Self-organizing Service Structures for Cyber-physical Control Models with Applications in Dynamic Factory Automation
A Fog/Edge-based Solution Pattern Towards Service-Oriented Process Automation

Maximilian Engelsberger and Thomas Greiner
Institute of Smart Systems and Services (IOS3), Pforzheim University, Tiefenbronnerstr. 65, Pforzheim, Germany

Keywords: Dynamic Service Management, Service Orchestration, Service Placement, Service Embedment, Bubble Model, Edge/Fog Computing, Factory Automation, Cyber-physical Production Systems, IT/OT Convergence, Industry 4.0, IoT / IIoT.

Abstract: The convergence of information technology and operational technology is a strong force in fabric automation. Service-oriented architectures and cloud computing technologies expand into next generation production systems. Those technologies enable a lot of new possibilities; such as high agility, global connectivity and high computing capacities. However, they also bring huge challenges regarding flexibility and reliability through increasing system dynamics, complexity and heterogenity. New solution patterns are needed to conquer those challenges. This paper proposes a new fog-oriented approach, which shows how future production systems, that are often called cyber-physical production systems, can deal with dynamically changing services and infrastructure elements. The goal is to provide an adequate degree of flexibility and reliability across the whole production lifecycle. Therefore, an event property model ("bubble model"), a multi-criterial evaluation metric and extensions to Kuhn-Munkres and Add algorithm are described. The overall concept is evaluated by an application example from the field of process engineering. With the help of practical case studies and dynamic system simulations, qualitative results are gained.

1 INTRODUCTION

Service-oriented architectures (SOA) and cloud computing technologies expand into next generation production systems (Broy, 2010). As part of a bigger transformation process, information technology (IT) and operational technology (OT) converge. This generates a wide range of new possibilites within factory automation, which is traditionally very static, isolated and often based on proprietary technologies (Kowalewski, 2015). With a wide use of IT in fabric automation, a lot of standards and quasi-standards can be used. Thereby, a central technology is the Internet Protocol (IP) family (Mor et al., 2016). This leads to completely new possibilites like better system agility, higher vertical connectivity, the use of huge computing capacities and storage space etc. (Ruggerie et al., 2016). This transformation, which is frequently called Industry 4.0, will finally result into a lot of new business cases. Nevertheless, such a process of change brings up a lot of risks to an application domain, which is naturally depending on high reliability and robustness. Eminent factors that endanger reliability are increasing system dynamics, complexity and heterogenity. This ranges from small sensor nodes to big datacenters and from tiny PID-controllers (Proportional-Integral-Derivative) to huge optimization algorithms. In cyber-physical production systems (CPPS) all physical processes and structures are mapped to virtual control models, which represent and control state and behavior of those entities (Alur, 2015). These models may be encapsulated into services, which need to be managed by the system. Especially, all questions of service orchestration and infrastructure utilization must be automatically reconfigured under dynamic changes. Those might be

- Changes on functional/non-functional elements
- Changes on infrastructure and network
- Changes on sensors, actuators and smart objects

If IT will be useful, it must provide answers to the biggest cost factors in factory automation throughout the complete lifecycle of a production plant: Engineering efforts, production stops, maintenance etc. Therefore, it is meaningful to automate as many of the technical low-level tasks assigned to this costs, as

Engelsberger, M. and Greiner, T.
Self-organizing Service Structures for Cyber-physical Control Models with Applications in Dynamic Factory Automation - A Fog/Edge-based Solution Pattern Towards Service-Oriented Process Automation.
DOI: 10.5220/0006365502660274
In *Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER 2017)*, pages 238-246
ISBN: 978-989-758-243-1

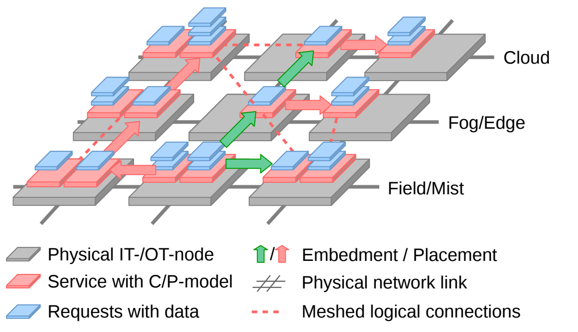


Figure 1: CPPS system topology showing physical and logical structures of services and infrastructure.

possible. From a service-oriented point of view, many of those event-triggered tasks can be summarized by the following questions:

- Where is data acquired?
- Where can data be processed?
- Which requirements need to be fulfilled?
- Where is the processing output needed?

Thus, new design patterns for production systems are needed, answering those questions for all service-based control models in a situation-aware manner. Mechanisms of self-adaptation must orchestrate the services utilizing the cloud, the fog and the local operational field. The goal is to automatically re-arrange the service structure on the system's physical computing infrastructure, in order to reach the needed flexibility and reliability over the whole system lifecycle. The contribution of this paper is a new fog/edge-oriented approach, showing how future production systems can deal with a wide range of dynamic events. The method described here presents an advanced variant of the approach described in (Engelsberger and Greiner, 2017), now revised from a service-oriented field of view and extended by partial corrections and improvements. In section 4.1, an event property model is proposed, which allows to characterize different events in a CPPS by a set of different properties. Further, in section 4.2, a multi-criterial matching metric is described, which allows to evaluate functional and non-functional properties of service-based control models. Further, in sections 4.3 and 4.4, two algorithmic extensions are described, realizing dynamic service operations in order to operate against disturbing run-time events. Finally, in sections 5 and 6, the concept is evaluated by practical case studies from the field of process engineering and evaluated through dynamic system simulations, where quantitative results are gained.

2 RELATED WORK

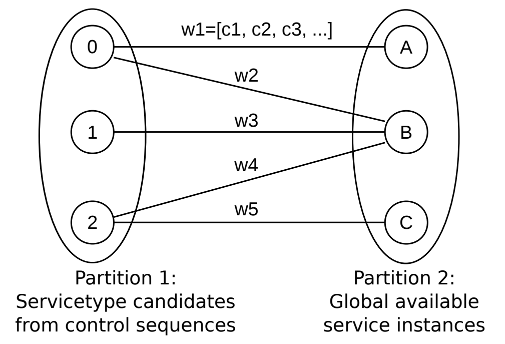
The service-oriented management of IT-resources is a well known concept in cloud computing. It brings a lot of flexibility in terms of scaling the providing service resources to its current needs by utilizing the available (virtualized) hardware infrastructure (Buyya, 2010). Fog/Edge computing techniques dissolve the borders between local and remote cloud computing resources (De Meer et al., 2006). Existing approaches are limited to classical IT-resources, like storage space, computing capacity and bandwith etc. (Dubois et al., 2016), (Wang et al., 2015). They do not consider the special properties and requirements of CPPS like:

- Field process dynamics
- Real-Time- and Safety-Criticality
- Heterogeneous infrastructure and services
- Sensor- and actuator properties
- Cyber-physical model properties

There are also many approaches using load balancing techniques for increasing reliability and stability (Oueis et al., 2015). In load balancing, there are working jobs (loads) which are distributed to currently available worker nodes (Oueis et al., 2015). In contrast to load balancing, the proposed method does not only handle dynamic loads, but creates and optimizes the service-structures underneath those working packets, see Fig. 1. Further, the presented topic has of course a strong relation to the field of service orchestation (Jain et al., 2016). Most methods in this field consider dynamic changes in the services or IT-infrastructure, but they do not offer suitable solutions to dynamic events in the OT-infrastructure, like from sensors, actuators or smart objects. In contrast to that, the described approach allows to model and handle functional and non-functional, CPPS-relevant service-resources, as described in the resource type model (Engelsberger and Greiner, 2016). This model expands the classical service-resource model from cloud computing with the service-resources appearing in a CPPS. An approach for self-organizing service structures for the application in CPPS is described in (Engelsberger and Greiner, 2016) as well. In this approach, service-replication and placement is covered, but a multi-criterial evaluation metric and an approach for dynamic embedment of service-resources is missing. Service embedment allows to evaluate the most suitable service instance, which currently exists in the system, in order to perform a certain task, with respect to actual requirements and run-time properties. Service placement allows to (re-)arrange the services itself, if a task's requirements are not longer fulfilled, on a given node. The missing model of different event properties and their combination are limitations of earlier approaches in this field (Engelsberger and Greiner, 2017). Some of the existing work focus on techniques, which are described as service placement in this work: (Mor et al., 2016), (Oueis et al., 2015), (Wang et al., 2015). Other authors focus on techniques, which are described as service embedment in this work: (Taherkordi and Eliassen, 2014), (Bosman et al., 2015), (Zhang and Cai, 2010). A comprehensive method, combining embedment, placement and a multi-criterial evaluation metric as a comprehensive solution pattern for CPPS is still not visible.

3 BASIC ALGORITHMIC APPROACHES

Because of the flat and decentral organizational structure of CPPS, meshed service-structures evolve. These services and their interconnections can be modeled and manipulated effectively by the use of graph theory. For different management operations, there exist algorithmic approaches.
The main task performed by the service embedment is to match the requirements of a set of given process control sequences to the actual properties of a set of currently available service instances. Such n:m-shaped matching problems can be solved by the Kuhn-Munkres or Hungarian algorithm, which is well known in graph theory (Thulasiraman et al., 2016). It solves a combinatorial optimization problem, by maximizing the matching - or sum of selected edge-weights - between two bipartite sets of vertices. That edge-weights allow to assign arbitrary metrics between each possible two-set of vertices. Because of the bipartite structure of the graph, which is a must for the algorithm, it is not possible to have more than one edge between one possible pair from partition 1 and partition 2, see Fig. 2. Thats why another approach is needed, in order to apply the proposed multi-criterial metric on the given single edge-weight graph structure.
Within service placement, the vertices of a meshed service-structure has to be placed on a meshed node-structure of the physical infrastructure. There are already existing methods to place groups of providing services on groups of consuming services, in a way, that the overall costs are minimized. This is equivalent to the p-Median problem (Krishna, 2014). One possible solution method is the Add algorithm, which needs an input data structure in form of *p* providers and *q* consumers (Domschke and Drexl, 1996). Because of the flat and decentral structure of a CPPS, services are often providers and consumers at the same time. Hence, there is no fixed hierarchical structure between the services, but a meshed one, which changes its structure over time. This problem structure is not directly applicable to existing methods, like the Add algorithm. Thats why a new approach is needed, which analyzes the given service structure and transforms it in a form, where already existing solutions can be applied.


w1=[c1, c2, c3, ...]
w2
w3
w4
w5
0
1
2
A
B
C
Partition 1: Servicetype candidates from control sequences
Partition 2: Global available service instances
Figure 2: Combination of bipartite graph matching problem with multi-criteria matching metric.

4 IMPROVED APPROACH FOR SELF-ORGANIZING SERVICE STRUCTURES IN DYNAMIC FACTORY AUTOMATION

The following subsections explain the improved approach for self-organizing service structures in dynamic factory automation. In 4.1, a new event property model is described. In 4.2, the multri-criterial matching metric is introduced. The metric is used in 4.3 for the extension of the Kuhn-Munkres algorithm. Finally, in section 4.4, the extension of the Add algorithm for the use of meshed service structures is described.

4.1 Event Property Model

In CPPS, dynamic events may occur during the factory's run-time. In the described event property model ("bubble model"), run-time starts with the first module going in operation and ends with the last module, going out of operation. This includes all events during production phase, as well as functional/non-functional changes and maintenance. During production-phases, the sensitive modules and their downstream dependancies are locked against dynamic management operations in order to

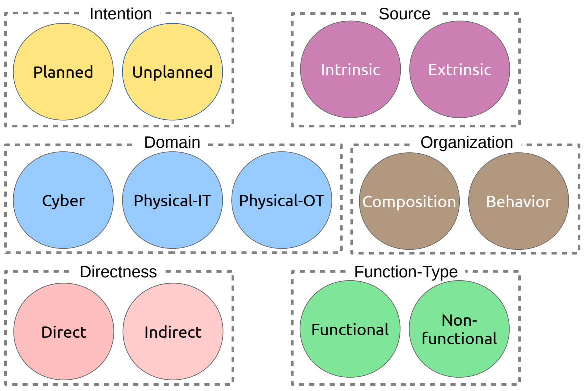


Figure 3: Event property model (“bubble model”) with relevant properties.

not endanger running charge-processing. Only on emergency exceptions, it is possible to remove the management locks: (A) To prevent process-faults or (B) to minimize time to get back to a working processing. The model itself allows to characterize different dynamic events, from various sources, in a formalized way, see Fig. 3. In the following, certain properties and assigned valid values are given, followed by a short explanation:

- **Intention**: An event may be planned or unplanned
- **Source**: An event may be triggered system-intrinsic or -extrinsic
- **Domain**: An event may occur in the Cyber-, Physical-IT- or Physical-OT-domain
- **Organization**: An event may be described as compositional or behavioral
- **Directness**: An event may be triggered direct or indirect by other events
- **Function-Type**: An event might be of functional or non-functional type

With the given event property model, it is possible to characterize and compare different run-time events by their type, domain, intention etc., by merging single bubbles to complex structures. This can be used to construct a complete set of proto-cases, showing all principal types of events, which the proposed method can deal with. The model is applied to the case studies, discussed in section 5.

4.2 Functional- and Non-functional Multi-criterial Metric

In this section, a multi-criterial evaluation metric for functional- and non-functional pairs of requirements and properties is described with some improvements and corrections to (Engelsberger and Greiner, 2017): First, a set of matching criteria $c \in C$ is defined as the following 4-tupel:

$$c = (\delta, \mu, \psi, \varphi) \tag{1}$$

In the following, the parameters from the 4-tupel $c \in C$ are defined: Given deviation

$$\delta \in \mathbb{R} | \delta >= 0 \tag{2}$$

describing the deviation between reference requirement

$$r_{ref} \in \mathbb{R} | r_{ref} > 0 \tag{3}$$

and actual property

$$p_{act} \in \mathbb{R} | p_{act} >= 0 \tag{4}$$

of a given pair of sequence and service instance. With the following meanings:

- $\delta = 0$: no fulfillment
- $(\delta > 0) \wedge (\delta < 1)$: partial fulfillment
- $\delta = 1$: full fulfillment
- $\delta > 1$: overfulfillment

Next, given a mandatory flag

$$\mu \in \{0, 1\} \tag{5}$$

In positive logic, 0 means not mandatory and 1 means mandatory. Given

$$\psi \in \mathbb{R} | \psi >= 0 \tag{6}$$

defining the priority of the given criteria. Next, given a flag

$$\varphi \in \{0, 1\} \tag{7}$$

defining if the given criteria is functional (with value 1) or non-functional (with value 0). In the following steps, the calculation of the service matching metric $SMM(C)$, is described. In earlier work, an alternative metric is named resource matching metric $RMM(C)$, see (Engelsberger and Greiner, 2017). For instance, the relative deviation of the actual metric value from the reference value is determined:

$$\delta = \frac{p_{act}}{r_{ref}} \tag{8}$$

As next step, the priority and mandatory-flag, which are assigned to one criteria, are taken into respect:

$$\zeta = \begin{cases} \delta \cdot \psi : (\mu = 1 \wedge \delta >= 1) \vee (\mu = 0) \\ 0 : else \end{cases} \tag{9}$$

Therefore, the interim value ζ is the product of the deviation δ and the priority ψ, if the mandatory-flag

μ equals 1 and the deviation δ is equal to or greater 1. This means if the mandatory flag is set, consider the deviation only, if the requirements are fulfilled or over-fulfilled ($>= 1$). Otherwise set ζ to zero. If mandatory flag isn't set, consider the deviation, independent from its actual value. Next, the interim value is normalized to the maximum value of all given criteria:

$$\zeta_{norm} = \frac{\zeta}{\zeta_{max}} \tag{10}$$

As a final step, the service matching metric $SMM(C)$ over all $c \in C$ is determined:

$$SMM(C) = \sum_{n=1}^{N=|C|} \zeta_{norm} \tag{11}$$

The resulting metric $SMM(C)$ is used in the subsequently described algorithmic approach.

4.3 Extension of the Kuhn-Munkres Algorithm With a Multi-criterial Metric

This section describes the extension of the Kuhn-Munkres algorithm with the multi-criterial metric from section 4.2. The goal is to have multiple criteria for a single matching between control-sequences and service instances, as described in 3. In the following paragraphs, the relevant CPPS architecture elements are explained: A possible factory concept consists of modular electro-mechanical production units, which are networked via Ethernet (IEEE 802.3). Each of the modules is equipped with its own Process Control Model (PCM), including a formalized set of control sequences, see (Engelsberger and Greiner, 2017). In order to execute a specific sub-production task, a set of requirements is assigned to each control sequence. An example sub-production task might be "fill media reactor I with 1500*ml* of liquid media type 112 from inlet *A*". Further, service-instances, which are also named cyber-resources in (Engelsberger and Greiner, 2017), exist within the global scope of the CPPS. A set of functional and non-functional properties are assigned to each of that service instances. The task, which dynamically evaluates the best-matching service instance to each of the existing control sequences, is called embedment operation. Therefore, a bipartite graph structure $G(V,E)$ has to be defined. "A graph G is called bipartite, if the number of vertices in V is the union of two disjunct sets S and T, in such way, that each edge in the set of E has exactly one ending vertex in S and one vertex in T"(Teschl and Teschl, 2006) and (Engelsberger and Greiner, 2017):

$$G = G(S \cup T, E) \tag{12}$$

In the described context, the vertices of set S are control sequences with assigned requirements and the vertices of set T are service instances with assigned properties. Edges between a certain set of two vertices from T and S represent one possible matching. The aggregated metric $SMM(C)$, containing its individual criteria values, is now assigned to each of the existing edges. In order to preserve correct functionality of the matching algorithm, it is important not to violate the rules of the bipartite graph structure, like it was defined above. The proposed approach fulfills this requirement. After executing the extended method with the bipartite graph input data structure, the maximum matchings between the sequences and instances are known. Now, the modules can connect to the service instances and send working packets to it in order to solve sub-processing tasks.

```
while control != abort do
   for all sequence controls SC do
      G.addVertex(sc);
      S.addVertex(sc);
        for all service instances SI do
             G.addVertex(si);
             T.addVertex(si);
   for all sequence controls SC do
      for all service instances SI do
         SMM = 1- SMM(si, sc);
         e1 = G.addEdge(si, sc);
         G.setEdgeWeight(e1, SMM);
matching = KuhnMunkresMinimalMatching(G,S,T);
```

4.4 Extension of the Add Algorithm to Meshed Service Structures

This section describes the extension of the Add algorithm in order to make it able to handle meshed service networks as input data structure. This is done by a series of additional isolation- and weighting-steps of a given service-structure. The idea of the so called placement operation is to optimize the requirements-fulfillment of a CPPS by a dynamical placement or rearrangement of unconstrained service instances on the physical infrastructure. Service-instances are unconstrained, if they havn't any physical dependancies to a certain node, like specific I/O-devices, real-time constraints etc. Typical unconstrained service instances might be - but not limited to - tasks like analyzation, optimization, archiving, monitoring etc.

The problem is equal to the p-Median problem, where $p \in \mathbb{N} \setminus \{0\}$ providing services are placed in such a position to $q \in \mathbb{N} \setminus \{0\}$ consuming services, that the resulting distance costs are minimal (Laporte et al., 2015). As definition for the distance costs the multi-criterial matching metric from section 4.2 can be

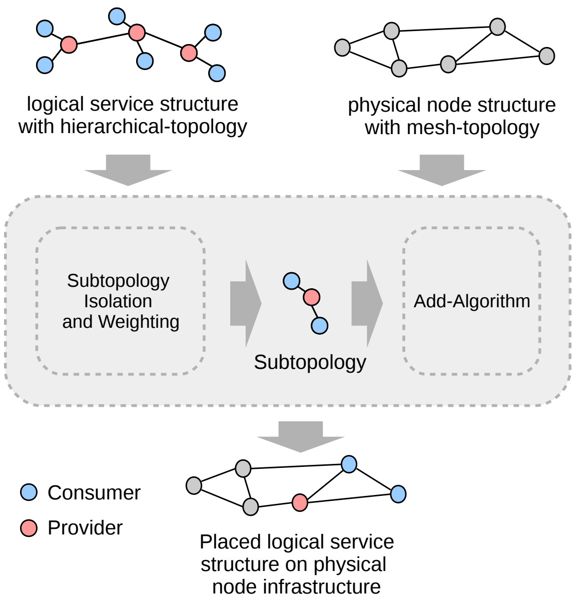

logical service structure with hierarchical-topology
physical node structure with mesh-topology
Subtopology Isolation and Weighting
Subtopology
Add-Algorithm
Consumer
Provider
Placed logical service structure on physical node infrastructure

Figure 4: Add algorithm with subtopology isolation and weighting extension.

used with appropriate criteria. The metric allows to define QoS-specific (non-functional) parameters like actual traffic, latency, jitter, timing-constraints etc., as well as functional parameters like sensor-accuracy, actuator-degeneration-criteria and so on. As described in (Engelsberger and Greiner, 2017), existing solution methods are able to solve the single-stage p-Median problem, when a set of possible locations for the providing services and a set of consuming services with fixed locations are given. As explained, the difference to this concept is, that in a realistic SOA-based production system, certain services are provider and consumer at the same time. An example might be an agitation-optimization service, which consumes data from a media-temperature service and provides data to the agitator service. If this principle is expanded to a complete production-system, this results in a complex meshed service-structure, like in Fig. 4, section 4.4. The Add algorithm needs an input data structure in the form of p providers vs. q consumers, so the meshed service structure needs to be analyzed and transformed in an appropriate way. The following algorithmic extension is proposed:

In **step 1**, the isolation and weighting process is done by a graph-based breadth-first seach, which identificates the service instances without dependancies. The found services are sorted by their placement priority.

In **step 2**, their dependancies are determined. With this information, a new subtopology is built. The subtopologies are used to generate a reference cost matrix, by aggregating the connection requirements. The reference cost matrix is defined as a single-row matrix $R^N_{res,x}$ with:

$$r_i \in [0, 100] \tag{13}$$

In **step 3**, the costs of all physical connections in the network are determined. These connection costs are stored in the actual connection cost matrix $A^{N\times M}_{pnet}$ with:

$$a_{ij} \in [0, 100] \tag{14}$$

In **step 4**, the difference costs between $R^N_{res,x}$ and $A^{N\times M}_{pnet}$ are calculated. Therefore, a fitting function is used, so all resulting values are within

$$d_i \in [0, 100] \tag{15}$$

The fitting function is defined as

$$d_{ij} = -\frac{1}{2} \cdot (r_i - a_{ij}) + 50 \tag{16}$$

The result is the difference cost matrix $D^{N\times M}_{costs}$ with:

$$d_{ij} \in [0, 100] \tag{17}$$

The difference cost matrix is a valid input data structure for the Add algorithm, as shown in the following algorithm:

```
while allvisited != true do
    V2V_queue += V_start;
    BreadthFirstSearchIteration(G_srv,
    V2V_queue, V_actual);
  if !(IOconstrained(V_actual)) then
    Unconstr_queue += V_actual;
  for all V in Unconstr_queue do
    CalculateQoSPriority(V);
    SortByQoSPriorityDesc(Unconstr_queue);
  for all V in Unconstr_queue do
    if !(PartOfPlacedSubStruct(V)) then
      G_subset = getWeightedSubset(G_srv,
      G_phys, V);
      placement = Add(G_subset);
```

The extended method is now able to place services from meshed structures in such way on the physical network nodes, that the connection costs are minimized with respect to the service priority.

5 APPLICATION CASE STUDY

This section picks up the CPPS system description from 4.3 and describes the derived case studies. The CPPS testing facility itself is described in detail in (Engelsberger and Greiner, 2016). It is a modular and decentral system architecture with very heterogeneous system nodes, from very small real-time controllers in the field to very huge IT-resources in the

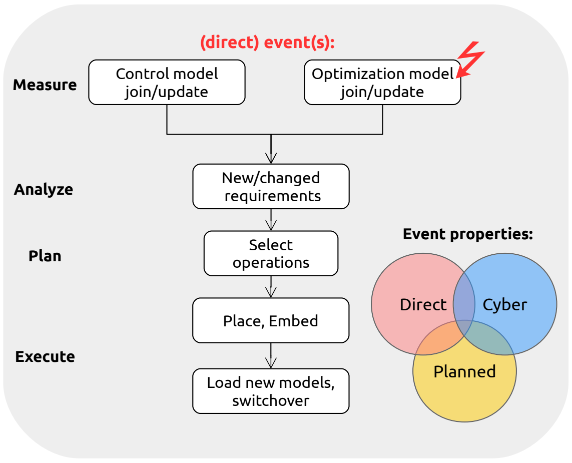


Figure 5: Example 1: Agitation model updates.

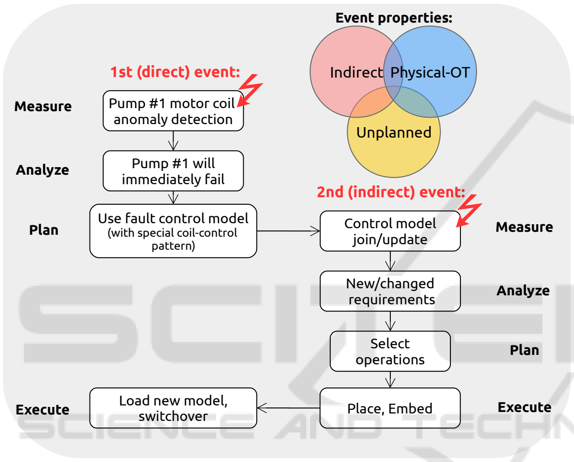


Figure 6: Example 2: Motor coil anomaly detection.

local fog and in the cloud. The described application scenario comes from process engineering and represents a production process of liquid media. Fig. 7 shows a cut-out of this process with two processing stages: The first is the boiling reactor, which is encapsulated in a production module. It is able to heat-up liquid media to a given reference temperature. The second is the agitation module, which is able to agitate the media to a given reference viscosity. These modules depend on field devices like networked sensors, actuators and smart objects. Corresponding to the event property model in 4.1, this group of devices is summarized as physical OT domain. As described in section 1, such systems are getting more and more complex and fragile with size and number of dynamic changes. In the subsequent sections, two examples are given, what may change during run-time and how the method can handle those dynamic events.

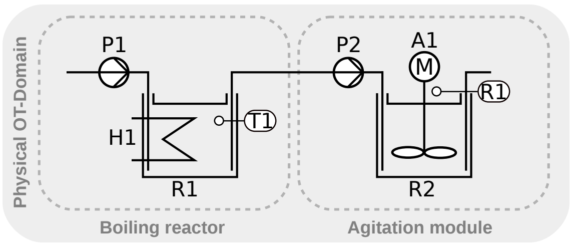


Figure 7: Relevant cut-out of the production process including boiling reactor and sediment removal whirlpool.

5.1 Example 1: Agitation Model Update

The first example focuses on the agitation module. Two parallel occuring events of the same type are taken into account: (A) A control model join/update and (B) an optimization model join/update, see Fig. 5. Both are part of the engineering or maintenance process, happening during lifetime of the facility, but not necessarily during a running operation. The control model of a CPPS-module represents the control sequences, needed to control a certain aspect of the system: The agitation unit, in this case. The control model is encapsulated as a software service and interacts with other services in the system-wide meshed service-structure. The optimization model is a service as well. In this case it is supposed to compute and optimize the operational parameters for the agitation control model, taking into account actual conditions. This could be the actual media temperature, which is served by another service from the boiling reactor. Characterized by the event property model in 4.1, both events are of intention “planned”, by their engineering character. Further, they are of direction “direct”, because their update is the immediate cause for the event. Finally, the event source is the “cyber-domain”, because they are triggered by model code changes and not by any physical resources.
If the models change, so their requirements change and it might become necessary to perform some service-structure changes. In the given example a re-arrangement of the place, where the service-models are located, might increase the requirement fulfillment. The control model needs to be placed on a node, where all real-time-requirements are fulfilled, in order to control the agitation unit. The optimization model needs to be placed on a node, with a certain computing capacity, in order to process the optimization algorithm in an appropriate time. Finally, the dynamic operations, calculated by the algorithm, are deployed on the system and the switchover to the new service configuration can be done.

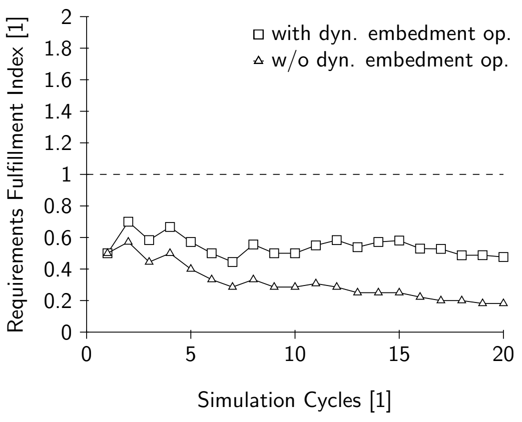


Figure 8: Results from the simulation-run with service embedment algorithm based on a multi-criterial metric.

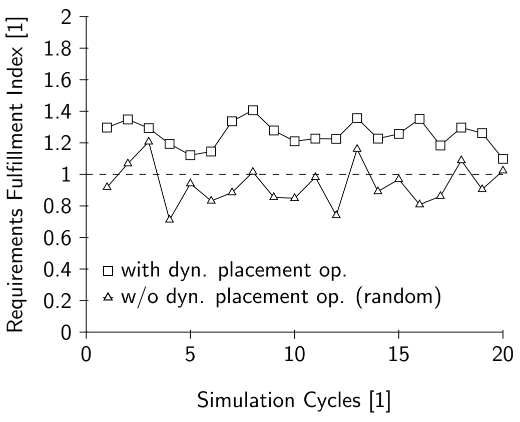


Figure 9: Results from the simulation-run with service placement algorithm based on a multi-criterial metric.

5.2 Example 2: Pump Motor Coil Anomaly Detection

The second example focuses on the agitation module, as well. One direct source event occurs: An anomaly of media pump #1's motor coil is detected, see Fig. 6. The sensor says pump #1 will immediately fail. As an reaction it is assumed to be possible to switchover to a fault control model. The selection of the fault control model triggers a second (now indirect) event, which is to update the control model. This update brings up new requirements, which the system needs to fulfill. The event is of "intention"unplanned, because no one decided to produce this motor coil anomaly. It is of directness "indirect", because a first source event triggers a second model join event. In this case, the newly joined model needs to be placed on the system, depending on its requirements. Further, the newly joined model needs to be embedded into the other services, in order to use it. Finally, the dynamic operations, calculated by the algorithm, are deployed on the system and the switchover to the new service configuration can be done.

6 SIMULATION AND RESULTS

In this section, the previously described metric, algorithms and use-case-scenarios are transformed into software simulations, in order to evaluate the concepts against non-adaptive service-structures. For this purpose a simulation environment is written in Java, where it is possible to describe infrastructure and services from IT- and OT-domain and to realize models of production-processes via appropriate interfaces. Furthermore, the simulation environment offers interfaces to test and evaluate different metrics and algorithms of adaptive behavior, as well as interfaces to generate dynamic events and/or to collect them from physical infrastructures.
In order to evaluate the embedment operation, 0 to 10 joins and leaves of control sequences and service instances, as well as relevant properties and requirements are pseudo-randomly generated, within 20 simulation cycles. The results are shown in Fig. 8.
In order to evaluate the placement operation, joins and leaves of control sequences and service instances, as well as changes of the transport costs of the physical network are simulated. Therefore, 0 to 10 joins and leaves of control sequences and service instances, as well as relevant properties and requirements are pseudo-randomly generated, within 20 simulation cycles. The results are shown in Fig. 9.
Both graphs show the Requirements Fulfillment Index (RFI), an evaluation metric which aggregates the requirement fulfillment for all control sequences, across the overall CPPS. The evaluation metric is calculated by the following equation: $RFI = AVG(\sum SMM(C))$ over all managed service instances.

7 CONCLUSIONS

Future cyber-physical production systems (CPPS) are very heterogenous and dynamic across their whole lifecycle. From the technology point of view, IT and operational technology (OT) converge. That makes new IP- and SOA-based technologies accessible in automation applications. As a side-effect, CPPS get more and more complex and fragile. In this paper, a new solution-pattern is proposed, showing how future production systems can deal with disruptive run-time events by dynamic adaptations on service level, utilizing the possibilities of the local operational field, the local fog/edge and the global cloud. A new fog-oriented approach is described, showing how an adequate degree of reliability and flexibility is possible under dynamic changes. For this purpose, an event property model ("bubble model"), a multi-criterial evaluation metric and two extensions to already existing algorithmic approaches are shown, which realize operations of service structure adaptation. The method is tested under realistic conditions by an application example from the field of process engineering. Within two case studies and software simulations, the concept is evaluated and quantitative results are gained. The results show, that both of the described algorithmic extensions are able to extend the RFI against non-adaptive approaches. Further investigations are needed to learn more about mutual reactions between both algorithms and different CPPS-specific cases.

ACKNOWLEDGEMENTS

This work is supported by the Baden-Wuerttemberg Ministry of Science, Research and the Arts (MWK) within the scope of Cooperative Research Training Group.

REFERENCES

Alur, R. (2015). *Principles of Cyber-Physical Systems*. MIT Press.

Bosman, J., van den Berg, H., and van der Mei, R. (2015). Real-Time QoS Control for Service Orchestration. *IEEE 27th International Teletraffic Congress*.

Broy, M. (2010). *Cyber-Physical Systems (acatec DISKUTIERT)*. Springer.

Buyya, R. (2010). *Cloud Computing: Principles and Paradigms*. John Wiley & Sons.

De Meer, H., Sterbenz, J., and EuroNGI. (2006). *Self-Organizing Systems: First International Workshop, IWSOS 2006 and Third International Workshop on New Trends in Network Architectures and Services, EuroNGI 2006, Passau, Germany, September 18-20, 2006, Proceedings*. Computer Communication Networks and Telecommunications. Springer.

Domschke, W. and Drexl, A. (1996). *Logistik: Standorte*. Oldenbourgs Lehr- und Handbücher der Wirtschafts- u. Sozialwissenschaften. De Gruyter.

Dubois, D. J., Valetto, G., Lucia, D., and Nitto, E. D. (2016). Mycocloud: Elasticity through Self-Organized Service Placement in Decentralized Clouds. *IEEE 8th International Conference on Cloud Computing (CLOUD)*.

Engelsberger, M. and Greiner, T. (2016). Application-independent Approach for the Dynamic Management of IT-Resources in Cyber-Physical Systems.

Engelsberger, M. and Greiner, T. (2017). Handling Strategy of Dynamic Resource Events in Cyber-Physical Production Systems by a Multi-Criterial and Multi-Operational Approach. *Industrial Technology (ICIT), 2017 IEEE International Conference on*.

Jain, P., Datt, A., Goel, A., and Gupta, S. C. (2016). Cloud service orchestration based architecture of OpenStack Nova and Swift. *Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on*.

Kowalewski, S. (2015). Cyber-Physical Systems - A UMIC Perspective. Technical report, RWTH AACHEN UNIVERSITY.

Krishna, P. (2014). *Challenges, Opportunities, and Dimensions of Cyber-Physical Systems*. Advances in Systems Analysis, Software Engineering, and High Performance Computing.

Laporte, G., Nickel, S., and da Gama, F. (2015). *Location Science*. Springer International Publishing.

Mor, N., Zhang, B., Kolb, J., Chan, D. S., Goyal, N., Sun, N., and Lutz, K. (2016). Toward a Global Data Infrastructure. *IEEE Internet Computing*, 20(3).

Oueis, J., Strinati, E. C., and Barbarossa, S. (2015). The Fog Balancing – Load Distribution for Small cell Cloud Computing. *Vehicular Technology Conference (VTC Spring)*, 81:1–6.

Ruggerie, M., Malaguti, G., Dariz, L., and Selvatici, M. (2016). *In-Tractor Cloud: A Vision of Service-Oriented System Design - Enabled by High-Speed In-Vehicle Networks for a Safer Task- and Machine Management*. SAE Technical Papers scopus(scholar).

Taherkordi, A. and Eliassen, F. (2014). Models@run.time for Creating in-Cloud Dynamic Cyber-Physical Ecosystems. volume 6. IEEE.

Teschl, G. and Teschl, S. (2006). *Mathematik für Informatiker*. Springer.

Thulasiraman, K., Arumugam, S., Brandstädt, A., and Nishizeki, T. (2016). *Handbook of Graph Theory, Combinatorial Optimization, and Algorithms*. CRC Press.

Wang, S., Urgaonkar, R., Zafer, M., He, T., Chan, K., and Leung, K. K. (2015). Dynamic Service Migration in Mobile Edge-Clouds.

Zhang, Y. and Cai, W.-d. (2010). Criticality-Driven QoS Adaptive Dynamic Resource Management for Distributed and Embedded Safety and Mission Critical Systems. *International Conference on New Trends in Information Science and Service Science (NISS)*, 4.