to enterprise systems was presented. Via the web services backend systems are able to subscribe to events and take advantage of real-time information flow e.g. for optimization of production planning or reaction to unexpected changes. The clear advantage of pushing SOA concepts down to mediator level, is that business application developers can design and implement new functionality in enteprises without focusing on the devices but on the services they provide. As such another abstraction layer based on well-known and used web service standards will ease the integration of backend and shop-floor systems. The scenario depicted in this paper is a proof of concept for this easier and tighter integration. The results show that the dynamic nature of the shop floor can be utilized efficiently to plan further production orders and even implement last minute changes on the production line using real time data (real time reconfiguration based on the application needs). As all higher level communication is done via web services (in our case DPWS) it is easy for other entities (whether they are services or devices) to subscribe and get the necessary info while in parallel being agnostic to the actual implementation details. This greatly increases interoperability and reduces costly integration time. In our next steps we plan to further work on better web service integration, also with the usage of OPC UA as well as improvements regarding fulfillment of typical industrial automation specific requirements like real-time constraints, reliability, safety or security. Special attention will be paid to improve easy configuration of integration components like gateway or mediator based on device description technology known within industrial automation. This will pave the way for auto-configuration on the gateway and mediator level and enhance integration of legacy systems into higher level management applications.

6. Acknowledgments

The authors would like to thank the European Commission and the partners of the European IST FP6 project "Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices" (SOCRADES - www.socrades.eu), for their support.

References

[1] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation", IEEE Transactions on Industrial Informatics, 1:62-70, 2005.

[2] Manufuture: A vision for 2020, Report of the High Level Group, November 2004, European Commission, Brussels, Belgium, ISBN 92-894-8322-9 `http://www.forum-manufuturep.org/documentos/manufuture_vision_en.pdf`

[3] Stamatis Karnouskos, Oliver Baecker, Luciana Moreira Sa de Souza, Patrik Spiess, "Integration of SOA-ready Networked Embedded Devices in Enterprise Systems via a Cross-Layered Web Service Infrastructure", 12th IEEE Conference on Emerging Technologies and Factory Automation, September 25-28,2007, Patras, Greece

[4] Luciana Moreira Sa de Souza, Patrik Spiess, Moritz Koehler, Dominique Guinard, Stamatis Karnouskos, and Domnic Savio, "SOCRADES: A Web Service based Shop Floor Integration Infrastructure", Internet of Things 2008 Conference, March 26-28, 2008, Zurich, Switzerland.

[5] J. Schlimmer, et al, "Devices Profile for Web Services", Feb. 2006, `http://specs.xmlsoap.org/ws/2006/02/devprof/DevicesProfile.pdf`

[6] OPC Unified Architecture (OPC-UA), OPC Foundation `http://www.opcfoundation.org/UA`

[7] Microsoft Embedded CE 6.0R2: Beckhoff automation help with introduction, Control Engineering `http://www.controleng.com/article/CA6508859.html`

[8] SAP Manufacturing Integration and Intelligence (MII), `http://www.sap.com/solutions/manufacturing/manufacturing-intelligence-software`

[9] SIEMENS SIMATIC WinCC, `http://www.automation.siemens.com/hmi/html_76/products/software/wincc`