Continuous Adaptation Management in Collective Intelligence Systems

Angelika Musil[1,2][0000−0002−1025−1626], Juergen Musil[1][0000−0002−2163−3603], Danny Weyns[2,3][0000−0002−1162−0817], and Stefan Biffl[1]

[1] Christian Doppler Lab SQI, Inst. of Inf. Systems Engineering, TU Wien, Austria
{angelika,jmusil}@computer.org, stefan.biffl@tuwien.ac.at

[2] Dep. of Comp. Science, KU Leuven, Belgium / Linnaeus University, Sweden
danny.weyns@kuleuven.be

Abstract. Collective Intelligence Systems (CIS), such as wikis and social networks, enable enhanced knowledge creation and sharing at organization and society levels. From our experience in R&D projects with industry partners and in-house CIS development, we learned that these platforms go through a complex evolution process. A particularly challenging aspect in this respect represents uncertainties that can appear at any time in the life-cycle of such systems. A prominent way to deal with uncertainties is adaptation, i.e., the ability to adjust or reconfigure the system in order to mitigate the impact of the uncertainties. However, there is currently a lack of consolidated design knowledge of CIS-specific adaptation and methods for managing it. To support software architects, we contribute an architecture viewpoint for continuous adaptation management in CIS, aligned with ISO/IEC/IEEE 42010. We evaluated the viewpoint in a case study with a group of eight experienced engineers. The results show that the viewpoint is well-structured, useful and applicable, and that its model kinds cover well the scope to handle different CIS-specific adaptation problems.

Keywords: Collective intelligence systems · Adaptation · Architecture viewpoint

1 Introduction

In the last decades, *Collective Intelligence Systems* (CIS), such as wikis, social networks, and media-sharing platforms, enable enhanced knowledge creation and sharing at organization and society levels alike. Today, CIS are widely adopted and influence a large number of people in their daily lives. Established CIS platforms have a longevity well over a decade and beyond. Consequently, CIS represent a significant system domain to research from different perspectives.

A CIS is a *complex socio-technical multi-agent system* that realizes environment-mediated coordination based on bio-inspired models in order to create a *perpetual cycle* of knowledge and information *aggregation and dissemination* among its agents (*actors*) [12, 18]. The system is heavily driven by its actors who continuously contribute content to a network of information artifacts [15] (*CI artifacts*), which represents the coordinative substrate and is hosted by an adaptive system layer that handles processing [17, 23] of aggregated content (*monitoring, analysis, and information filtering*) and information dissemination (using *rules, triggers, and notifications*). This feedback loop between the actor base and the computational system is an essential feature of a CIS and must be carefully designed and maintained and may not be underestimated.

From extensive experience in R&D projects with industry partners and in-house CIS development, we learned that these platforms typically go through a complex evolution process during which they mature, leading to a significant increase of user base size and accumulated content. Thereby, a particular challenge for software architects represents the multiple inherent uncertainties which continuously affect the system. In particular, when designing CIS the available knowledge is not adequate to anticipate all potential changes due to dynamics in the system context, such as changes of conditions, requirements, resources, or the emergence of new requirements and factors to consider. One way to deal with and mitigate the impact of uncertainties is to design systems that adapt or can be adapted when the missing knowledge becomes available [10].

Recent efforts to support software architecture aspects of CIS comprise an architecture pattern as foundation of CIS [13], a reference architecture [17], an architecture framework [14], and an architecture description language [3]. A particular challenging aspect with regard to evolution represents *adaptation of CIS*, which is a multi-dimensional problem that spans the full life-cycle of such platforms. However, the aspect of adaptation has not yet been investigated from a CIS architecture perspective. Traditional adaptation approaches that are applicable to common software system concerns in CIS are not directly applicable to CIS-domain-specific concerns. Examples include adaptation elements in the information dissemination phase of the feedback loop, when in the CIS life-cycle should adaptation activities be performed, or how to address uncertainties effecting the significant CIS perpetual cycle. Based on experiences from stakeholders in industry and our own experiences with studying and developing CIS, we identified a lack of consolidated design knowledge about the adaptation solution space specific to these systems. Current practice in the CIS domain showed that adaptation in CIS is added in an ad-hoc manner as a reaction to certain major incidents, such as rapid decrease of user activities or spam information generated by bots. However, incorporating adaptation mechanisms in an ad-hoc way may lead to unpredictable consequences on the system and unintended system behavior. Furthermore, there is a lack of methods to support software architects to address CIS-specific adaptation with reasonable effort and systematically design, describe and plan it.

To address these challenges, we study the *what*, *when*, and *how* of continuous adaptation management in the CIS domain. Our goal is to provide software architects with CIS-specific adaptation decision-making and management capabilities during the evolution of a CIS software architecture. To achieve this goal, we applied an empirically grounded research approach. We started with a survey of existing CIS to identify if adaptation is a relevant concern and what kind of adaptation is handled in practice. In addition, we reviewed literature regarding research work on adaptation-related concerns and specifics with focus on CIS. Next, we conducted a series of in-depth interviews with companies that have successfully built and operate CIS in order to identify their problems and challenges and to collect best practices on adaptation management in CIS. The collected data provided input for the identification of relevant stakeholders, their concerns during architecture design and requirements for architectural models to address these CIS-specific concerns. Based on the consolidated data and synthesized knowledge, we developed a novel architecture viewpoint, which provides an adaptation-specific view on CIS architectures and is implementation agnostic. The *Continuous Adaptation Management Viewpoint (CIS-ADAPT)* comprises four model kinds and aims at supporting software architects across the CIS life-cycle with a particular focus on the adaptation areas of modeling, scoping, binding time, and evolution of CIS. To evaluate the viewpoint's applicability and usefulness, we conducted a case study with eight experienced engineers.

The remainder of this paper is structured as follows: Section 2 summarizes related work. Section 3 describes the research question and methodology we followed. Section 4 presents the proposed architecture viewpoint with its model kinds. Section 5 describes a case study we used to evaluate the viewpoint's applicability and usefulness. Finally, Section 7 concludes and suggests future work.

2 Related Work

To the best of our knowledge, CIS-specific adaptation has not been the focus of previous research work. Hence, we discuss a selection of representative work on architecture-based adaptation and related architecture approaches in general.

Architecture-based adaptation [16, 9] is an established approach to engineer adaptive systems that focuses on the central role of software architecture in such systems through abstraction and separation of concerns. Two fundamental concerns of adaptive systems are domain concerns that are handled by the managed subsystem (that realizes the system functionality) and adaptation concerns that are handled by the managing subsystem (i.e., quality concerns about the managed subsystem) [25]. A key approach to realize the managing subsystem is by means of a so called *MAPE* feedback loop (Monitor-Analyze-Plan-Execute) [8]. One well-known architecture-based self-adaptive framework is *Rainbow* [5]. This approach uses an abstract architectural model to monitor software system run time specifications, evaluates the model for constraint violations, and if required, performs global and module-level adaptations. The reference model *FORMS* [26] (FOrmal Reference Model for Self-adaptation) provides a vocabulary for describing and reasoning about the key architectural characteristics of distributed self-adaptive systems and their concerns.

To support reuse of known solutions, [27] consolidated a number of design approaches for decentralized control in self-adaptive systems in form of MAPE patterns. The authors discussed drivers for the design of self-adaptive systems when choosing one of these MAPE patterns (e.g., optimization, scalability, ro-

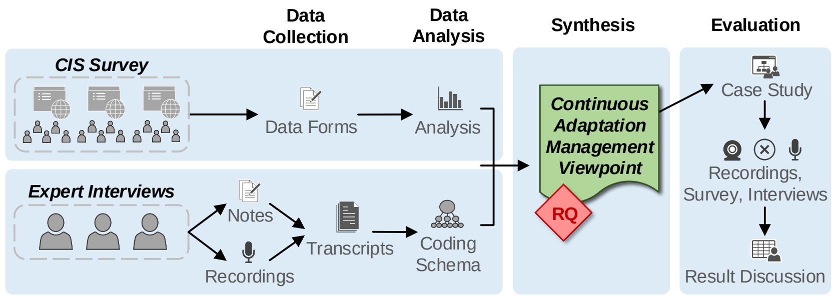


Fig. 1. Applied multi-phase research method

bustness). [19] presented twelve adaptation-oriented design patterns that are collected from literature and open sources projects. These patterns are clustered around monitoring, decision-making, or reconfiguration. The patterns are at the level of software design in contrast to our architecture-centric perspective that we adopt in this work.

One architecture viewpoint related to our work is the variability viewpoint presented in [4]. However, the focus of that viewpoint was on enterprise software systems and variability in general. Furthermore, [24] presented an approach to augment an architecture viewpoint with a particular variability viewpoint. Although both viewpoints follow ISO/IEC/IEEE42010 [7], they focus on variability concerns but do not consider binding times and system domain specifics.

In conclusion, the proliferation of domain-specific adaptation approaches continues, since the high degree of domain knowledge and complexity widens the gap between general purpose adaptation approaches and the required additional efforts of practitioners to make these approaches work and sustainably manage in specific application domain like CIS.

3 Research Methodology

The main objective of this research is to improve the architectural understanding of CIS and in particular to consolidate design knowledge on adaptation in CIS in order to support software architects to handle it. Based on experiences of stakeholders in the field that built and operate CIS and our own experiences with studying and developing CIS, we identified the following research question: *What are architectural principles to handle CIS-specific adaptation along its life-cycle and how can we codify these principles in a systematic way to make them useful and applicable for software architects?*

To answer this research question, we applied an empirically grounded research method, shown in Fig. 1. We performed a *survey of existing CIS* and a series of *semi-structured interviews* with software architects and senior software engineers of different CIS companies. In the next step the analyzed results and derived knowledge were consolidated in form of an architecture viewpoint for continuous adaptation management in CIS following the ISO/IEC/IEEE 42010

Table 1. Identified adaptation types with examples of elements and their option space

Adaptation Type	Adaptation Element Examples	Element Adaptation Option Examples
Actor	Role & privilege	Editor, administrator, moderator
	Application client	Desktop, web, app, messenger
Aggregation	Artifact attribute	Category, review, votes, tags, comments, actor views
	Interaction rule	Adding, commenting, up-voting, tagging
Processing	Monitoring mechanism	Hot topics monitoring, abnormal behavior monitoring
	Information filtering mechanism	Recommender system, artifact changes, actor activities
Dissemination	Trigger mechanism	Email, app message, on-site notification
	Dissemination rule	Monthly digest, daily report, weekly recommendations

standard [7]. Finally, we evaluated the usefulness and applicability of the proposed viewpoint by conducting a case study with experienced engineers who used the viewpoint to perform adaptation-specific design tasks in CIS key elements. More details, generated material and results of the research activities are available online [11]. In the remainder of this section, we briefly summarize the survey and interviews. The viewpoint and its evaluation are presented in the following sections.

CIS Survey. To investigate different types of CIS-specific adaptation that address key elements and processes in various CIS application contexts, we conducted a system survey based on a defined protocol describing the search strategy, selection and system quality assessment criteria, data extraction process, and data analysis methods. In total, we identified around 100 different CIS based on searches from different sources, such as the web-traffic rankings from Alexa[3], Wikipedia, digital libraries of scientific work, and domain experts from research and industry. We selected 30 CIS based on the quality of the available material to assess the system, including design documentation, user guide, and API specification. We collected data by exploring interaction workflows from an end-user perspective and reviewing available system design and documentation material. Based on subsequent analysis of the collected data and material, we derived initial information about characteristic adaptation points in CIS key elements and processes. Table 1 summarizes the main outcome of the survey in terms of adaptation types in CIS and their refinements.

Expert Interviews. Based on the survey results, we conducted interviews with 10 technical stakeholders covering a variety of roles in CIS engineering, e.g., CTO, software architect, senior engineer, and product manager. The participants come from different Austrian and US companies and organizations that operate a CIS platform in various application domains including medical, societal networking, employer/platform review & rating, and video/music sharing. The participants had 2-10 years experience with CIS engineering and operation. The goal of the interviews was to obtain additional data about stakeholders and adaptation concerns, rationales for adaptation design, and life-cycle aspects. The main selection criteria for participants was their experience in the CIS domain. By applying the guidelines by Hove and Anda [6] and Seaman [22], we designed

[3] http://www.alexa.com/topsites/global (last visited at 02/25/2019)

semi-structured interviews with a combination of specific and open-ended questions (e.g., What are the features of your system that have changed over time? What was your intention of these changes?). We asked them about the different phases they have gone through since the beginning of their software platform and challenges and difficulties they faced during design and engineering activities. The last part dealt with their experiences with respect to platform evolution and CIS-specific adaptation, adaptation management challenges and practices as well as the decision-making process. Each interview took about 50 minutes and was recorded for analysis. For data analysis, we applied coding [21] and grounded theory [2] to transform, structure, and analyze the interview transcripts. The findings of the interviews confirmed and complemented the previous results from the survey and revealed how designing and planning CIS-specific adaptation over the system's life-cycle was managed.

One particular insight is that in later stages changes to adaptation are handled less often than in the beginning and only in a conservative way in order to prevent negative effects on the system's behavior and success. So it is essential to consider the right timing for a CIS's evolution and when to introduce new adaptation elements and options. Changes in CIS-specific elements can have a significant impact on the behavior of the system and consequently on the behavior of the actors.

4 Continuous Adaptation Management Viewpoint

From the data collection and analysis discussed in the previous section, we defined the *architecture viewpoint for continuous adaptation management in collective intelligence systems (CIS-ADAPT)* which unifies CIS-specific aspects with established adaptation approaches. The viewpoint frames the essential concerns of stakeholders with an interest in handling CIS-specific adaptation across the system's life cycle, starting from its inception and during its operation. The viewpoint defines a set of four model kinds for identifying, designing and realizing adaptation in CIS key elements. It is important to note that the focus of this viewpoint is on CIS-specific adaptation and its impact on the system architecture. As such, architects may use additional architectural approaches, such as additional viewpoints or patterns, to deal with adaptation in traditional software system elements and other stakeholder concerns. The architecture viewpoint is structured using the template of the ISO/IEC/IEEE 42010 standard [7].

Table 2 shows an overview of the identified stakeholders and their adaptation concerns addressed by this viewpoint. This viewpoint particularly focuses on the technicalities of adaptation management in CIS, which are no direct concerns of system users, who contribute continuously to it. Thus the users are no stakeholders in terms of this viewpoint, but they are certainly affected by the design decisions made by applying this viewpoint.

The viewpoint comprises four model kinds presented in Tables 3 and 4: *adaptation types*, *adaptation definition*, *adaptation in time* and *adaptation workflow.*

Table 2. Continuous Adaptation Management Viewpoint for CIS - Overview

Overview: The architecture viewpoint deals with the main stakeholder concerns related to the continuous management of CIS-specific adaptation and defines models for the identification, design and realization of adaptation elements and their space of possible options across the system's life-cycle. The models show the relevant architectural information that is essential to guide a successful preparation for anticipated changes in the system's environment or requirements.
Stakeholders: *Architect(s)* who design and describe the CIS architecture and identify the commonalities and the adaptation space in the system. *Owner(s)* who define the CIS's purpose and business goals and operate it to provide the service to the users. *Manager(s)* who are responsible for overseeing CIS operation. *Analyst(s)* who assess the performance of a CIS in terms of quality criteria.
Concerns: *C1 - Adaptation Identification:* How can adaptation be exploited to enhance the operation of a CIS? What are possible adaptation elements in a CIS? What are the implications of adaptation elements in the design of a CIS? *C2 - Adaptation Management:* What options are available to resolve an adaptation element? What are the effects of different options? What are dependencies between different adaptation elements and options? When are adaptation elements resolved? Who is responsible for handling the adaptation and selecting adaptation options? *C3 - Adaptation Evolution:* When are adaptation activities be performed in the CIS life-cycle? How does adaptation influence the CIS evolution?

Adaptation Types Model Kind. This model kind describes the subject of adaptation, comprising four CIS-specific adaptation types along with adaptation elements: (1) *Actor*, (2) *Aggregation*, (3) *Processing*, and (4) *Dissemination*, e.g., an adaptation element of the type *Actor* is *Incentive Mechanism.* Concrete options of this adaptation element can be: awarding badges, up-votes, and likes. Concrete options for adaptation element *Dissemination Rule* of type *Dissemination* are artifact change reports, weekly digests, monthly personal recommendations. This model kind supports architects with defining what adaptation types and adaptation elements are relevant to implement in the context of the specific CIS-of-interest based on the concretely identified adaptation types.[4]

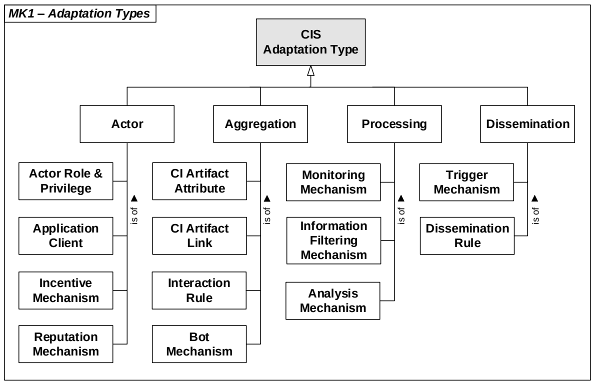
Adaptation Definition Model Kind. This model kind describes *what* adaptation is. It defines the possible adaptation options of an adaptation element, i.e., the adaptation space, each option representing a particular setting of the element. An adaptation element and its adaptation options are subject to constraints, i.e., they can exclude one another or may have dependencies, e.g., only actors with editor role can activate an artifact protection mechanism. A CIS element adaptation option can be optional or mandatory. Adaptation is then defined as addressing uncertainties by selecting adaptation options for

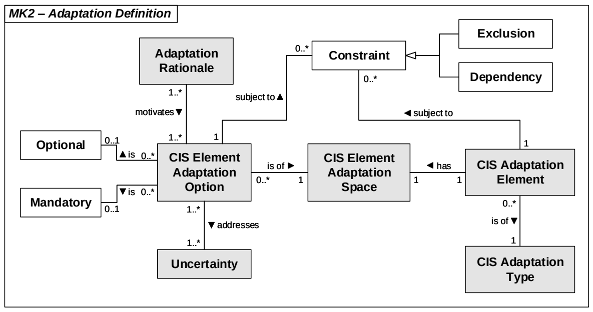
[4] Gray shaded boxes in model kinds represent links between multiple model kinds.

Table 3. Continuous Adaptation Management Viewpoint for CIS - Model Kinds

Model Kinds:
MK1 - Adaptation Types (deals with concern C1): A model that describes *where* adaptation can likely be achieved in a CIS to address uncertainties by identifying potential points of adaptation in CIS-specific system areas along with possible alternatives.
MK2 - Adaptation Definition (deals with concern C2): A model that clarifies *what* adaptation is about in the CIS-of-interest and describes details about the adaptation elements selected for adaptation, the associated element adaptation space of options to address particular uncertainties, and what constraints are applied on their relations.
MK3 - Adaptation in Time (deals with concern C3): A model that describes *when* adaptation activities are applied by responsible entities and how adaptation evolves across the CIS's life-cycle.
MK4 - Adaptation Workflow (deals with concern C2): A model that describes *how* the adaptation elements are realized and resolved, and who is responsible for selecting adaptation options and triggering the changes.

Metamodels:



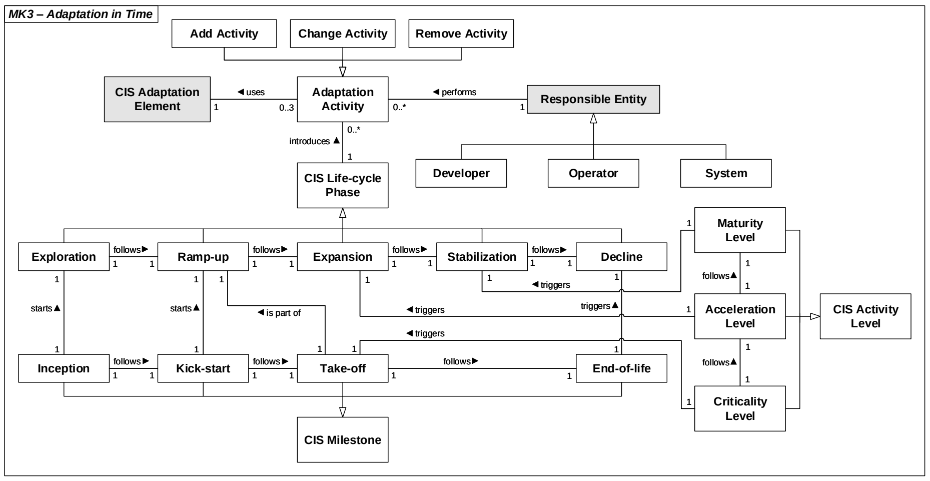


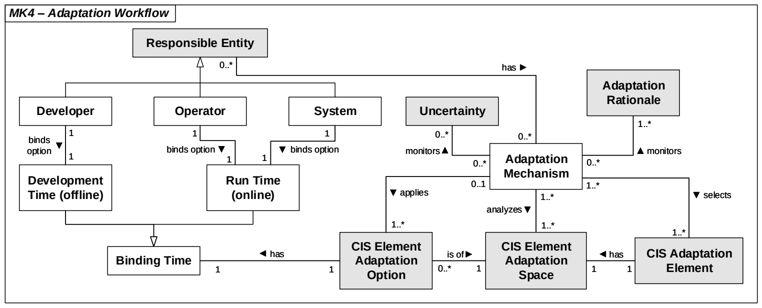
Key: UML

elements according to the adaptation rationales (goals). For instance, a lack of actor attention for specific artifacts observed during operation (uncertainty) may be handled by activating an awareness trigger (adaptation option) to increase contributions to these artifacts (rationale).

Table 4. Continuous Adaptation Management Viewpoint for CIS - Model Kinds

Metamodels:





Key: UML

Analyses:
A1 - Adaptation Effect Analysis (using MK1 and MK2): Assesses the effects of different adaptation option selections on the activities of the system and the actor base using a set of scenarios.
A2 - Adaptation Option Conflict Analysis (using MK2, MK3 and MK4): Reviews the relations and dependencies between adaptation elements and their spaces of options that are simultaneously deployed and bound in different life-cycle stages.

Adaptation in Time Model Kind. Grounded on the life-cycle and timeline model for self-adaptive software systems [1], this model describes *when* adaptation can be applied throughout a CIS's life-cycle in five phases: (1) *Exploration* phase, (2) *Ramp-up* phase, (3) *Expansion phase*, (4) *Stabilization* phase, and (5) *Decline* phase. Besides the phases, we identified characteristic milestones that a CIS can achieve and activity levels to reach. The *exploration* phase starts with the *inception* of the design and building of a first version of the system-of-interest. Then the *ramp-up* phase is triggered by the *kick-start* milestone of the official launch of the system-of-interest. During this phase the CIS can reach another milestone when the number of active users and generated content suddenly takes-off. This *take-off* is triggered by reaching a certain *level of criticality*. Then the *expansion* phase is triggered by reaching a certain *level of acceleration*. The *stabilization* phase is then triggered by reaching a certain *level of maturity*. Finally, the *decline* phase is triggered by reaching the *"end-of-life"* point.

Any responsible entity can perform adaptation activities, i.e., add, change, or remove activities to an adaptation element (by adapting its adaptation options) in different phases of the CIS's life-cycle. For instance, the operator introduces a monitoring mechanism aiming to identify irregular activities in expansion phase. This activity can be affected by reaching a certain CIS milestone (e.g., take-off milestone) or activity level (e.g., criticality level). If an option of an adaptation element is not relevant anymore, a responsible entity can remove it, e.g., the system may turn off a dissemination rule when user activity is increased over a period of time.

Adaptation Workflow Model Kind. This model kind describes *how* CIS-specific adaptations are realized. The adaptation workflow is realized by an adaptation mechanism associated with a responsible entity which can be a developer, an operator, or the system. A developer can apply adaptations offline (and deploy them on the running system), while an operator and the system can apply adaptations online. An adaptation mechanism realizes a feedback loop. The mechanism monitors uncertainties and checks whether the system complies with its goals (rationales). If the system goals may be jeopardized, the adaptation space of the adaptation elements is analyzed, i.e., the options available for adaptation, to mitigate the uncertainties. Based on this analysis, the adaptation mechanism selects adaptation options for adaptation elements. These options are then executed in the system.

Adaptation Effect Analysis. This analysis uses a set of scenarios to assess the effects of selecting different adaptation options on the behavior of the system and the actor base. The analysis results help identifying improvements of the adaptation elements and their adaptation options. The results can also provide insights in the conditions when selected options may improve or degrade the CIS behavior, e.g., in the form of increase/decrease of user activity. In the exploration and ramp-up phases, adaptation effect analysis can be done using simulation or via tool-assisted user testing. In later phases further approaches like A/B testing and/or feature toggles can be added to enable automated, data-driven processes for performance analysis, simulation and selection of adaptation options. Fig. 2 shows the effects of adaptation for a CIS pilot that we developed using a NetLogo analysis model. The graphs on the left show results when no dissemination is used. The graphs on the right show results when a slow-cycled global dissemination rule and a short-cycled actor-personalized dissemination rule are activated. The results show that the contribution distribution (top) got a steeper tail at the beginning with the dissemination rules activated, whereby the actor activity (bottom) remained unchanged.

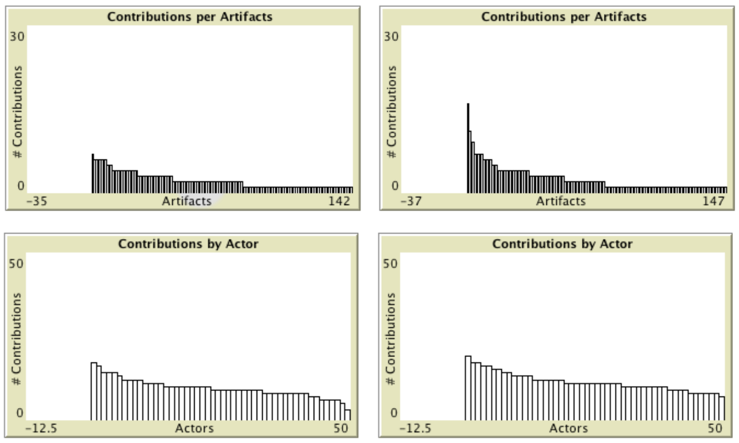


Fig. 2. Analysis results: none (left) or two (right) dissemination rules activated

Adaptation Option Conflict Analysis. This analysis performs a review of the relations and dependencies between adaptation elements, options, and adaptation elements and options that are simultaneously deployed and bound in the different stages of the CIS's life-cycle. The analysis results help to identify possible conflicts and inconsistencies between CIS adaptation elements/options that need to be resolved. In early stage phases, conflict detection and resolution can be performed manually by the architect by using the CIS-specific adaptation definition and workflow models. In later stage phases automated tool-support, such as feature-to-code traceability and consistency checking of the CIS adaptation models, is necessary to make conflict identification and resolution viable.

5 Evaluation of the Viewpoint

To obtain qualitative evidence of the usefulness and applicability of the CIS-ADAPT viewpoint, we performed an in-depth study with eight engineers without any experience in CIS design and development. Participants had between 1 and 7 years of industrial experience in software engineering/software architecture and are active in Austrian companies as project managers, software architects and software developers in various domains. To obtain qualitative data from different perspectives, criteria to select the participants include a mix of male and female engineers as well as a broad range of industry experience to get also insights into how less experienced engineers use the viewpoint.

We applied a *case study* design to plan our qualitative in-depth study and followed the guidelines for case studies in software engineering provided by Runeson et al. [20]. The concrete objective of the case study is answering the following questions: (1) *To what extent does the viewpoint support correct handling of CIS-specific adaptation problems?* (2)*How useful are the model kinds with regard to managing CIS-specific adaptation?*

Here we summarize the case study design and the results. For a detailed description and the evaluation material, we refer the interested reader to [11].

5.1 Case Study Design

In this case study the participants were instructed to apply the architecture viewpoint in three adaptation-related design tasks addressing CIS key elements of a given scenario. The case study was organized as a 6-hours session at TU Wien. We provided all participants with the same material to perform each task, including a general description of the CIS scenario, its domain and stakeholders, a set of pre-defined architecture models related to the particular view on CIS adaptation management which they had to extend or modify according to the tasks, and the viewpoint description with its model kinds and analyses.

Before starting with the design tasks, participants were introduced to CIS in general, software architecture concepts in the context of ISO/IEC/IEEE 42010, and the CIS-ADAPT architecture viewpoint. The participants were also introduced to the CIS scenario and questions were answered to avoid any misunderstanding of the assignment. After the first part, participants were asked to complete a short survey to gather their background information, including their education and experience with (CIS) software architecture design as well as adaptation handling in architecture design.

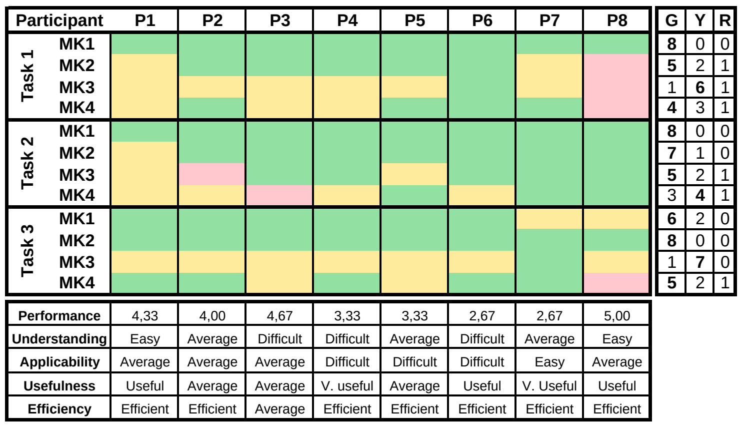
While the participants performed the design tasks, we video recorded their actions and progression to gather data how they used the viewpoint in the given scenario. At the end of the study session, we collected the modified architecture models and the participants were asked to complete a short survey to assess the applicability, usefulness and understandability of the applied architecture viewpoint and its model kinds. Finally, we conducted individual semi-structured interviews of about 10 minutes each to collect data about the participant's experiences and challenges during the application of the viewpoint.

We analyzed in total 14 hours of video material as well as the design models that the participants produced while accomplishing the given tasks to identify how they applied the viewpoint and used its model kinds and model elements. The survey results allowed us to better understand and reason about the usefulness and understandability of the viewpoint from an architect's perspective. Finally, the interviews provided us insights into the experiences and challenges the participants had to face as well as feedback for improvement.

5.2 Case Study Results

Eight participants completed 3 tasks, each of which required to use the 4 models of the viewpoint. In total each participant produced 12 models across all tasks, resulting in 96 models in total across all tasks and participants.

In task 1, participants extended the space of each of two pre-defined adaptation elements with a new element adaptation option. In task 2, participants modified an existing option from manual to automated application of the option at run time. In task 3, participants defined and introduced a new adaptation



Participant		P1	P2	P3	P4	P5	P6	P7	P8	G	Y	R
Task 1	MK1	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	**8**	0	0
	MK2	🟨	🟩	🟩	🟩	🟩	🟩	🟨	🟥	**5**	2	1
	MK3	🟨	🟨	🟨	🟨	🟨	🟩	🟨	🟥	1	**6**	1
	MK4	🟨	🟩	🟨	🟨	🟩	🟩	🟩	🟥	**4**	3	1
Task 2	MK1	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	**8**	0	0
	MK2	🟨	🟩	🟩	🟩	🟩	🟩	🟩	🟩	**7**	1	0
	MK3	🟨	🟥	🟩	🟩	🟨	🟩	🟩	🟩	**5**	2	1
	MK4	🟨	🟨	🟥	🟨	🟩	🟨	🟩	🟩	3	**4**	1
Task 3	MK1	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	**6**	2	0
	MK2	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	**8**	0	0
	MK3	🟨	🟨	🟨	🟨	🟨	🟨	🟩	🟨	1	**7**	0
	MK4	🟩	🟩	🟨	🟩	🟨	🟩	🟩	🟥	**5**	2	1

Performance	4,33	4,00	4,67	3,33	3,33	2,67	2,67	5,00
Understanding	Easy	Average	Difficult	Difficult	Average	Difficult	Average	Easy
Applicability	Average	Average	Average	Difficult	Difficult	Difficult	Easy	Average
Usefulness	Useful	Average	Average	V. useful	Average	Useful	V. Useful	Useful
Efficiency	Efficient	Efficient	Average	Efficient	Efficient	Efficient	Efficient	Efficient

Fig. 3. Overview of the results of 3 design tasks performed by 8 participants (G=Green: correct solutions; Y=Yellow: partially correct; R=Red: incorrect)

element to the system and added two options to its space.

From the created 96 models, 61 (63.5%) were solved correctly, 29 (30.2%) with some deviations, and only 6 (6.3%) models were incorrect. Fig. 3 shows an overview of the model defects across all participants and tasks. Hence, in the context of the design tasks, we can answer the first evaluation question (*To what extent does the viewpoint support correct handling of adaptation problems in CIS?*) positively. Nevertheless, some of the participants commented on the complexity of elements of the viewpoint, e.g., *"For me, this [MK3] was the most difficult model, because it has many aspects such as phases and milestones and all interact. Also, this one is more formal. For understanding, you can exactly see how level and phase and milestones are linked."* or *"The workflow model [MK4] was difficult, because it was not clear in the task description specifically when the resolving should actually happen."* In the following, we elaborate on the analysis of the usefulness of each model kind.

MK1. For the *adaptation types model*, 22 of 24 designs were performed without defects, 2 with defects, and none incorrect. The usefulness of MK1 was scored 2.5/5 on average by the participants. Positive feedback includes *"[...] the model provides a good overview about the adaptation types [...]"* and *"[...] it was useful to see the available choices that you have, also when it comes where to add new options and elements [...]"*. Some critical remarks were *"[...] I personally would map the types to my components so that they are not so generic like in the study scenario [...]"* and *"[...] the model was not really necessary for me, because its parts have been repeated already in model 2 [...]"*. In conclusion, the usefulness of MK1 for the tasks at hand is moderate, the opinions among participants differ.

MK2. For the *adaptation definition model*, 20 of 24 designs were performed without defects, 3 with defects, and 1 incorrect. The usefulness of MK2 scored 4.1/5 on average by the participants. Some of the positive feedback of the participants include *"[...] with regards to utility, the definition model was definitely the* *best."* and *"[...] the most helpful models for me have been models 2 and 3."* One rather negative comment but showing its criticality was *"The definition model was the most challenging for me because it was so central and the following models depend on it. [...] you cannot do much meaningful with the later models if you do not have the definition model straight."* In conclusion, MK2 was regarded as a central model and indicated as highly useful in the tasks at hand.

MK3. For the *adaptation in time model*, 7 of 24 designs were performed without defects, 15 with defects, and 2 incorrect. The usefulness of MK3 was scored 2.4/5 on average by the participants. One of the positive comments was *"Model 3 and 4 have been pretty useful, in particular if you have to consider the run time aspects. That was particularly useful."*. A critical comment was *"The model was tricky for me, because there is no definitive solution when there is the ideal point in time - you know, too early or too late [...]"*. In conclusion, MK3 was the worst performing model kind in terms of correct solutions. Regarding utility the average score was moderate for the given tasks at hand. One recurring comment was that the illustration of the CIS life-cycle that was used during the introduction session would be a beneficial add-on for the viewpoint, e.g., one participant commented *"The life-cycle diagram would make using this model easier. I redraw it from memory at the beginning so that I can better envision the life-cycle, instead of just relying on the model kind."*

MK4. For the *adaptation workflow model*, 12 of 24 designs were performed without defects, 9 with defects, and 3 incorrect. The usefulness of MK4 was scored 4/5 on average by the participants. One of the positive comments was *"The workflow model helps to create a more flexible system and you see clearly which risks are covered."* A critical comment was *"It was not always clear when it was run time and when it was development time. Also the dependencies between tasks were rather loose. I think sometimes you cannot sharply discriminate clearly between user tasks and system tasks, as it is suggested in the model kind."* In conclusion, MK4 has shown to be a very useful model for the tasks at hand.

6 Threats to Validity

We briefly discuss potential validity threats of this study and ways how they were mitigated.

Internal Validity. By using well-defined data extraction forms in the CIS survey and an interview guide, we attempted to conduct the study in a consistent and objective way to reduce the risk to affect the validity of the data provided by the study subjects. Especially during the interviews we needed to be very careful when giving specific examples so that we do not influence the given answers. For both data collection methods we performed a pilot study for improvement, e.g., to identify questions/data items that are confusing or do not provide enough informative quality. Also expert feedback was used to counter-check the consistency and integrity of the data collection instruments.

To address potential threats of misinterpretation of the collected data, the findings have been derived by two researchers and two additional experienced researchers cross-checked and validated the analysis results and conclusions. Furthermore, during the interviews we regularly summarized the given information and asked the participants to verify the correctness of the interpretation.

External Validity. The presented models are the result of an in-depth analysis of the gathered data but might be limited by the samples we investigated. To increase the generalization of the results to a broader context and strengthen the study results, we plan to conduct a CIS survey with a larger system sample and do more expert interviews. For the evaluation of the viewpoint, we performed a case study with eight participants. To enhance generalization of the results, this qualitative inquiry should be extended with additional cases in other domains.

7 Conclusion

In this paper, we presented an architecture viewpoint for continuous adaptation management in CIS, aligned with ISO/IEC/IEEE 42010. The viewpoint is intended to address CIS-specific adaptation concerns and existing limitations. It was designed to be compatible with other adaptation approaches so that our contribution represents a useful addition to domain-specific adaptation approaches. A qualitative evaluation with eight experienced engineers in a case study shows that the viewpoint is well-structured and particularly useful to handle different CIS-specific adaptation problems. In future work, we plan to refine the viewpoint and extend its evaluation. Furthermore, we plan to further develop the analysis part of the viewpoint and consider to develop tool support for it.

Acknowledgments. The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

References

1. Andersson, J., et al.: Software Engineering Processes for Self-Adaptive Systems. In: Software Engineering for Self-Adaptive Systems II, LNCS, vol. 7475, pp. 51–75. Springer, Berlin, Heidelberg (2013)
2. Corbin, J., Strauss, A.: Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Sage Publications, Inc., 3rd edn. (2007)
3. Dorn, C., Taylor, R.N.: Coupling Software Architecture and Human Architecture for Collaboration-Aware System Adaptation. In: Proc. Int. Conf. on Software Engineering. pp. 53–62. IEEE (2013)
4. Galster, M., Avgeriou, P.: A Variability Viewpoint for Enterprise Software Systems. In: Proc. of Joint WICSA/ECSA. pp. 267–271. IEEE Computer Society (2012)
5. Garlan, D., et al.: Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure. Computer **37**(10), 46–54 (2004)
6. Hove, S.E., Anda, B.: Experiences from Conducting Semi-structured Interviews in Empirical Software Engineering Research. In: Proc. 11th IEEE Int. Software Metrics Symposium. pp. 23–32. IEEE Computer Society (2005)

7. ISO/IEC/IEEE 42010: Systems and softw. engineering - Architecture descr. (2011)
8. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. Computer **36**(1), 41–50 (2003)
9. Kramer, J., Magee, J.: Self-Managed Systems: An Architectural Challenge. In: Future of Software Engineering. pp. 259–268. IEEE Computer Society (2007)
10. Mahdavi-Hezavehi, S., Avgeriou, P., Weyns, D.: A Classification Framework of Uncertainty in Architecture-Based Self-Adaptive Systems With Multiple Quality Requirements. In: Managing Trade-offs in Adaptable Software Architectures, pp. 45–77. Morgan Kaufmann (2017)
11. Musil, A., Musil, J., Weyns, D., Biffl, S.: Supplementary Material: Continuous Adaptation Management in Collective Intelligence Systems (2019), `http://qse.ifs.tuwien.ac.at/ci/material/pub/ecsa19/`
12. Musil, J., Musil, A., Biffl, S.: Introduction and Challenges of Environment Architectures for Collective Intelligence Systems. In: Agent Environments for Multi-Agent Systems IV, LNCS, vol. 9068, pp. 76–94. Springer International Publishing (2015)
13. Musil, J., Musil, A., Biffl, S.: SIS: An Architecture Pattern for Collective Intelligence Systems. In: Proc. 20th EuroPLoP. pp. 20:1–20:12. ACM (2015)
14. Musil, J., Musil, A., Weyns, D., Biffl, S.: An Architecture Framework for Collective Intelligence Systems. In: Proc. 12th WICSA. pp. 21–30. IEEE (2015)
15. Omicini, A., Ricci, A., Viroli, M.: Artifacts in the A&A Meta-model for Multi-agent Systems. Autonomous Agents and Multi-Agent Systems **17**(3), 432–456 (2008)
16. Oreizy, P., et al.: An Architecture-Based Approach to Self-Adaptive Software. IEEE Intelligent Systems **14**(3), 54–62 (1999)
17. Pääkkönen, P., Pakkala, D.: Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems. Big Data Research **2**(4), 166–168 (2015)
18. Parunak, H.V.D.: A Survey of Environments and Mechanisms for Human-Human Stigmergy. In: Environments for Multi-Agent Systems II, LNCS, vol. 3830, pp. 163–186. Springer (2006)
19. Ramirez, A.J., Cheng, B.H.C.: Design Patterns for Developing Dynamically Adaptive Systems. In: Proc. ICSE Workshop on Softw. Eng. for Adaptive and Self-Managing Systems. pp. 49–58. ACM (2010)
20. Runeson, P., Host, M., Rainer, A., Regnell, B.: Case Study Research in Software Engineering: Guidelines and Examples. Wiley Publishing, 1st edn. (2012)
21. Saldana, J.: The Coding Manual for Qualitative Researchers. Sage, 2nd edn. (2013)
22. Seaman, C.B.: Qualitative methods in empirical studies of software engineering. IEEE Transactions on Software Engineering **25**(4), 557–572 (1999)
23. Sumbaly, R., Kreps, J., Shah, S.: The Big Data Ecosystem at LinkedIn. In: ACM SIGMOD Conference. pp. 1–10. ACM (2013)
24. Tekinerdogan, B., Sözer, H.: Variability Viewpoint for Introducing Variability in Software Architecture Viewpoints. In: Proc. WICSA/ECSA Companion. pp. 163–166. ACM (2012)
25. Weyns, D.: Software Engineering of Self-adaptive Systems. In: Handbook of Software Engineering, pp. 399–443. Springer (2019)
26. Weyns, D., Malek, S., Andersson, J.: 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 (2012)
27. Weyns, D., et al.: On Patterns for Decentralized Control in Self-Adaptive Systems. In: Software Engineering for Self-Adaptive Systems II, LNCS, vol. 7475, pp. 76–107. Springer (2013)