Chapter 13
Patterns for Self-Adaptation in Cyber-Physical Systems

Angelika Musil, Juergen Musil, Danny Weyns, Tomas Bures, Henry Muccini, and Mohammad Sharaf

Abstract Engineering Cyber-Physical Systems (CPS) is challenging, as these systems have to handle uncertainty and change during operation. A typical approach to deal with uncertainty is enhancing the system with self-adaptation capabilities. However, realizing self-adaptation in CPS, and consequently also in Cyber-Physical Production Systems (CPPS) as a member of the CPS family, is particularly challenging due to the specific characteristics of these systems, including the seamless integration of computational and physical components, the inherent heterogeneity and large-scale of such systems, and their open-endedness.
In this chapter we survey CPS studies that apply the promising design strategy of combining different self-adaptation mechanisms across the technology stack of the system. Based on the survey results, we derive recurring adaptation patterns that structure and consolidate design knowledge. The patterns offer problem-solution pairs to engineers for the design of future CPS and CPPS with self-adaptation capabilities. Finally, the chapter outlines the potential of Collective Intelligence Systems for CPPS and their engineering based on the survey results.

Key words: Collective Intelligence Systems, Cyber-Physical Systems, Patterns, Self-Adaptation, Software Architecture, Systematic Study

---

Angelika Musil, Juergen Musil
Institute of Software Technology and Interactive Systems, TU Wien
e-mail: {angelika,jmusil}@computer.org

Danny Weyns
Dept. of Computer Science, KU Leuven, and Dept. of Computer Science, Linnaeus University
e-mail: danny.weyns@kuleuven.be

Tomas Bures
Dept. of Distributed and Dependable Systems, Charles University Prague
e-mail: bures@d3s.mff.cuni.cz

Henry Muccini, Mohammad Sharaf
DISIM Department, University of L'Aquila
e-mail: henry.muccini@univaq.it,massharaf@yahoo.com

13.1 Introduction

Cyber-Physical Production Systems (CPPS) form a distinct sub-category of the more general family of *Cyber-Physical Systems (CPS)*. Though distinctively focused on production, CPPS, as a member of the CPS family, share many common traits with other types of CPS, such as distributed robotics, autonomous vehicular systems, smart grid, and smart spaces. These common traits include the strong coupling of physical environment and the computing system via sensors and actuators, involvement of humans-in-the-loop, the necessity of coping with a multitude of heterogeneous models (e.g., physical, electrical, mechanical, control), the need of real-timeness, and strong requirements on dependability.

The close relation to the environment, humans-in-the-loop and the complex interplay of the heterogeneous models brings a high level of uncertainty as a critical factor to be taken into account and addressed when designing CPS, and consequently also CPPS. Examples of uncertainty include the unpredictability of human actions, unexpected emergent behavior of the environment (typically stemming from unanticipated interactions among constituents of the environment and the CPS due to the fact that a CPS is an inherent part of the environment it observes and controls), unexpected or faulty interplay between CPS components, and incomplete requirements. The presence of uncertainty makes it difficult to design the complete behavior of a very complex CPPS with guaranteed dependability, as parts of the knowledge required for such a design may only become available at run time.

A viable software-based solution to the problem of uncertainty lies in equipping the system with self-adaptation capabilities. Self-adaptation adds introspective capabilities to the system allowing it to be aware of its internal state and structure, reason about itself and its goals, identify potential problems in its ability to dependably achieve its goals, and adapt itself to cope with the identified problems. Self-adaptation was already introduced in the area of enterprise systems by IBM in 2003 (Kephart and Chess, 2003). Similar concepts are nowadays regularly applied also for instance in cloud computing where an application automatically reconfigures to scale with the current load and to avoid virtual machines that perform badly due to resource sharing. For a guided tour through the history of the field of self-adaptation, we refer the interested reader to (Weyns, 2017).

As outlined in Chapter 1 of this book, the key research question addressing the *modelling of CPPS flexibility and self-adaptation capabilities (RQ C1)* discusses a very relevant topic for CPPS engineers. From the perspective of this research question, this chapter elaborates concretely on effective architectural approaches and best practices to combine self-adaptation mechanisms to handle uncertainty challenges and concerns. Although other chapters of this book also address a CPPS architecture perspective, we focus on the design of self-adaptation capabilities.

In this chapter, we aim at providing insight on how self-adaptation can be used in addressing uncertainty in CPPS. Since there is rather a general lack of knowledge on self-adaptation specifically in CPPS, we take a generalization step and overview self-adaptation related to the larger family of CPS. Since CPS are systems that do not focus on one layer of the technology stack, but their engineering crosses all

layers, self-adaptation mechanisms are also relevant to be considered on all layers. This claim is supported by the results of a recent systematic literature review aiming at assessing state-of-the-art approaches to handle self-adaptation in CPS at an architectural level. The study revealed that, remarkably, 36% of the investigated studies combine different adaptation mechanisms across the technology stack to realize adaptation in a CPS (Muccini et al, 2016). Therefore, this chapter follows this promising architecture design strategy for CPS and focuses explicitly on combinations of different types of adaptation mechanisms that may span various layers within a system. We do so by the means of a systematic literature mapping with the goal to identify recurring adaptation patterns used in addressing uncertainty by self-adaptation. We further relate these patterns to the specific field of CPPS to give an insight on how to exploit self-adaptation to CPPS. Finally, we outline the potential of *Collective Intelligence Systems (CIS)* for CPPS and their engineering based on the study results by presenting three emerging research directions.

The remainder of this paper is organized as follows: Sect. 13.2 introduces background information about uncertainty types, self-adaptation approaches, and collective intelligence systems. In Sect. 13.3 the research questions and research methodology we used are presented. Sect. 13.4 summarizes the method used to conduct the systematic mapping study. The results of the systematic mapping study are presented in Sect. 13.5, followed by a summary of threats to its validity in Sect. 13.6 and a reflection on the results in Sect. 13.7. Sect. 13.8 describes and discusses the three identified adaptation patterns in CPS. We further explore the potential of collective intelligence systems for CPS and CPPS in Sect. 13.9 and Sect. 13.10 summarizes related work. Finally, Sect. 13.11 draws conclusions and outlines future work.

13.2 Background

This section provides a general introduction to uncertainty types in adaptive systems, different adaptation approaches, its purpose and different methods, as well as collective intelligence systems as a promising enhancement to CPS and CPPS architectures.

13.2.1 Uncertainties

When designing CPS the available knowledge is often not adequate to anticipate all the run time conditions the system will encounter (e.g., missing or inaccurate knowledge regarding the availability of resources, concrete operating conditions that the system will face at run time, and the emergence of new requirements while the system is operating). To that end, Garlan (2010) argues that in todays software systems uncertainty should be considered as a first-class concern throughout the whole system life cycle. In the context of adaptive systems, Ramirez et al (2012) provide a

taxonomy for uncertainty that describes common sources of uncertainty and their effect on requirements, design and run time phases of the system. Esfahani and Malek (2013) present an extensive list of sources of uncertainties with examples. Moreover, these authors investigate uncertainty characteristics, i.e., reducibility versus irreducibility, variability versus lack of knowledge, and spectrum of uncertainty. Perez-Palacin and Mirandola (2014) present another taxonomy for uncertainty for adaptive systems based on three dimensions: location, level, and nature of uncertainty. Mahdavi-Hezavehi et al (2016) present a classification framework for uncertainty in adaptive systems, which is based on a systematic review of the literature. This classification is shown in Table 13.1.

Table 13.1 Uncertainty Dimensions (Mahdavi-Hezavehi et al, 2016)

Uncertainty Dimension	Description	Options
Location	Refers to the locale, where uncertainty manifests itself within the whole system.	Environment, model, adaptation functions, goals, managed system, resources
Nature	Specifies whether the uncertainty is due to the imperfection of available knowledge, or is due to the inherent variability of the phenomena being described.	Epistemic, variability
Level/Spectrum	Indicates the position of uncertainty along the spectrum between deterministic knowledge and total ignorance.	Statistical uncertainty, scenario uncertainty
Emerging time	Refers to time when the existence of uncertainty is acknowledged or uncertainty is appeared during the life cycle of the system.	Run time, design time
Sources	Refers to a variety of circumstances affecting the adaptation decision, which eventually deviate systems performance from expected behavior.	Variety of options based on the sources of uncertainty (e.g., abstraction, model drift, etc. for model uncertainty; sensing, effecting etc. for adaptation functions)

One way to deal with uncertainties is to design systems that adapt themselves during run time, when the lacking knowledge becomes available. Adaptive systems are capable of autonomously modifying their run time behavior to deal with dynamic system context, and changing or new system requirements in order to provide dependable systems. However, realizing adaptation in CPS is particularly challenging due to specifics of these systems include the blurring boundaries between the system and its environment, large scale and inherent complexity, the role of end-users, multi-level uncertainty, open-endedness, among others (Bures et al, 2015).

13.2.2 Adaptation

Adaptive systems are capable of modifying their run time behavior in order to achieve systems objectives. Unpredictable circumstances such as changes in the systems environment, system faults, new requirements, and changes in the priority of requirements are some of the reasons for triggering adaptation action in a self-adaptive system. To deal with these uncertainties, an adaptive system continuously monitors itself, gathers data, and analyzes them to decide if adaption is required. Different paradigms for realizing adaptation have been developed. We summarize three paradigms that appeared in the study presented in this paper: architecture-based adaptation, multi-agent based approaches, and self-organizing based approaches. Examples of other adaptation approaches, out of scope of this chapter, are computational reflection and approaches based on principles from control theory.

Architecture-based Adaptation

Architecture-based adaptation (Oreizy et al, 1998; Garlan et al, 2004; Kramer and Magee, 2007; Weyns et al, 2012) is one well-recognized approach that deals with uncertainties at run time. The essential functions of architecture-based self-adaptation are defined in the MAPE-K (i.e., Monitor, Analyze, Plan, Execute, and Knowledge component) reference model (Kephart and Chess, 2003). By complying with the concept of separation of concerns (i.e., separation of domain-specific concerns from adaptation concerns that deal with uncertainties), the MAPE-K model has shown to be a suitable approach for designing feedback loops and developing self-adaptive systems (Weyns et al, 2013a). One well-known architecture-based self-adaptive framework is Rainbow (Garlan et al, 2004). Rainbow uses an abstract architectural model to monitor software system run time specifications, evaluates the model for constraint violations, and if required, performs global or module-level adaptations. Calinescu et al (2011) present a quality of service management framework for self-adaptive services-based systems, which augments the system architecture with the MAPE-K loop functionalities. In their framework, the high-level quality of service requirements are translated into probabilistic temporal logic formulae which are used to identify and enforce the optimal system configuration while taking into account the quality dependencies. Moreover, utility theory can be used (Cheng et al, 2006) to dynamically compute trade-offs (i.e., priority of quality attributes over one another) between conflicting interests, in order to select the best adaptation strategy that balances multiple quality requirements in the self-adaptive system.

Multi-Agent based Approaches

Multi-agent systems belong to a class of decentralized systems in which each component (agent) is an autonomous problem solver, typically able to operate successfully in various dynamic and uncertain environments (Wooldridge, 2001). These agents interact to solve problems that are beyond their individual capabilities or knowledge. Multi-agent systems have features that are key to engineering adaptive systems – specifically loose coupling, context sensitivity, and robustness to failures and unexpected events (Weyns, 2010; Weyns and Georgeff, 2010). Agents are self-contained, goal-directed entities. They get their adaptability from goals. When multiple agents are available, a goal can be achieved by selecting among the agents at run time, for example using negotiation (Fatima et al, 2006), rather than requiring a hardwired design. An agent includes a specification of the situation or context in which it is appropriate or expected to achieve its target goal. A calling agent can simply post the goals it wishes to achieve and select only those agents appropriate to the goal and current processing context: the right agent at the right time in the right circumstances. Similarly, an agent's internal processes are typically associated with a context condition describing the situations in which the process can achieve its specified goal. This means that processes "self select" according to the desired goal and prevailing situation. Goal-directed multi-agent systems eliminate most of the complexity needed for handling failures (Minsky and Murata, 2004). Failures and unexpected events cause the original goal to be reposted and tried again, without the need for explicit exception handling. The goal-directed mechanism will automatically try them until success or ultimate failure.

Self-Organizing based Approaches

Self-organization is a dynamic and adaptive process where a system acquires and maintains structure itself, without external control (De Wolf and Holvoet, 2004). The essence of self-organization is an adaptable behavior that autonomously acquires and maintains an increased order. Self-organizing systems exhibit the following essential properties: increase in order (exhibiting useful behavior), autonomy (absence of external control), robustness (adaptability in the presence of perturbations), and dynamicity (dynamics that handle changes). Self-organizing systems may expose emergent behavior at the global level that dynamically arises from the interactions between the parts at the local level. The engineering of self-organizing systems if often inspired by natural phenomena, for example from biology such as ant behavior and swarms (Di Marzo Serugendo et al, 2006). The principle idea is to exploit the robustness and flexibility of these natural systems as a metaphor for engineering computing systems. As an example, field-based coordination relies on virtual computational fields (e.g., distributed data structures), mimicking gravitational and electromagnetic fields, as the basic mechanisms with which to coordinate activities among open and dynamic groups of application components. This enables components to spontaneously interact with each other via the mediation of

fields to self-organize their activity patterns in an adaptive way (Mamei et al, 2006; Weyns et al, 2008). In a recent paper, Bures et al (2013) propose a component-based approach that exploits principles of self-organization. In this approach, autonomic components dynamically form so called ensembles that share data to organize themselves on the fly. The authors present the DEECo component model, a concrete realization of the approach.

13.2.3 Collective Intelligence Systems

In the last decades, a form of user-contribution-driven web platforms has intensively influenced the way of today's knowledge creation and sharing processes. Today, this kind of software systems is very popular and widespread in use in our daily lives. Well-known examples of such CIS include Facebook[1], Wikipedia[2], YouTube[3], and Yelp[4]. CIS are *socio-technical multi-agent systems* that aim to harness the collective intelligence of interacting human actors by providing a web-based environment for sharing, distributing and retrieving topic-specific information in an efficient way (Musil et al, 2015a). A CIS posses a characteristic system model that is illustrated by Fig. 13.1. It consists of three layers: (1) a *proactive actor base*, (2) a *passive CI artifact network*, and (3) a *reactive/adaptive computational analysis, management and dissemination (AMD) system* (Musil et al, 2015b). Between the layers, the CIS realizes a perpetual feedback loop connecting the human actor base and the reactive computational coordination environment and consisting of two essential phases: aggregation and dissemination (Musil et al, 2015b). In the aggregation phase, the individual actor contributes explicitly or implicitly new content to so-called CI artifacts by performing defined local activities. These CI artifacts store the aggregated information in a defined structure and are part of a passive artifact network. Defined rules of coordination in the reactive and adaptive AMD system govern the processing and analysis of the artifact data as well as the extraction of consolidated information. In the following dissemination phase, the AMD system uses both active and passive dissemination mechanisms to make the actors aware about artifact content changes and overall actor activities in the system environment as well as stimulate subsequent actor interaction. Thus the resulting bottom-up feedback loop constitutes a *stigmergic process* (Heylighen, 2016) enabling indirect, environment-mediated communication and coordination (Musil et al, 2015b). In addition, the stigmergic process enables self-organization which realizes adaptation within the CIS environment.

To support software architects in the design of CIS architectures, Musil et al (2015c) proposed the *architecture framework for collective intelligence systems*

[1] `http://www.facebook.com/` (last visited 01/15/2017)

[2] `http://www.wikipedia.org/` (last visited 01/15/2017)

[3] `http://www.youtube.com/` (last visited 01/15/2017)

[4] `http://www.yelp.com/` (last visited 01/15/2017)

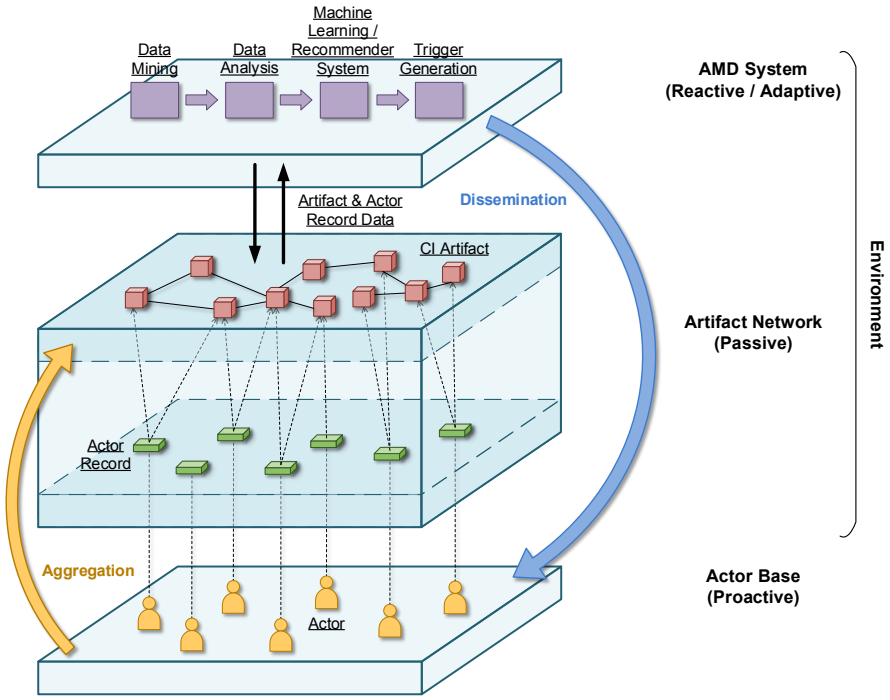


Fig. 13.1 Multi-layer CIS model with three main components and the stigmergic process (Musil et al, 2016a)

(CIS-AF) as one approach that provides guidance for realizing new CIS solutions. The CIS-AF is developed as a methodology to efficiently describe the core elements of a CIS architecture, which are documented in the *Stigmergic Information System (SIS) architecture pattern* (Musil et al, 2015b), without being limited in its technical implementation.

13.3 Research Questions

This chapter basically focuses on contributions to answer *RQ C1 - Modelling of CPPS flexibility and self-adaptation capabilities* specified in Chapter 1 of this book. Concretely, we aim to consolidate existing design knowledge on self-adaptation strategies to address uncertainty in CPPS and to identify novel and promising approaches that need further research. Since there is rather a general lack of knowledge on self-adaptation specifically in CPPS, we broaden the scope of our investigation and provide insight on how self-adaptation capabilities of the more general family of CPS are designed. Thus CPPS engineers can learn from application experiences with CPS for dealing with adaptation challenges and concerns in CPPS.

To address this goal, we identified the following research questions:

RQ1 *- How is self-adaption applied in cyber-physical systems in general?*

We aim to analyze how state-of-the-art approaches make use of self-adaptation mechanisms and models to handle uncertainty while architecting CPS. In addition, we focus on self-adaptation applied in CPS in the manufacturing domain.

RQ2 *- How can this knowledge be applied and exploited to cyber-physical production systems and their engineering?*

Based on a better understanding of existing developed adaptation strategies to address challenges and concerns of CPS, we seek to closely examine common approaches, considerations and advances to identify recurring patterns, models or tactics. The documentation of such architectural knowledge should support CPPS engineers with the realization and coordination of self-adaptation. In addition, this consolidated design knowledge base can provide a strong foundation for designing self-adaptation capabilities in CPPS engineering that can be further researched and extended by CPPS researchers.

RQ3 *- Can principles from collective intelligence systems provide innovation for adaptive CPPS and CPPS engineering?*

CIS are complex adaptive socio-technical systems that apply stigmergic adaptation with humans-in-the-loop. They represent a well-known approach for adaptation used in particular, predominantly social, domains. This research question aims to go beyond the typical application contexts of CIS and to explore CIS capabilities applied in the environment of CPPS. This contributes to a new perspective on CPPS with the focus on social interactions and social dynamics as innovative enhancements.

To answer these research questions we applied an iterative research approach with three steps. In the first step, we reviewed the state-of-the-art in literature using a *systematic mapping study* method and consolidated existing design knowledge on self-adaptation strategies in CPS. The goal of the first step is to answer RQ1. In the second step, we synthesized and analyzed the collected knowledge to derive recurring adaptation *patterns* that can be applied for engineering CPPS. The goal of the second step is to answer RQ2. In the third step, we explored a new perspective on adaptive CPPS by introducing *collective intelligence system principles* with humans, but also machines-in-the-loop. The goal of the third step is to answer RQ3.

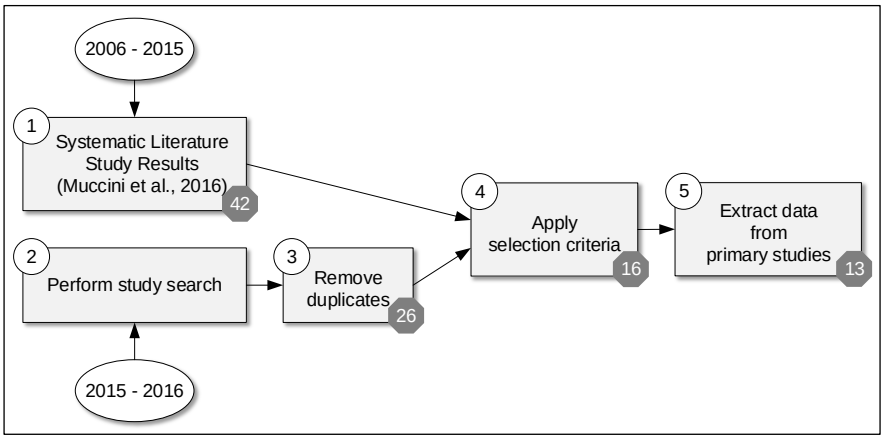


Fig. 13.2 Applied systematic mapping study process

13.4 Systematic Mapping Study Method

In order to get an overview of the current state of primary studies focusing on self-adaptation approaches in CPS on an architectural level and get insights into recurring patterns and models, we performed a systematic mapping study (SMS). To apply this research method in an unbiased, objective and systematic way, we followed the guidelines by Kitchenham and Charters (2007). In contrast to a systematic literature review, a SMS is applied to review a specific software engineering topic area and classifies primary research papers in that specific domain (Kitchenham et al, 2011). Thus the research questions for such a study are generally broader defined and more high level to provide an overview of a certain topic (Kitchenham et al, 2011). In the following we briefly summarize the performed study process and activities[5]. For detailed information about the study and its results, we refer the interested reader to the study protocol (Musil et al, 2016c) and the study website[6].

The study started with defining an initial study protocol. Since the protocol is a critical element of a systematic study, it was piloted by reviewing a sample of 4 papers. In the following, the study protocol was revised with respect to the pilot results. Once all reviewers agreed on the protocol, the phase of conducting the SMS started by applying the search strategy and selection criteria, data extraction form, data analysis methods, and reporting strategy defined in the protocol.

Fig. 13.2 shows the overall systematic mapping study process that we applied with the number of remaining papers after each phase of the study. The study was conducted by 6 researchers. Two reviewers defined the initial protocol. The retrieving and selecting publications process was performed by two other reviewers. Four

[5] If the reader is familiar with this research method, this section can be skipped.

[6] Supplementary material of the study is available at: `http://qse.ifs.tuwien.ac.at/ci/material/pub/mde-cpps17/`

reviewers extracted the data from the selected studies. Finally, they synthesized and analyzed the data as well as prepared the final study report. These final steps were crosschecked by the other two reviewers.

13.4.1 Search and Selection Strategy

The initial set of primary studies under investigation is based on the replication package of the study *"Self-Adaptation for Cyber-Physical Systems: A Systematic Literature Review"* by Muccini et al (2016), where the search and selection strategy as well as the inclusion/exclusion criteria are defined, that were used for retrieving the studies. The scope of the systematic literature review includes studies from 2006 to mid 2015. Therefore, the SLR protocol is reused to extend the set of primary studies by searching and selecting studies that were published since mid 2015 (end of scope of the SLR) and are relevant for this SMS.

In order to cover as many as possible relevant studies about self-adaptation approaches applied in CPS on an architectural level, we performed searches in four of the largest scientific online databases as sources of primary studies: *IEEE Xplore Digital Library*, *ACM Digital Library*, *SpringerLink*, and *ScienceDirect*. For these searches, we used defined keywords and combinations of them to identify candidate papers, e.g., *software*, *architect*, *cyber-physical*, *control system*. The involved reviewers applied the search strategy to identify potential study candidates. The search results are documented in a spreadsheet where the identified candidate studies are collected and stored. In addition, duplicates are removed. Each paper is indexed by a unique identifier and title.

The identified set of candidate studies is carefully assessed and filtered for their actual relevance to answer RQ1 by two reviewers. Therefore, the study goals and well-defined study selection criteria are used to determine which studies to include or exclude. Hence the inclusion and exclusion criteria defined in the SLR protocol (Muccini et al, 2016) are extended. A study is included if it is compliant to the following inclusion and exclusion criteria:

IC 1: Studies proposing, leveraging, or analyzing an architectural solution, architectural method or technique specific for CPS.
IC 2 (updated): Studies in which multiple types of self-adaptation are explicitly used as an instrument to engineer CPS.
IC 3: Studies subject to peer review (Wohlin et al, 2012) (e.g., journal papers, papers published as part of conference proceedings).
IC 4: Studies published since 2006.
IC 5 (new): Studies in which self-adaptation mechanisms are applied at least at two layers of the technology stack.
IC 6 (new): Studies comprising at least a minimal description of a concrete scenario or use case.

EC 1: Studies that are written in a language other than English, or that are not available in full-text.
EC 2: Secondary studies (e.g., systematic literature reviews, surveys, etc.).
EC 3 (new): Studies of poor quality (e.g., poorly described architecture or use case).

Results of selections and rejections are crosschecked by two other reviewers and any disagreements are discussed and resolved. Finally, the set of studies to be included in the data collection process is finalized.

13.4.2 Data Extraction

For each study remaining after the selection process, we independently investigated and extracted pre-defined data. In addition to including all the data items needed to answer RQ1, the data extraction form provides standard information about the publication. The definition of pre-defined extraction forms with data items allows to survey each study in the same way (objectively) and reduces the room for bias. Table 13.2 gives an overview of the data items that were collected from the primary studies to answer the research question. Each primary study was assigned to and reviewed by at least two reviewers. After discussion of the individual results for each study with the other reviewers, the extracted data were collected and documented in a spreadsheet in a consistent manner.

Table 13.2 Data Extraction Form

Data Item	
(D1) Study title	(D2) Publication year
(D3) Venue	(D4) Country
(D5) Application domain	(D6) Overall architectural style
(D7) Overall system goal	(D8) Type of distribution
(D9) Uncertainties considered	(D10) Adaptation purposes/goals
(D11) Adaptation mechanisms applied	(D12) Location of the adaptation mechanisms in the technology stack
(D13) Inter-adaptation coordination mechanisms	

13.4.3 Data Analysis and Reporting

The process of analyzing and synthesizing the collected data of the SMS includes the application of descriptive statistics and representation and interpretation of the

results with respect to RQ1. Besides standard information about each included paper (study title, publication year, venue, country), data items needed to answer RQ1 were collected and analyzed. Data item (D5) captures the reported application domain of the study to ensure representative and evaluated results. In addition, the application description supports a better understanding of the approach, and maybe allows to draw conclusions about a more beneficial application of particular adaptation mechanisms in one domain. Data item (D9) focuses on different types of addressed uncertainties in the environment, in parts of the system itself, and in requirements/goals for which adaptation is applied. This knowledge supports a better understanding of the focus of current research and shows what uncertainty types are mostly addressed and what areas of uncertainties are not yet addressed at all. Data item (D10) summarizes different purposes and goals of applying adaptation mechanisms in a CPS, while data item (D11) is used to identify and investigate the types of adaptation mechanisms applied in the study. To generalize the technology stack that is commonly used for applying adaptation mechanisms, data item (D12) is used to create a general layer model. Finally, data item (D13) captures the interaction and coordination between different adaptation mechanisms across the layers.

The analysis results and their visual representation are documented in a spreadsheet that is available at the study website[7].

13.5 Adaptation in Cyber-Physical Systems

After applying inclusion and exclusion criteria to the initial set of 42 primary studies from the SLR by Muccini et al (2016) as well as to the extended set of 26 studies, data were extracted from a total number of 13 primary studies to answer RQ1. An overview list of all selected primary studies is shown in Table 13.3. After finishing the data collection, the results were checked for consistency and completeness as well as documented in a spreadsheet. This section presents the results of the conducted systematic mapping study.

Fig. 13.3 presents the variety of application domains (D5) where proposed self-adaptation approaches were applied to evaluate their efficiency and performance. The results show that transportation (23%) is the dominant application domain, followed by robot navigation (15%), energy (15%) and manufacturing (15%).

All studies report to enable adaptation in the CPS to address uncertainties (D9) in the environment and to make context-aware decisions. The identified clusters of uncertainty types in the environment are presented in Fig. 13.4(a). As CPS operate in real-time and thus have to deal with environments that are usually subject to volatile and dynamic conditions (such as weather conditions, road conditions, traffic flow, danger zones, parking spaces), the most dominant uncertainty type that need to be addressed can be described as dynamic conditions (77%). For enabling the exchange of knowledge and data or negotiations between components of the CPS,

[7] Supplementary material of the study is available at: `http://qse.ifs.tuwien.ac.at/ci/material/pub/mde-cpps17/`

Table 13.3 Final list of primary studies retrieved in the systematic mapping study

ID	Title	Reference
1	An Architecture of Cyber Physical System based on Service	(Yu et al, 2012)
2	An Architecture Framework for Experimentations with Self-Adaptive Cyber-Physical Systems	(Kit et al, 2015)
5	C-MAP: Framework for Multi-agent Planning in Cyber Physical Systems	(Mukherjee and Chaudhury, 2013)
7	Context-Aware Vehicular Cyber-Physical Systems with Cloud Support: Architecture, Challenges, and Solutions	(Wan et al, 2014)
9	Towards Context-aware Smart Mechatronics Networks: Integrating Swarm Intelligence and Ambient Intelligence	(Gupta et al, 2014)
13	A multi-agent RFID-enabled distributed control system for a flexible manufacturing shop	(Barenji et al, 2014)
19	Multi-Agent Control System for Real-time Adaptive VVO/CVR in Smart Substation	(Nasri et al, 2012)
37	Coupling heterogeneous production systems by a multi-agent based cyber-physical production system	(Vogel-Heuser et al, 2014)
41	Cloud Robotics: Architecture, Challenges and Applications	(Hu et al, 2012)
51	Continuous Collaboration: A Case Study on the Development of an Adaptive Cyber-physical System	(Hölzl and Gabor, 2015)
62	Cloud-Assisted Context-Aware Vehicular Cyber-Physical System for PHEVs in Smart Grid	(Kumar et al, 2015)
63	Cyber-physical-social system in intelligent transportation	(Xiong et al, 2015)
67	Cross-layer Virtual/Physical Sensing and Actuation for Resilient Heterogeneous Many-core SoCs	(Sarma et al, 2016)

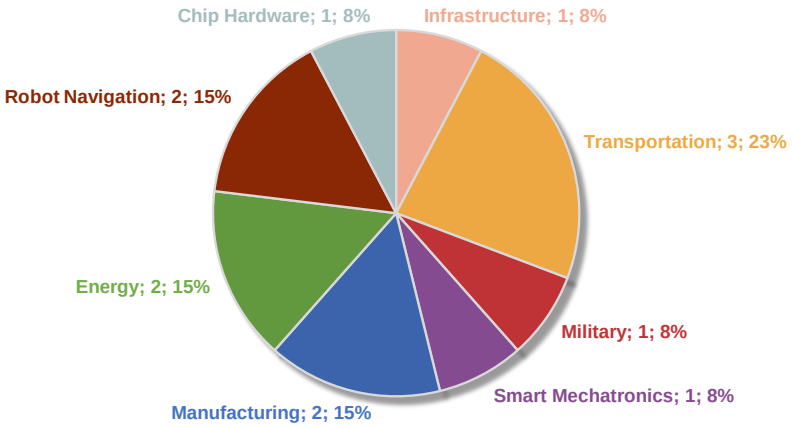
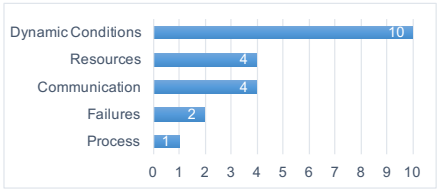
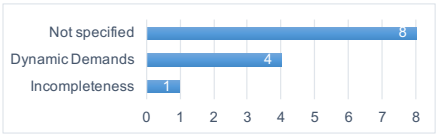


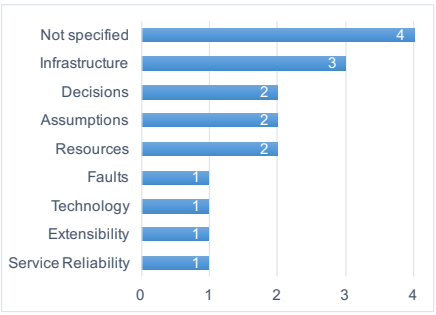
Fig. 13.3 Overview of identified application domains (D5)



(a) Uncertainties in the environment



(b) Uncertainties in requirements and goals



(c) Uncertainties in parts of the system itself

Fig. 13.4 Types of uncertainties addressed by existing approaches for self-adapting CPS (D9)

the communication reliability in the environment is also a relevant issue to consider. Due to unforeseen noises in the environment, CPS need to deal with limited communication and bandwidth as well as latencies (31%). In addition, not predictable resource constraints (such as fuel, ammunition, personnel, ingredients) in the environment (31%) can lead to obstacles for the correct operation of CPS that need action to be taken. Further identified uncertainty types related to the environment are failures (such as production system breakdown, infrastructure failure) reported by 15% of the studies and process (such as unforeseen changes of the requested production process) reported by 8%.

Only five studies mention uncertainties (D9) in requirements and goals that are affected by their proposed adaptation approaches. In particular, they describe dynamic demands by customers or the market (31%) and incompleteness of specified requirements (8%) as uncertainty types. The results are summarized by Fig. 13.4(b).

Most of the studies also considered uncertainty in parts of the CPS itself to be addressed by adaptation approaches, but the results are quite diverse as Fig. 13.4(c) illustrates. The most frequently mentioned uncertainty type is infrastructure (such as broken hardware, aging effects) considered by 23% of the studies. Other uncertainty types are system decisions (15%), assumptions of behavior and knowledge of other components (15%), system resources (15%), internal faults (8%), changing technology (8%), extensibility to integrate unknown features into the system (8%), and service reliability (8%).

In our mapping study we further investigated the purposes and goals of designing self-adaptive CPS (D10). We clustered the results of the identified adaptation

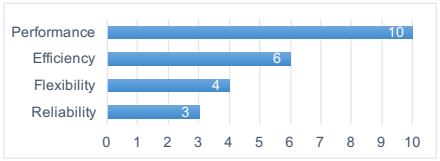


Fig. 13.5 Purposes and goals of adapting CPS (D10)

purposes and goals as presented in Fig. 13.5. As the dominant adaptation purpose, we identified performance in 77% of the studies. Other stated adaptation purposes and goals are efficiency (46%), flexibility (31%), and reliability (23%).

The analysis of the technology stacks that are used in the studies for applying adaptation mechanisms (D12) revealed a general architecture model of CPS comprising the following 6 different layers:

1. *Physical Layer:*
   This layer represents physical real-world components of the CPS that interact with humans. Examples include vehicles, production systems, robots, road infrastructure, parcels, and smart meter. The physical components monitor the environment, collect information with sensors and take actions to modify the environment with actuators.

2. *Proxy Layer:*
   This layer constitutes the transition from the real-time physical world to the virtual world where the physical components are represented by intelligent mechanisms. Examples include interfaces, software agents, and smart components. These mechanisms communicate the collected context information to the computational system in the upper layers and receive responses based on the sent data.

3. *Communication Layer:*
   The interaction between the proxy layer and the upper layers is enabled by this layer. A variety of technologies are available (wired/wireless, short/long-range) to use for the communication process. Examples include Bluetooth, ZigBee, WLAN, LAN, and special communication protocols.

4. *Service and Middleware Layer:*
   This layer provides context-aware services and middleware to process and analyze the collected data according to defined goals. In some studies these services are located in the cloud. Examples include traffic cloud services, controller intelligent agents, component framework, and optimization unit.

5. *Application Layer:*
   This layer represents the domain-specific application that is in charge of system control and responsible for realization of the system goals. Therefore, it has to acquire all required resources and make justified decisions to achieve the demanded

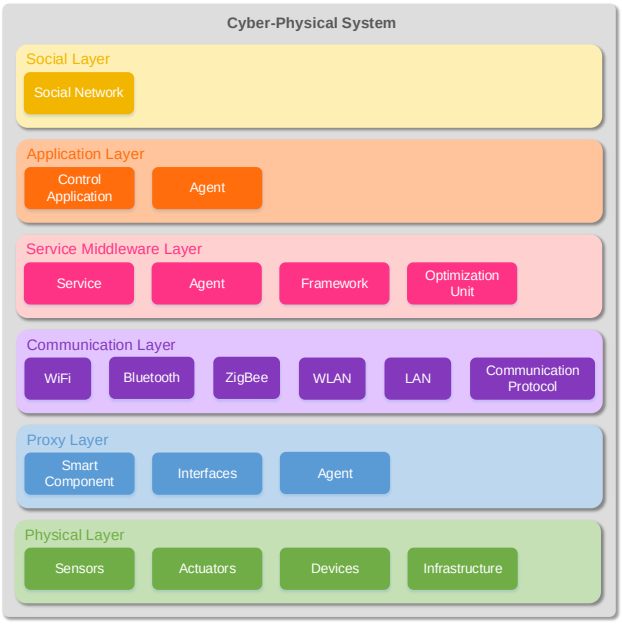


Fig. 13.6 General multi-layer architecture of CPS

Quality-of-Service. Examples include software agents and control applications.

6. *Social Layer:*
   This layer represents an optional extension of the common CPS architecture. It integrates social systems into a CPS architecture by providing specific mechanisms to humans, organizations, and societies so that they can offer knowledge and feedback. The combination of social and physical sensing information can then achieve more intelligent and improved decisions by a CPS. An example could be a social network.

Fig. 13.6 illustrates the identified general multi-layer CPS architecture.

We further investigated the types of adaptation mechanisms that are applied in proposed CPS architecture designs (D11). Fig. 13.7 shows the frequencies of different self-adaptation mechanisms as well as their locations in the technology stack (D12). Smart elements are mostly applied (77%) and always located at the proxy layer. They represent the physical components capable of self-adaptation. Other types of adaptation mechanisms are broader distributed across the technology stack. Multi-agent systems (69%) are applied at the proxy layer, service middleware layer and application layer, followed by MAPE (54%) at the proxy layer and service middleware layer, autonomous entities (38%) at the proxy layer, service middleware layer and application layer, collaborative entities (23%) at the proxy layer, service middleware layer and application layer, swarm (15%) at the service middleware layer and application layer. Self-organization (at the application layer) and social

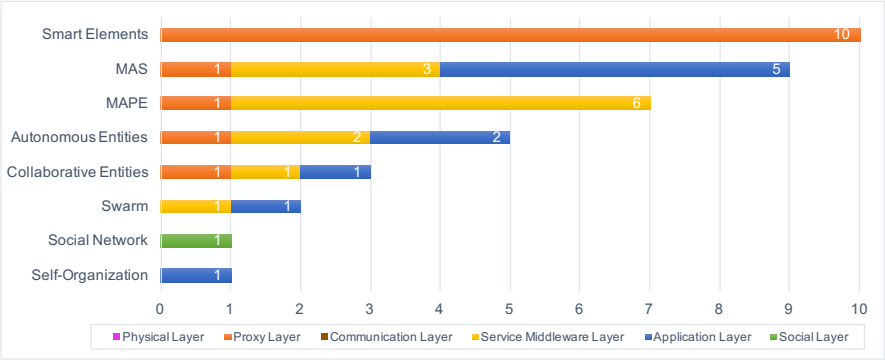


Fig. 13.7 Self-adaptation mechanisms applied at multiple layers in CPS architectures (D11/D12)

network (at the social layer) are equally applied (both 8%) as adaptation mechanism. Each primary study proposed an application of adaptation at the proxy layer and service middleware layer.

In the primary studies we observed the application of combinations of different types of adaptation mechanisms that interact and coordinate across multiple layers of the technology stack (D13). By applying such solution approaches the CPS is capable to deal with different uncertainties and concerns at a time using adaptation. Fig. 13.8 presents the observed combinations of adaptation mechanisms. The results show that the majority of primary studies combine MAPE with smart elements or MAS with smart elements (both 31%) in a CPS architecture design. Combinations of multiple multi-agent systems were realized in 23% of the primary studies. 15% of the primary studies equally combined self-organization or swarm with autonomous entities, MAS with MAPE, MAPE with MAPE, and autonomous entities with MAPE. Only 8% of the primary studies combined MAS with swarm and MAPE with collaborative entities.

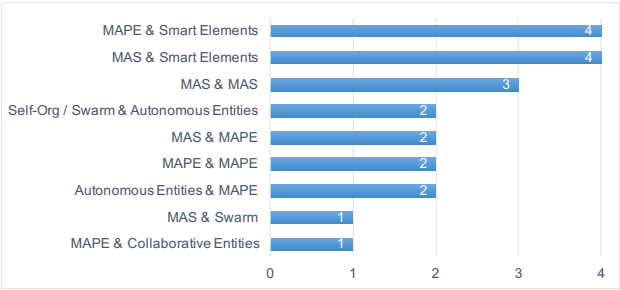


Fig. 13.8 Combinations of adaptation mechanisms (D13)

13.6 Threats to Validity

As with any empirical research, there are threats to the validity of this study that need to be considered. The following potential validity threats were identified and discussed how to mitigate them in order to strengthen the outcomes of the study.

- *Quality of the selected primary studies.* We defined several inclusion/exclusion criteria to ensure sufficient quality of the selected primary studies for the mapping study, but we did not apply a systematic and detailed quality assessment procedure in order to critically evaluate the quality of each paper as it is common in systematic literature reviews. To mitigate this weakness to some extent, we particularly added the inclusion criterion that each primary study must provide a description of a concrete scenario or use case to draw conclusions for real-world applications. In addition, we discussed and excluded some papers with determined poor quality during the mapping study.
- *Adaptation of data items.* Based on our expertise, we defined a set of data items for the data collection of the mapping study. During piloting the data extraction form, we identified some data items that did not always fit for the selected primary studies or were omitted. Based on the results of this initial review, we updated the data extraction form for the remaining studies to ensure a consistent data collection and analysis of the results.
- *Limited CPPS expertise of research team.* The research team consisted of experts in the fields of self-adaptation, software architecture, cyber-physical systems and collective intelligence systems without special experience with cyber-physical production systems. In order to reduce bias, we consolidated other researchers with expertise in the CPPS domain and discussed the outcomes.
- *Generality of the results for CPPS.* Due to a general lack of knowledge on self-adaptation specifically in CPPS and the nature of a mapping study to provide a broad overview of a research topic area, we reviewed state-of-the-art self-adaptation approaches related to the larger family of CPS. However, during the mapping study we focused on self-adaptation applied in CPS in the manufacturing domain and eventually included some primary studies with described use cases in this domain. During the data analysis, we came to the conclusion that these studies are no outliers and informally discussed the results with CPPS experts. Nevertheless we are careful to generalize the results for CPPS and see a need for further research in this direction to enhance the validity of the results.
- *Small number of primary studies.* Since CPS and specifically CPPS is still a quite young research field, the number of primary studies providing insight on how self-adaptation can be used in addressing uncertainty in these systems is small. Thus the data extracted from this mapping study can only be considered as evaluation of the current state-of-the-art. This study should be replicated after a period of time.

13.7 Reflection of the Systematic Mapping Study Results

The goal of the systematic mapping study was to capture, consolidate and document state-of-the-art design strategies and best practices how engineers currently deal with adaptation across the technology stack in CPS. Based on the analysis results of the collected data, we were able to synthesize this knowledge to derive recurring patterns in CPS architectures that can be reused to engineer adaptive CPPS to handle uncertainty. In particular, we studied the interaction and coordination between different applied adaptation mechanisms across the layers. Finally, the results of this investigation supports the proposal of three identified adaptation patterns to answer RQ2, whereby each pattern is applicable for a different purpose. The consolidation and documentation of this design knowledge represents a valuable contribution for CPPS engineers as a useful starting point for the system's design with respect to adaptation. The provided insights into evaluated and effective design strategies, that enable self-adaptation in CPS, are promising to be useful for reuse in CPPS architecture design as well. In evaluations of the proposed approaches with scenarios in different application domains, the solution designs demonstrated their applicability and effectiveness to address particular uncertainties and concerns related to self-adaptation. In addition, they were observed to support the CPS in the realization of the stated adaptation purposes and goals.

However, these results require further investigation with specific focus on the CPPS domain and related application scenarios, but our contributions can serve as a useful basis for future research. The identified patterns are introduced in the following section in more detail.

13.8 Patterns for Self-Adaptation

To derive the patterns, we carefully studied the collected data and analysis results of the conducted systematic mapping study. We created a comprehensive table[8] presenting the concrete designs of self-adaptation mechanisms applied across the technology stack for the application scenario of each investigated study. The comparison across all represented solution designs highlighted areas that follow similar or equal strategies, enabling us to identify three multi-adaptation patterns with different combinations of multiples types of self-adaptation within a system: SYNTHESIZE-UTILIZE, SYNTHESIZE-COMMAND, and COLLECT-ORGANIZE. In particular, a multi-adaptation pattern provides knowledge about (1) the kinds of used adaptation mechanisms, (2) their layer locations, and (3) the cross-layer inter-adaptation interactions between the respective mechanisms. This section describes each pattern according to the pattern writing form provided by Meszaros and Doble (1997).

[8] Supplementary material of the study is available at: `http://qse.ifs.tuwien.ac.at/ci/material/pub/mde-cpps17/`

A pattern aims to capture best practices that address certain recurring problems in a specific context for reuse and guidance (Meszaros and Doble, 1997). Thereby it is important to communicate the purpose of the pattern, the concrete context in which to apply it, the problem it addresses, a description of the solution to solve the problem, and the effects and consequences it creates in detail. Such a detailed pattern description efficiently supports the reader to decide about the applicability of the pattern of interest in a specific scenario and eventually also to guide the application. The patterns are structured using the following template based on Meszaros and Doble (1997):

- **Name:** a name to refer to the pattern
- **Context:** a short description of the situation in which to apply the pattern
- **Problem:** a short description that describes a specific recurring problem that the pattern solution aims to solve
- **Solution:** kind of a reusable model that solves the problem
- **Consequences:** set of rationales why the proposed solution is most appropriate for the stated problem (benefits) as well as a set of other effects and limitations to make clear where the pattern is not applicable
- **Known Uses:** a short description of representative use cases that illustrate the application of the solution and the positive effects to address the problem

13.8.1 Synthesize-Utilize Pattern

Context: A distributed application is composed of diverse physical resources and application entities, whereby each of which possesses data that is heterogeneous, spatially distributed and continuously changing.

Problem: A distributed application seeks to improve the utility of its services to the physical resources by dynamically exploiting rich context information.

Solution: SYNTHESIZE-UTILIZE is composed of a MAPE-like adaptation mechanism on the service middleware layer and autonomous entities on the application layer. The pattern is illustrated in Fig. 13.9, which also depicts the concrete workflow steps across the layers. The characteristic workflow between the layers is as follows: (1) The service middleware layer receives data from the physical resources via the proxy layer and requests data from the autonomous entities on the application layer. (2) The service middleware layer uses MAPE-like adaptation for the continuous collection and synthesis of data. (3) The consolidated data is then provided to the autonomous entities on the application layer. (4) The autonomous entities dynamically optimize their services to the physical resources by collaborating with each other and by using the integrated data that is offered from the service layer.

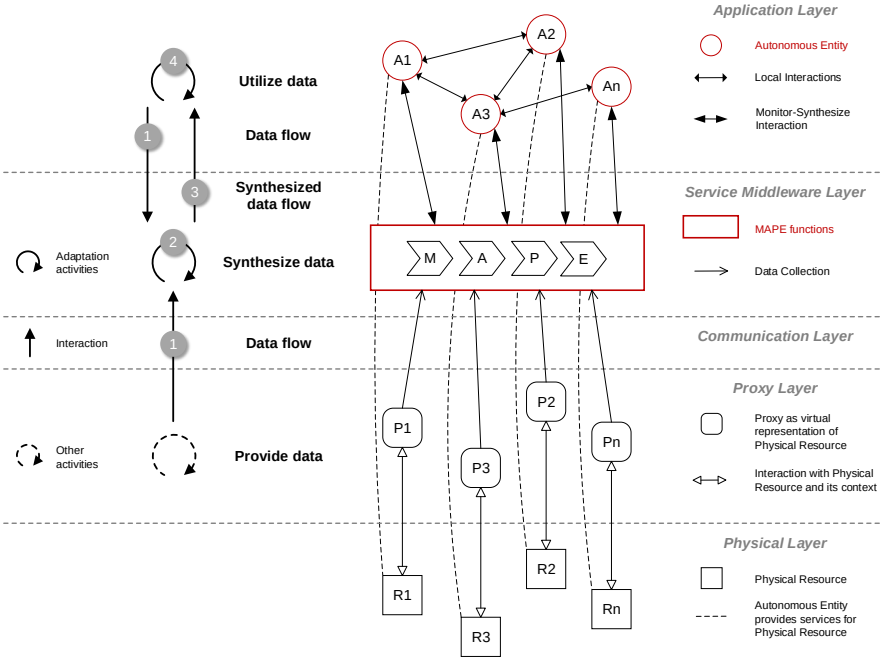


Fig. 13.9 Synthesize-Utilize pattern with adaptation mechanisms (red) and characteristic workflow steps across layers

Consequences:

+ Efficient sourcing of heterogeneous data from a diverse set of systems and its consolidation that can be used for situated optimization.
+ Reduced effort for adding and removing data sources.
– Timing aspects.
– Varying quality and granularity of data requires additional data acquisition and integration effort.

Example "Intelligent Transportation System": In this example the physical resources include vehicles and sensors that are positioned along the road infrastructure. On the application layer, services are provided to traffic participants on a global scale (traffic-related information) and individual scale (smart parking, routing/navigation), as well as to traffic management authorities (traffic performance, congestion control). By collecting data from individual vehicles and road sensors, the service middleware layer can monitor traffic-related information. It then continuously integrates the data into traffic models that are created from dynamic simulations and experiments. Based on the evaluated models the traffic information is consolidated and forwarded with respect to the individual traffic services. The consolidated data is used for optimization and if necessary personalized to the individual recipient (e.g., traffic participant). Further the services locally interact (e.g., via Vehicle-2-

Vehicle networking, flexible web service orchestration) with each other in order to factor in additional context information.

Identified in Primary Studies: ID-07 Dynamic Parking Service (Wan et al, 2014), ID-63 Intelligent Transportation System (Xiong et al, 2015)

13.8.2 Synthesize-Command Pattern

Context: A distributed application produces its functionality by employing an assembly of heterogeneous, physical resources which are independent and have different capabilities and capacities.

Problem: A distributed application exploits data of individual resource to improve its overall utility by changing the resource configuration that produces the applications functionality.

Solution: SYNTHESIZE-COMMAND is composed of a MAPE-like adaptation mechanism on the service middleware layer and a multi-agent system (MAS) on the application layer which manages the physical resources. The pattern is illustrated in Fig. 13.10, which also depicts the concrete workflow steps across the layers. The characteristic workflow between the layers is as follows: (1) The service middleware layer receives data from the physical resources via the proxy layer. (2) The service middleware layer uses MAPE-like adaptation for the continuous collection and synthesis of physical resource data. (3) The consolidated data is then used to derive commands which are sent to the MAS on application layer. (4) The agents in the MAS locally interact and reorganize so that the commands are performed in an efficient way on the physical layer.

Consequences:

+ MAPE-based "plug-in" model allows selection of appropriate adaptation function.
+ Separation of concerns: reconfiguration of request and its execution.
+ Easy extensibility of resources on the physical layer.
– Cross-layer coordination is complex.
– Reduced autonomy of physical resources due to high dependence on central command coordination.

Example "Flexible Manufacturing Shop": In this example the physical resources include various kinds of manufacturing equipment which are grouped into stations. The application layer consists of manufacturing resource agents, whereby the agents are connected to the physical resources with specific agent-machine interfaces. On the service middleware layer, the MAPE-like adaptation mechanism is realized in a station controller which collects data on realizable capabilities from the station

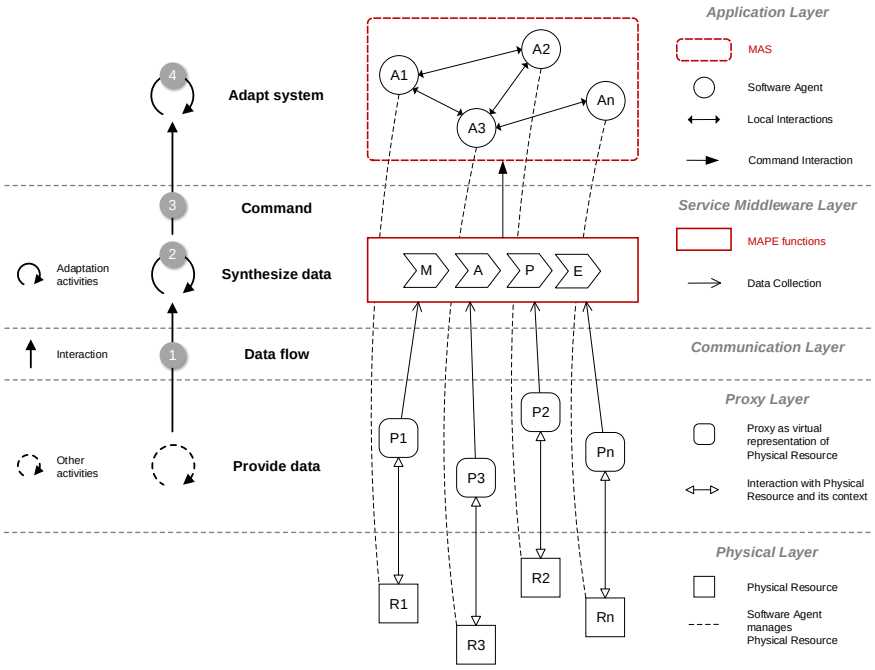


Fig. 13.10 Synthesize-Command pattern with adaptation mechanisms (red) and characteristic workflow steps across layers

and compares it with a product capability list in order. If the product capabilities are realizable on the station, the station controller assigns information on related manufacturing resources and process steps for efficient production to the resource manufacturing agents. The resource manufacturing agents forward the production information to the associated manufacturing equipments and continuously send feedback about the status of the manufacturing process back to the station controller, which is case of constraint conflicts would adapt the process on the respective station.

Identified in Primary Studies: ID-01 Water Resource Management (Yu et al, 2012), ID-13 Flexible Manufacturing Shop (Barenji et al, 2014)

13.8.3 Collect-Organize Pattern

Context: A distributed application provides services to autonomous, cyber-physical entities, whereby each entity generates their own, local models and collects data that is spatially distributed and continuously changing.

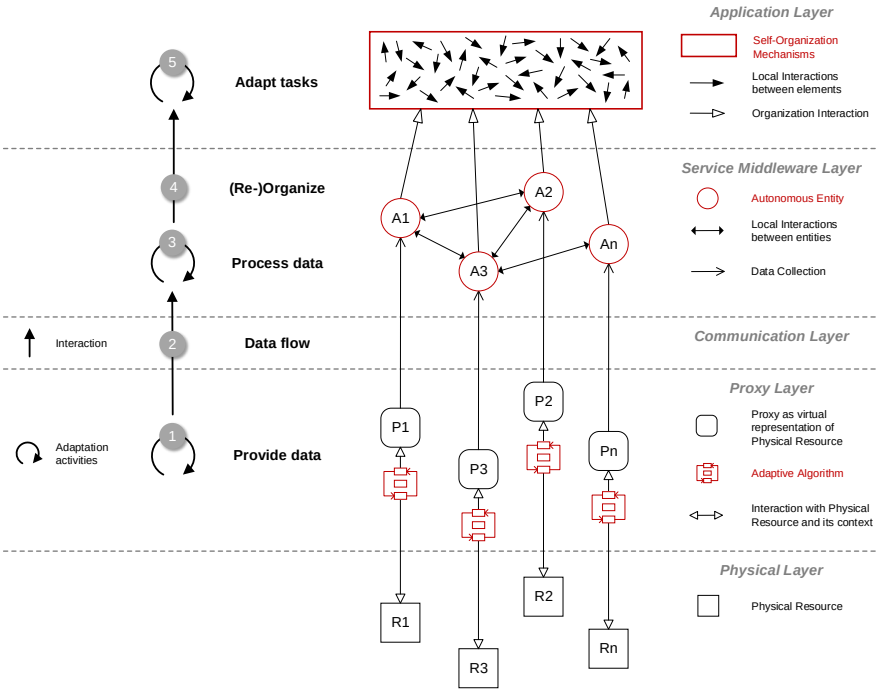


Fig. 13.11 Collect-Organize pattern with adaptation mechanisms (red) and characteristic workflow steps across layers.

Problem: A distributed application seeks to improve the overall utility of its service, which requires the autonomous entities to efficiently share information and coordinate their tasks on a local basis.

Solution: COLLECT-ORGANIZE is composed of adaptive algorithms on the proxy layer, autonomous entities on the service middleware layer and self-organization mechanisms on the application layer. The pattern is illustrated in Fig. 13.11, which also depicts the concrete workflow steps across the layers. The characteristic workflow between the layers is as follows: (1) On the proxy layer, adaptive algorithms continuously integrate data from physical resources into local models. (2) Autonomous entities on the service middleware layer exchange local information in order to generate and update global models. (3) On the application layer, self-organization mechanisms facilitate the adaptation with regard to situated tasks among the individual entities based on the local and global models.

Consequences:

+ Efficient organization in highly dynamic application scenarios.

+ Cyber-physical entity has autonomy and maintains operational, even if service middleware layer adaptation fails.
– Cross-layer coordination is complex, in particular with regard to emergent behavior induced by self-organization mechanisms.
– Pattern requires smart systems at the bottom of the architecture instead of "dumb" physical systems.

Example "Smart Parking": In this example the physical resources include smart vehicles which are equipped with intelligent sensors that collect data about available parking space. The service provided on the application layer is a smart parking service, which supports the vehicles in the discovery of parking space and with negotiating the allocation of suitable space. On the proxy layer, the data from the sensors is integrated into models using adaptive algorithms (like MAPE-K loops) and real-time CPS control logic. On the service middleware layer, an autonomous entity, representing the vehicle, receives the model information. These entities are realized as autonomous components which are dynamically grouped into ensembles (subset of vehicles in proximity) consisting of a coordinator and multiple members. Within an ensemble the autonomous components continuously exchange their partial models in order to collectively produce a more complete, global model of the situation. The consolidated model is then forwarded to the application layer, where it is used by the smart parking service to derive a list of suitable parking spaces. These spaces are particular for the individual vehicle with respect to the other vehicles, leading to emergent self-organizational interaction, since the ensembles are highly temporal.

Identified in Primary Studies: ID-02 Smart Parking (Kit et al, 2015), ID-51 Rescue Robot Navigation (Hölzl and Gabor, 2015)

13.9 Potential of Collective Intelligence Systems for Cyber-Physical Systems and Cyber-Physical Production Systems

This section presents an overview about emerging directions of how Collective Intelligence Systems are used in CPS and CPPS. Based on the background of CIS in Sect. 13.2.3 and the findings of our systematic mapping study in Sect. 13.5, we describe the three directions of capability augmentation, emergent machine-to-machine interactions, and multi-disciplinary knowledge integration and coordination.

1. Collective Intelligence Systems for Capability Augmentation

The results of the systematic mapping study reveal a recent trend to add an additional "social" layer in a CPS architecture that involves components in a CPS to address human and social factors. So-called *Cyber-Physical-Social Systems (CPSS)* (Wang,

2010) consider social and human dynamics as an essential part of an effective CPS design and operation. A CPSS is defined as a complex system that is constituted by three parts: a physical system, a social system including human beings, and a cyber system that connects both of them (Xiong et al, 2015). The social system is a human-centered system, like social networking sites and social media platforms, that explicitly involves individuals, organizations, and societies in the processes of a CPS for information exchange and feedback. One application scenario of a CPSS architecture is an intelligent transportation system (Xiong et al, 2015) where the social system aims to effectively aggregate (by so-called *social sensor network*) and disseminate up-to-date travel-related information and resources from multiple sources. Examples of this shared information are emergency events, traffic jams, navigation, road conditions, and car-sharing information. Such useful information can have influences on the decisions and behavior of individuals as well as on transportation authorities who can use this information to improve their services and management. As one successful example of such a CIS which support a transportation system is *Waze*[9]. They concluded that CPSS have a significant value, but there are existing challenges to control and manage them by applying traditional theories and methods which demand the research of novel approaches.

Considering and utilizing the potential of humans-in-the-loop, their interactions and the created social dynamics offer new opportunities for CPPS as well. According to the scenario of a "Cyber-Physical System for the factory of the future" investigated by the German acatech - National Academy of Science and Engineering (2011), industrial production systems should be able to react virtually in real-time to changing customer demands as well as changes in the market and the supply chain. A CIS in the social layer of a CPPS architecture can support this vision by facilitating to effectively incorporate humans into the CPPS processes and by introducing CIS-specific capabilities. Companies as well as customers (local or globally distributed) can interact with each other by using a CIS for sharing opinions, experiences, requirements as well as discussing new ideas and designs for future products. The combination of information and feedback from multiple sources (different sensors in the physical system and different human groups in the CIS) enables a more efficient sensing of the CPPS and thus improves its decision-making processes. For example, based on the collected knowledge a CPPS can react rapidly to customer feedback and optimize the manufacturing of tailored customer products or correct defective production models.

2. Collective Intelligence Systems as Enabler for Emergent Machine-To-Machine Interactions

CIS-based CPPS architectures highlights the potential of a new kind of social interactions that goes beyond the typical human-to-human interactions. So far CIS approaches consider humans as essential entities in the critical feedback loop to be

[9] `http://www.waze.com/` (last visited 01/15/2017)

successful and effective (e.g., *Yelp*[10] in the domain of business ratings and reviews or *Facebook*[11] for creating a social network of friend relationships). But if humans were regarded as one of many variability points in a CIS architecture, they could be replaced by machines as for instance robots or CPPS to realize machine-to-machine configurations (Musil et al, 2015a). The integration of a CIS with machines that represent its actor base into a manufacturing environment enables the connection and communication of several groups of CPPS or single systems to support the machines to share their information and experiences among each other, which is illustrated by Fig. 13.12. Single globally distributed, adaptive and evolutionary production units that belong to different operators are situated in a specific, local context and thus have only awareness about local available information but have difficulties to access remote information from other machines involved in the production processes. The creation of a global network of cooperating and interacting industrial plants as well as the aggregation and coordination of their collective intelligence by applying CIS mechanisms, that have proven to be effective, offers the possibility of a global access to relevant and critical information and data of individual machines with respect to context, status, defects, diagnoses, experiences, influences, effects, analytics, and learned capabilities. In their work Bauernhansl et al (2014) recognize on numerous occasions the potential of social media platforms, besides data mining and mobile, as a pivotal enabling technology for smart factories of the future. *Factory Social Media* is expected to play an important role in future CPPS process improvement efforts by enabling effective and efficient bottom-up information collection and dissemination capabilities in human-human, human-machine and machine-machine scenarios. But the authors also mention the lack of their current application, although there is a clear need to support the increased computerization of physical systems and address the resulting need to organize and coordinate.

Approaches in this promising direction can be found in the work of Mukherjee and Chaudhury (2013) who elaborated on this topic and proposed a novel multi-agent planning framework. For illustration they used the scenario of a network centric battle space with military CPS. The challenge here is the collaboration and coordination of plans between multiple planner agents. To deal with uncertainty while planning, Mukherjee and Chaudhury (2013) explored (1) bottom-up biologically inspired continuous planning in order to adapt to changing environment and (2) Blackboard-based multi-agent coordination. Similar to this approach, CIS use the nature-inspired coordination mechanism of stigmergy (Heylighen, 2016) which enables bottom-up, environment-mediated coordination and indirect communication of agents via traces in the environment (Musil et al, 2015a). Thus the stigmergic process creates a positive feedback loop that promotes awareness among agents about the activities of others and stimulates further agent activities. The resulting feedback loop provides CIS with emergent, self-organizational capabilities and allows the system to adapt (Musil et al, 2015a).

[10] `http://www.yelp.com/` (last visited 01/15/2017)

[11] `http://www.facebook.com` (last visited 01/15/2017)

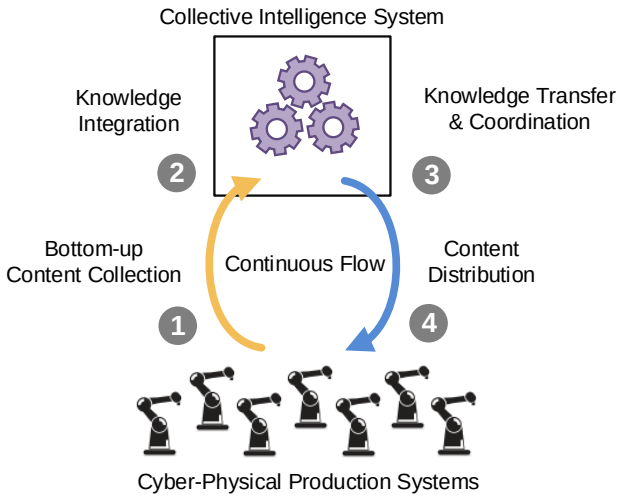


Fig. 13.12 Overview of a CIS as enabler for emergent machine-to-machine interactions with feedback loop of information aggregation (yellow) and distribution (blue)

The concept of CIS for machine-to-machine communication brings along new architectural design and realization challenges to deal with in an agenda for future research. Aspects like what processes can be supported by a CIS of machines-in-the-loop or what skills provided by machines are needed to play the role of humans in a CIS for CPPS need to be investigated.

3. Collective Intelligence Systems as Coordinators and Knowledge Integrators across, Heterogeneous, Multi-Disciplinary Domains

Similar to the support of CPPS processes during production, also the engineering of CPPS as group processes can be improved with CIS-based approaches. CPPS engineering processes involve humans from multiple engineering disciplines, such as electrical, mechanical and software engineering, who are essential factors in the planning, design, and realization of a well-functioning CPPS. Consequently the following challenges arise for the multi-disciplinary engineering teams in particular: heterogeneous representations as for instance of engineering data, models and terminologies, weak accumulation and integration of dispersed, local engineering know-how that is instrumental for the engineering processes, lack of traceability and awareness of activities and changes as well as required effective communication, coordination and sharing of knowledge and artifacts between teams across the organization (Jazdi et al, 2010; Musil et al, 2016b).

In such environments of multi-disciplinary projects in which CPPS are engineered, social interaction and communication between the experts are critical and thus need to be supported by social systems, like a CIS. This kind of a socio-technical system has the potential to enhance current engineering methods and tools

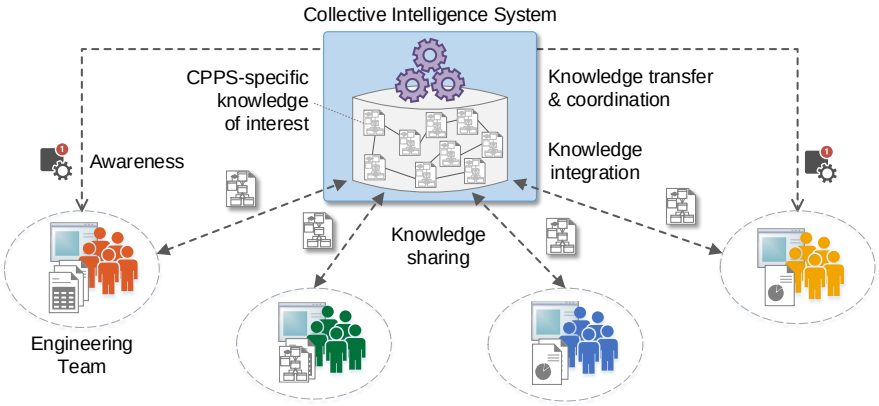


Fig. 13.13 Overview of a multi-disciplinary engineering scenrio with a CIS as coordinator and knowledge integrator

to overcome existing challenges and complexities (Musil et al, 2016b). The capabilities of a CIS support engineers to identify important implicit engineering knowledge and to integrate the dispersed, local information into an organization-wide and project-independent knowledge base in order to make it explicitly available (Musil et al, 2016b). The efficient and structured collection and maintenance of project-independent knowledge enabled by a CIS allows the sharing and awareness of engineering know-how without loss of information (Jazdi et al, 2010). So it provides the ability to deal with complexity of a CPPS and to reuse expert knowledge, and thus efficiently and transparently supports engineering, maintenance (Winkler et al, 2016), and modernization activities. Fig. 13.13 illustrates a scenario with a CIS as coordinator and knowledge integrator in multi-disciplinary engineering processes.

All three introduced CIS-focused concepts open up different new future research directions to investigate novel theories, methods and technologies, but provide a good starting point for a research agenda.

13.10 Related Work

This section presents an overview of related systematic studies in the areas of agent/multi-agent based, architecture-based, control theory-based, and self-organization-based adaptation. While we could not identify any secondary study directly looking at patterns of combined adaptation mechanisms in the field of CPS, we summarize related research that is worth mentioning.

First we looked at systematic studies surveying agent/multi-agent based approaches. Leitão (2009) presented a state-of-the-art survey on multi-agent system approaches for designing manufacturing control systems that exhibit intelligence,

robustness and adaptation. While the study does not deal directly with adaptation patterns for CPS, it reports on holonic- and agent-based architectures based on centralized and decentralized patterns such as agent federation centered in the mediator approach, adaptive control approach, hybrid control architecture, decentralized planning and rough level process planning, and hierarchical structure based on rules. Juziuk et al (2014) provided an overview of existing design patterns for multi-agent systems collected using a systematic literature review. In their study, the authors identified a total of 206 patterns. One cluster of associated patterns was formed around bio-inspired concepts such as pheromones, ants, and stigmergy. This cluster also included recent patterns related to self-organization and adaptive behavior. In addition, the authors investigated the types of systems for which these design patterns have been applied. They concluded that there is not a dominant type of system, but with regard to industrial applications the main reported application domains are process control and manufacturing, traffic and transportation.

Then we focused on research work in designing architecture-based adaptation. Weyns and Ahmad (2013) conducted a systematic literature review on claims and evidence for architecture-based self-adaptation. The authors discovered that 69% of the studies focus on a single feedback loop, with 37% of the primary studies using distinct components for each of the MAPE functions and 32% using components that mix (some of) the MAPE functions. Only 20% of the studies focus on multiple feedback loops. As commented by the authors, while MAPE serves a reference model, it is not generally considered as a reference architecture. Weyns et al (2013b) consolidated a number of well-known patterns of decentralized control in self-adaptive systems as well as described them with a simple notation to foster their comprehension. The presented MAPE patterns model different types of interacting MAPE loops with different degrees of decentralization. The set of described patterns include the *coordinated control*, *information sharing*, *master/slave*, and *regional planning*. In addition, the authors discussed drivers that should be considered by designers of self-adaptive systems when choosing one of these MAPE patterns (e.g., optimization, scalability, robustness). Ramirez and Cheng (2010) collected adaptation-oriented design patterns from the literature and open sources that support the development of self-adaptive systems. These patterns aim to facilitate the separate development of the functional and adaptive logic. In their work the authors present 12 adaptation-oriented design patterns for reuse of existing adaptation expertise and cluster them in three groups based on their overall objective in the self-adaptive system: monitoring, decision-making, or reconfiguration. Their patterns are at the level of software design in contrast to our architecture-centric perspective that we adopted in this chapter.

In addition, we had a look at control-based self-adaptation approaches. Patikirikorala et al (2012) systematically surveyed the design of self-adaptive software systems using control engineering approaches. The authors investigated control methodologies in self-adaptive systems and identified a set of design patterns, that are: feedback control system (with reactive decision making) implemented by 88.2% of the surveyed approaches, feed-forward control system (with proactive control mechanisms) utilized by 10.6% of the analyzed papers, and feedback and feed-forward

control systems used by only two over the 161 surveyed papers. A number of adaptive control strategies are also elicited and discussed in their work, being the Fixed and Adaptive the most commonly used (29.7% and 14.9%, respectively).

Research in self-organizing systems has brought forward a number of patterns. De Wolf and Holvoet (2007) consolidated and described a set of patterns to systematically design a self-organising emergent solution such as gradient fields and market-based control. The purpose of the patterns proposed in their work is to help engineers to decide which decentralised coordination mechanisms are promising to solve a certain problem, to provide best practice in using each coordination mechanism and to guide engineers in applying them. The benefits of a self-organising solution, as explained by the authors, are that it is constantly adapting to changes without a central controlling entity and is still robust to failures. In their work, Fernandez-Marquez et al (2013) provide a catalogue of bio-inspired mechanisms for self-organizing systems in form of modular and reusable design patterns. The authors investigated the inter-relations among self-organising mechanisms for engineering self-systems in order to understand how they work and to facilitate their adaptation or extension to tackle new problems. By analyzing these mechanisms and their behaviors in detail, they identified different levels: lower-level patterns include basic mechanisms (*repulsion, evaporation, aggregation, spreading*) that can be used individually and other more complex mechanisms composed of basic ones (*digital pheromone, gradients, gossip*). Higher-level patterns show different ways to exploit the basic and composed mechanisms (*flocking, foraging, quorum sensing, chemotaxis, morphogenesis*). The presented patterns are best exploited during the design phase.

All these systematic studies have in common that they did not focus on combinations of different types of adaptation mechanisms and cross-layer inter-adaptation interactions as we did in the mapping study and the adaptation patterns described in this chapter.

13.11 Conclusion & Future Work

In this chapter we reported the results of a systematic survey of CPS studies that combine different self-adaptation mechanisms across the technology stack of the system. The results show that the majority of primary studies combine either MAPE with smart elements or MAS with smart elements in CPS architecture design. From the designs of the primary studies, we derived three patterns: SYNTHESIZE-UTILIZE, SYNTHESIZE-COMMAND, and COLLECT-ORGANIZE. These patterns offer problem-solution pairs to engineers for the design of future CPS and CPPS with self-adaptation capabilities, whereby the SYNTHESIZE-COMMAND pattern seems to be particularly relevant for the design of CPPS. Based on the survey results and the background of CIS, we described three emerging directions of how CIS are used in CPS and CPPS: capability augmentation, emergent machine-to-machine inter-

actions, and multi-disciplinary knowledge integration and coordination. We hope that the research results presented in this chapter can contribute to push forward the important field of CPPS in general and the application of self-adaptation to it in particular.

Acknowledgements This work was supported by the Christian Doppler Forschungsgesellschaft, the Federal Ministry of Science, Research and Economy, the National Foundation for Research, Technology and Development in Austria, and TU Wien research funds.

References

acatech - National Academy of Science and Engineering (2011) Cyber-Physical Systems: Driving force for innovation in mobility, health, energy and production. Tech. rep., URL http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Publikationen/Stellungnahmen/acatech_POSITION_CPS_Englisch_WEB.pdf

Barenji RV, Barenji AV, Hashemipour M (2014) A multi-agent RFID-enabled distributed control system for a flexible manufacturing shop. The International Journal of Advanced Manufacturing Technology 71(9):1773–1791

Bauernhansl T, ten Hompel M, Vogel-Heuser B (eds) (2014) Industrie 4.0 in Produktion, Automatisierung und Logistik. Springer Vieweg

Bures T, Gerostathopoulos I, Hnetynka P, Keznikl J, Kit M, Plasil F (2013) DEECO: An Ensemble-based Component System. In: Proc. of the 16th International ACM Sigsoft Symposium on Component-based Software Engineering (CBSE '13), ACM, pp 81–90

Bures T, Weyns D, Berger C, Biffl S, Daun M, Gabor T, Garlan D, Gerostathopoulos I, Julien C, Krikava F, Mordinyi R, Pronios N (2015) Software Engineering for Smart Cyber-Physical Systems - Towards a Research Agenda. ACM SIGSOFT Software Engineering Notes 40(6):28–32

Calinescu R, Grunske L, Kwiatkowska M, Mirandola R, Tamburrelli G (2011) Dynamic QoS Management and Optimization in Service-Based Systems. IEEE Transactions on Software Engineering 37(3):387–409

Cheng SW, Garlan D, Schmerl B (2006) Architecture-based Self-adaptation in the Presence of Multiple Objectives. In: Proc. of the International Workshop on Self-adaptation and Self-managing Systems (SEAMS '06), ACM, pp 2–8

De Wolf T, Holvoet T (2004) Emergence and Self- Organisation: a statement of similarities and differences. In: Proc. of the 2nd International Workshop on Engineering Self-Organising Applications, pp 96–110

De Wolf T, Holvoet T (2007) Design Patterns for Decentralised Coordination in Self-organising Emergent Systems. In: Brueckner SA, Hassas S, Jelasity M, Yamins D (eds) Engineering Self-Organising Systems - 4th International Workshop on Engineering Self-Organising Applications (ESOA '06), LNCS, vol 4335, Springer Berlin Heidelberg, pp 28–49

Di Marzo Serugendo G, Gleizes MP, Karageorgos A (2006) Self-Organisation and Emergence in MAS: An Overview. Informatica 30(1):45–54

Esfahani N, Malek S (2013) Uncertainty in Self-Adaptive Software Systems. In: De Lemos R, Giese H, Müller HA, Shaw M (eds) Software Engineering for Self-Adaptive Systems II, LNCS, vol 7475, Springer Berlin Heidelberg, pp 214–238

Fatima SS, Wooldridge M, Jennings NR (2006) Multi-Issue Negotiation with Deadlines. Journal of Artificial Intelligence Research 27(1):381–417

Fernandez-Marquez JL, Di Marzo Serugendo G, Montagna S, Viroli M, Arcos JL (2013) Description and composition of bio-inspired design patterns: a complete overview. Natural Computing 12(1):43–67

Garlan D (2010) Software Engineering in an Uncertain World. In: Proc. of the FSE/SDP Workshop on Future of Software Engineering Research (FoSER '10), ACM, pp 125–128

Garlan D, Cheng SW, Huang AC, Schmerl B, Steenkiste P (2004) Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure. Computer 37(10):46–54

Gupta A, Pandey OJ, Shukla M, Dadhich A, Ingle A, Gawande P (2014) Towards Context-aware Smart Mechatronics Networks: Integrating Swarm Intelligence and Ambient Intelligence. In: Proc. of the International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT '14), IEEE, pp 64–69

Heylighen F (2016) Stigmergy as a universal coordination mechanism I: Definition and components. Cognitive Systems Research 38:4–13

Hölzl M, Gabor T (2015) Continuous Collaboration: A Case Study on the Development of an Adaptive Cyber-Physical System. In: Proc. of the IEEE/ACM 1st International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS '15), IEEE, pp 19–25

Hu G, Tay W, Wen Y (2012) Cloud Robotics: Architecture, Challenges and Applications. IEEE Network 26(3):21–28

Jazdi N, Maga C, Göhner P, Ehben T, Tetzner T, Löwen U (2010) Improved Systematisation in Plant Engineering and Industrial Solutions Business - Increased Efficiency through Domain Engineering. at-Automatisierungstechnik 58(9):524–532

Juziuk J, Weyns D, Holvoet T (2014) Design Patterns for Multi-agent Systems: A Systematic Literature Review. In: Shehory O, Sturm A (eds) Agent-Oriented Software Engineering, Springer Berlin Heidelberg, pp 79–99

Kephart JO, Chess DM (2003) The Vision of Autonomic Computing. Computer 36(1):41–50

Kit M, Gerostathopoulos I, Bures T, Hnetynka P, Plasil F (2015) An Architecture Framework for Experimentations with Self-Adaptive Cyber-Physical Systems. In: Proc. of the IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '15), IEEE, pp 93–96

Kitchenham BA, Charters S (2007) Guidelines for performing Systematic Literature Reviews in Software Engineering. Tech. Rep. EBSE-2007-01, Software Engineering Group, School of Computer Science and Mathematics, Keele University, UK, and Department of Computer Science, University of Durham, UK, URL `http://www.cs.auckland.ac.nz/~norsaremah/2007GuidelinesforperformingSLRinSEv2.3.pdf`

Kitchenham BA, Budgen D, Pearl Brereton O (2011) Using mapping studies as the basis for further research A participant-observer case study. Information and Software Technology 53(6):638–651

Kramer J, Magee J (2007) Self-Managed Systems: An Architectural Challenge. In: Future of Software Engineering (FOSE '07), IEEE Computer Society, pp 259–268

Kumar N, Singh M, Zeadally S, Rodrigues JJPC, Rho S (2015) Cloud-Assisted Context-Aware Vehicular Cyber-Physical System for PHEVs in Smart Grid. IEEE Systems Journal PP(99):1–12

Leitão P (2009) Agent-based Distributed Manufacturing Control: A State-of-the-art Survey. Engineering Applications of Artificial Intelligence 22(7):979–991

Mahdavi-Hezavehi S, Avgeriou P, Weyns D (2016) A Classification Framework of Uncertainty in Architecture-Based Self-Adaptive Systems With Multiple Quality Requirements. In: Mistrik I, Ali N, Kazman R, Grundy J, Schmerl B (eds) Managing Trade-offs in Adaptable Software Architectures, Morgan Kaufmann, pp 45–78

Mamei M, Menezes R, Tolksdorf R, Zambonelli F (2006) Case Studies for Self-organization in Computer Science. Journal of Systems Architecture 52(8-9):443–460

Meszaros G, Doble J (1997) A Pattern Language for Pattern Writing. In: Martin RC, Riehle D, Buschmann F (eds) Pattern Languages of Program Design 3, Addison-Wesley Longman Publishing Co., Inc., pp 529–574

Minsky NH, Murata T (2004) On Manageability and Robustness of Open Multi-agent Systems. In: Lucena C, Garcia A, Romanovsky A, Castro J, Alencar PSC (eds) Software Engineering for Multi-Agent Systems II, LNCS, vol 2940, Springer Berlin Heidelberg, pp 189–206

Muccini H, Sharaf M, Weyns D (2016) Self-adaptation for Cyber-physical Systems: A Systematic Literature Review. In: Proc. of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '16), ACM, pp 75–81

Mukherjee S, Chaudhury S (2013) C-MAP: Framework for Multi-agent Planning in Cyber Physical Systems. In: Proc. of the 5th International Conference on Pattern Recognition and Machine Intelligence (PReMI '13), Springer Berlin Heidelberg, LNCS, vol 8251, pp 237–242

Musil A, Musil J, Biffl S (2016a) Major Variants of the SIS Architecture Pattern for Collective Intelligence Systems. In: Proc. of the 21st European Conference on Pattern Languages of Programs (EuroPLoP '16), ACM, pp 30:1–30:11

Musil A, Musil J, Biffl S (2016b) Towards Collective Intelligence System Architectures for Supporting Multi-Disciplinary Engineering of Cyber-Physical Production Systems. In: Proc. of the 1st International Workshop on Cyber-Physical Production Systems (CPPS '16), IEEE, pp 1–4

Musil A, Musil J, Weyns D, Bures T, Muccini H, Sharaf M (2016c) Protocol for: Patterns for Self-Adaptation in Cyber-Physical Systems - A Systematic Mapping Study. Tech. Rep. IFS-CDL 16-02, Vienna University of Technology, URL `http://qse.ifs.tuwien.ac.at/publication/IFS-CDL-16-02.pdf`

Musil J, Musil A, Biffl S (2015a) Introduction and Challenges of Environment Architectures for Collective Intelligence Systems. In: Weyns D, Michel F (eds) Agent Environments for Multi-Agent Systems IV, LNCS, vol 9068, Springer International Publishing, pp 76–94

Musil J, Musil A, Biffl S (2015b) SIS: An Architecture Pattern for Collective Intelligence Systems. In: Proc. of the 20th European Conference on Pattern Languages of Programs (EuroPLoP '15), ACM, pp 20:1–20:12

Musil J, Musil A, Weyns D, Biffl S (2015c) An Architecture Framework for Collective Intelligence Systems. In: Proc. of the 12th Working IEEE/IFIP Conference on Software Architecture (WICSA '15), IEEE, pp 21–30

Nasri M, Farhangi H, Palizban A, Moallem M (2012) Multi-Agent Control System for Real-time Adaptive VVO/CVR in Smart Substation. In: Proc. of the IEEE Electrical Power and Energy Conference (EPEC '12), IEEE, pp 1–7

Oreizy P, Medvidovic N, Taylor RN (1998) Architecture-based Runtime Software Evolution. In: Proc. of the 20th International Conference on Software Engineering (ICSE '98), IEEE Computer Society, pp 177–186

Patikirikorala T, Colman A, Han J, Wang L (2012) A Systematic Survey on the Design of Self-adaptive Software Systems Using Control Engineering Approaches. In: Proc. of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '12), IEEE, pp 33–42

Perez-Palacin D, Mirandola R (2014) Uncertainties in the Modeling of Self-adaptive Systems: A Taxonomy and an Example of Availability Evaluation. In: Proc. of the 5th ACM/SPEC International Conference on Performance Engineering (ICPE '14), ACM, pp 3–14

Ramirez AJ, Cheng BHC (2010) Design Patterns for Developing Dynamically Adaptive Systems. In: Proc. of the ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '10), ACM, pp 49–58

Ramirez AJ, Jensen AC, Cheng BHC (2012) A Taxonomy of Uncertainty for Dynamically Adaptive Systems. In: Proc. of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '12), IEEE, pp 99–108

Sarma S, Muck T, Shoushtari M, BanaiyanMofrad A, Dutt N (2016) Cross-layer Virtual/Physical Sensing and Actuation for Resilient Heterogeneous Many-core SaCs. In: Proc. of the 21st Asia and South Pacific Design Automation Conference (ASP-DAC '16), IEEE, pp 395–402

Vogel-Heuser B, Diedrich C, Pantförder D, Göhner P (2014) Coupling heterogeneous production systems by a multi-agent based cyber-physical production system. In: Proc. of the 12th IEEE International Conference on Industrial Informatics (INDIN '14), IEEE, pp 713–719

Wan J, Zhang D, Zhao S, Yang LT, Lloret J (2014) Context-Aware Vehicular Cyber-Physical Systems with Cloud Support: Architecture, Challenges, and Solutions. IEEE Communications Magazine 52(8):106–113

Wang FY (2010) The Emergence of Intelligent Enterprises: From CPS to CPSS. IEEE Intelligent Systems 25(4):85–88

Weyns D (2010) Architecture-Based Design of Multi-Agent Systems. Springer Berlin Heidelberg

Weyns D (2017) Software Engineering of Self-Adaptive Systems: An Organised Tour and Future Challenges. In: Taylor R, Kang KC, Cha S (eds) Handbook of Software Engineering, Springer, (to appear)

Weyns D, Ahmad T (2013) Claims and Evidence for Architecture-Based Self-adaptation: A Systematic Literature Review. In: Proc. of the 7th European Conference on Software Architecture (ECSA '13), Springer-Verlag, pp 249–265

Weyns D, Georgeff M (2010) Self-Adaptation Using Multiagent Systems. IEEE Software 27(1):86–91

Weyns D, Boucké N, Holvoet T (2008) A field-based versus a protocol-based approach for adaptive task assignment. Autonomous Agents and Multi-Agent Systems 17(2):288–319

Weyns D, Malek S, Andersson J (2012) FORMS: Unifying Reference Model for Formal Specification of Distributed Self-adaptive Systems. ACM Transactions on Autonomous and Adaptive Systems 7(1):8:1–8:61

Weyns D, Iftikhar MU, Söderlund J (2013a) Do External Feedback Loops Improve the Design of Self-adaptive Systems? A Controlled Experiment. In: Proc. of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '13), IEEE, pp 3–12

Weyns D, Schmerl B, Grassi V, Malek S, Mirandola R, Prehofer C, Wuttke J, Andersson J, Giese H, Göschka KM (2013b) On Patterns for Decentralized Control in Self-Adaptive Systems. In: de Lemos R, Giese H, Müller HA, Shaw M (eds) Software Engineering for Self-Adaptive Systems II, LNCS, vol 7475, Springer Berlin Heidelberg, pp 76–107

Winkler D, Musil J, Musil A, Biffl S (2016) Collective Intelligence-Based Quality Assurance: Combining Inspection and Risk Assessment to Support Process Improvement in Multi-Disciplinary Engineering. In: Kreiner C, O'Connor RV, Poth A, Messnarz R (eds) Systems, Software and Services Process Improvement: Proc. of the 23rd European System, Software & Service Process Improvement & Innovation Conference (EuroSPI '16), CCIS, vol 633, Springer International Publishing, pp 163–175

Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) Experimentation in Software Engineering. Springer Berlin Heidelberg

Wooldridge M (2001) Multi-agent systems: an introduction. Wiley

Xiong G, Zhu F, Liu X, Dong X, Huang W, Chen S, Zhao K (2015) Cyber-physical-social System in Intelligent Transportation. IEEE/CAA Journal of Automatica Sinica 2(3):320–333

Yu C, Jing S, Li X (2012) An Architecture of Cyber Physical System based on Service. In: Proc. of the International Conference on Computer Science and Service System (CSSS '12), IEEE, pp 1409–1412