A Design Science Approach to Collective Intelligence Systems

Alexander Kornrumpf
FernUniversität in Hagen
Universitätsstr. 41, 58097 Hagen, Germany
alexander.kornrumpf@fernuni-hagen.de

Ulrike Baumöl
FernUniversität in Hagen
Universitätsstr. 41, 58097 Hagen, Germany
ulrike.baumoel@fernuni-hagen.de

Abstract

The inverse problem of collective intelligence is the problem of designing a collective intelligence system that is fit to overcome a given challenge. In this paper we propose a design science approach to this problem. We carry out the first steps of the design science research process showing the relevance of the problem and the objectives of a tool supporting the solution. While this can be seen as progress towards an agenda and methodology for long-termed research, as an immediate result we also present a conceptual model of the collective intelligence domain which provides new insight into the ontology of collective intelligence. The conceptual model may inform the design of collective intelligence systems independent of the supporting tool. It also illustrates the difference between collective intelligence systems and traditional information systems.

1. Introduction

Recently, paralleled by the rise of the so called web 2.0, applications of collective intelligence (CI) are becoming increasingly popular and successful [1]. While a precise and universal definition of CI or “human swarm intelligence” is still lacking [2], many different real-world phenomena are regarded to be CI. Organizations that utilize such phenomena range from Google to the Wikipedia foundation [3]. As Ickler [4] points out, the hallmark of CI in the web 2.0 is the participation of users in a valuable or intelligent way, which may also include related concepts such as “crowdsourcing” [5], the “wisdom of crowds” [6] or “wikinomics” [7]. The potential and success of CI based business models can not only be observed in practice but is also subject of current research, e.g. in [8–10]. But, surprisingly, this line of research has focused almost exclusively on concepts like principles [10], properties [11] and “building blocks” [3] of CI rather than on the underlying human-computer systems which enable the emergence of CI in the web 2.0.

In this paper, we suggest to complement the existing CI research agenda by a design science approach focused on such human-computer systems which we call CI systems (CIS). We argue that gaining an increased ability to purposefully create CIS that solve specific tasks will produce valuable theoretical insight. In particular, supplying a human designer with the information she needs in order to design a successful CIS requires solid theoretical understanding of CIS in the first place. In the long run, we envision a system that, on a meta-level, supports the design of CIS as a solution to arbitrary problems through a generalized design process. Therefore, this paper is concerned with design in two ways. First, with an approach to generalize CIS design to the point where it can be supported by a computer system, and second, with the design of such a supporting system. It is with respect to the second point and not to any specific CIS that we frame our research question: What steps and concepts does the design of arbitrary CIS need to encompass?

In the remainder of this paper we proceed as follows. In the next section, first we clarify the basic terminology by providing definitions of CI and CIS from literature. We briefly discuss related work with a focus on CIS design and ontology. We then point out, how design science methodology can be applied to CIS design and how conceptual modeling can be regarded as a first step in that direction. We also introduce a language for conceptual modeling, taken from previous work, which is particular fit to be applied to CIS models. In the main section, we make a case for the computer aided design of CIS as an application of design science methodology. We show the relevance of inverse problem of CI and provide a solid understanding of what makes the problem “special”. Then we show what a tool needs to accomplish to be a valuable support in the solution of the problem. Finally we turn the findings into a conceptual model of CIS which may serve as the starting point for the design of such tool.
2. Background

2.1 Terminology

In the introduction we have already used the terms collective intelligence (CI) and collective intelligence systems (CIS). We now provide definitions for these terms from previous work that represent our understanding of CI and CIS throughout this paper.

Different perspectives on CI are discussed by Aulinger [12], who comes up with the following necessary conditions for CI to occur:

1. “A group has the capability to overcome challenges through shared or individual processing of information.” and
2. “This capability allows the group to come to results superior to the results that could have been reached by conventional methods or by one member of the group alone.”

These conditions together summarize, what is shared among the many CI definitions and therefore are used as a definition in this paper.

Defining CI by necessary conditions for the capabilities of groups shows that CI in itself is merely a potential. One way to realize this potential is the creation of an adequate human-computer system, a CIS. The distinction between CI and CIS is not always made explicitly in previous work. However, the following definition of CIS is consistent with the above definition of CI:

“[CIS are] human-computer systems in which machines enable the collection and harvesting of large amounts of human-generated knowledge, while enabling emergent knowledge, i.e., computation and inference over the collected information leading to answers, discoveries, or other results that are not found in the human contributions”[13].

In literature, there is an ongoing confusion about the terms CI and swarm intelligence (SI). This seems to be due to the fact that collective intelligence has been inspired by the apparently “intelligent” behavior of animal swarms. In this paper we understand CI according to the first of above definitions to be a general term that applies to groups (collectives) of animals and humans alike with no other conditions on these groups than the ones stated. But obviously CI that is enabled by CIS according to the second of above definitions refers to the computer supported CI of humans. In the following we will speak in short of CI when we actually refer to this specific form of CI. SI is a form of CI but it is a much more restricted term that requires additional “swarm characteristics” of the collective. These characteristics are mostly displayed by animal swarms, but there are border cases where human collectives can be viewed as behaving like swarms [2]. Here also we will refer to cases of human SI simply as SI since we show only little interest in the SI of animals, robots etc. in this paper.

2.2 Related Work

As indicated in the introduction, the majority of previous work on CI focuses on the analysis of existing CIS or CI phenomena. This stream of research is probably best represented by [3]. However, these existing applications of CI are designed to solve highly specific problems. It is unknown what the historical design process of these applications was and whether insights from it can be generalized to the design of other CIS. Therefore we discuss previous work under the assumption that an approach at generalizing the design of CIS has to be conceptual rather than retrospective.

Baumöl et al. [14] present an ontology of SI. Swarms, as opposed to general collectives, are characterized by the individuals following a limited set of rules. Two implications are of interest here: First, that individuals can be modeled as agents employing multi-agent theory and second, that swarms can be determined by constructive means.

Swarm engineering [15, 16] is the science of “forcing” swarms to display a desired global behavior by designing appropriate individual behavior. While this is successful in robotics, the possibilities to design individual behavior directly in a collective of humans are obviously very limited. We conclude that the issue can only be addressed indirectly by designing the CIS and the ways of interacting with it as to encourage and incentivize the desired behavior.

In contrast to this swarm centric approach, Gregg suggests designing for a data centric approach, i.e. viewing data as the main asset that is to be collected, shared and aggregated within an application whereas the role of the individual user comes down to “add to, modify or otherwise enhance the data to improve its usefulness” [17].

Ickler [4] presents an ontology for what he calls web-based collective intelligence. Within this ontology, both approaches are integrated by allowing for “CI of the connected” as a result of self-organization [of swarms] and “CI of the unconnected” as a result of aggregation [of data]. The ontology also includes the notions of motivation for individual activity and results thereof.

Taking motivation into account is important, because, as mentioned above, collectively intelligent behavior of humans may have to be incentivized. It has been argued e.g., that the human individuals can be outright reluctant to participate in prediction tasks, a major application of CI [18]. In the CI subfield of
crowdsourcing, Leimeister et al. [19] e.g., have conducted design research on incentivizing motives relevant to participation in ideas competition, which they have found to be “learning”, “direct compensation”, “self-marketing” and “social motives”.

From an ontological point of view, Georgi and Jung [20] make arguments very similar to those made in [4]. Their contribution to the ontology of CIS can be seen in additionally differentiating between passive CIS, i.e., such CIS where pre-existing individual activity is utilized, and active CIS, i.e., such CIS where new and active behavior is demanded of the individuals. They also elaborate a process view of CI by taking into account the task that needs to be solved, the individual contribution to this task and the resulting output.

One common point in all of the previous work mentioned in this section is that the different authors argue for the creation of new instances of CI or CIS to solve yet unsolved problems without being able to precisely state how to accomplish this. It is unsatisfying that while CI is a success story and many successful principles of CI are known or conjectured, very little is known about how to apply these principles to new problems by systematic design rather than intuition. This gap in the body of knowledge is the main motivation behind the research question of this paper.

2.3 Methodology

We follow a combination of the design science guidelines (DSG) as introduced in [21] and the design science research process (DSRP) as derived thereof in [22]. In this paper we concentrate on the initial two stages of the DSRP, i.e. with respect to the research question stated in the introduction, we identify a central problem in CIS research and define the objectives of a solution to that problem. Whilst doing so, according to the DSG, several things have to be accomplished: First, the relevance of the problem needs to be shown in terms of business needs and interest to the research community. Second, we need to show, how the artifact envisioned as a solution can be applied to the problem as to provide grounds on which the artifact can be evaluated. Third, research rigor requires that the envisioned artifact is grounded in existing knowledge and methodology. Finally we discuss how the design of such artifact is intended to inform theory and contribute to the body of knowledge.

Following the DSG we regard design to be an iterative process, so that the artifact we present in this paper is not the final artifact but a conceptual model as a solution to our present research question. In further iterations of the process, this conceptual model may inform the design of an actual system – an instantiation – which may then be used by CIS designers. Even though this seems to be a long way to go, communicating our work in a conceptual stage is also justified by the DSG.

The term conceptual model refers to the representation of a domain for an intended purpose [23]. As such it differs from a design model which refers to the representation of a software system [24]. Against this background, at first it seems a bit far-fetched to address conceptual modeling with design science methodology but the principles are very well compatible [25]. While much has been written about modeling, and the authors differ in minor details, we still consider it safe to follow [25] and conclude that models can be characterized by four dimensions: purpose, result of mapping, language and value.

We will get a detailed discussion of purpose and value of the model “for free” as a byproduct of following the DSG throughout this paper. At this point we want to keep things on a general level. Of the common purposes of conceptual models [26], our main intention lies in providing input for the design of a (software) system and helping to understand a domain. The software system we are speaking of here is the meta-system to which we refer in our research question whereas the domain which shall be addressed by this system is the domain of CIS. As a consequence, our research is structured as to produce a conceptual model of CIS. It is important to note that we are not proposing a design model of CIS in the aforementioned sense of the word. This can be understood by taking into account that a CIS by the definition given in the previous section is more than a software system. It is a “human-computer system” or more broadly phrased a socio-technical system (STS).

Roughly speaking, the core of conceptual modeling is the mapping of an ontology – or ideally “the ontic reality” – of a domain into a given modeling language but for the latter there exists no generally accepted choice [24, 26]. Well known and popular choices however are the entity-relationship (ER) language and the unified modeling language (UML) [27]. However the expressiveness of ER is rather limited and the more expressive UML has its intended use in design models rather than conceptual models which makes very careful adaption necessary. Fortunately, conceptual models often need to be only semi-formal [23] and for the STS domain there exists a semi-structured modeling language (“SeeMe”) [28]. We employ this language in this paper.

Figure 1 illustrates the basic semantics of the SeeMe modeling language [29]. In SeeMe, models consist of three types of objects: roles (oval shapes), activities (rounded boxes) and entities (boxes). A “mousehole” containing “…” indicates, that an object
needs to be further specified but we are currently unable to do so. The semantics of relationships between objects depend on the type of the respective objects and the direction of the relationship. There are nine types of relationships which we enumerate now, using the numbering depicted in Figure 1:

1. Roles have expectations towards other roles.
2. Roles carry out activities.
3. Roles are described by entities.
4. Activities have an influence on roles.
5. Activities form a sequence with other activities.
6. Activities create and manipulate entities.
7. Entities are owned by roles.
8. Entities are used by activities.

9. Entities form a hierarchy with other entities.

3. Computer Aided CIS Design

3.1 Problem Identification

Problem identification is the first step of the DSRP [22]. We depart at a very basic consideration: CI by definition is a collective overcoming a challenge (with additional conditions met). This is the classical forward perspective on CI as assumed by research on existing CIS. When it comes to the design of new CIS, this perspective however is of little help. What a designer really needs to know is how to transform a given challenge into a CIS that solves a problem representing this challenge. This changed perspective is called the inverse problem of CI [30]. The two perspectives are compared in Figure 2:

We refer to the introduction and related work sections of this paper when claiming that this is indeed a real-word business problem because the business potential of CI is undisputed among researchers and we observe real businesses implementing CI. What makes the inverse problem relevant for research is that it is not easily solved in general. Despite the multitude of research on existing CIS, no evidence can be found, that productive real-world systems have been deliberately designed against the CI properties identified in said research, let alone being designed following a systematic design process. Successful CIS design still seems to be largely a matter of the designer’s intuition, and while failed designs naturally are not eagerly reported, they do occur [3]. An example that became infamous in the German-speaking community is a call to design the label of a dish liquid bottle to which an out-of-control crowd replied with joking suggestions such as advertising the liquid to be “chicken flavored”. Dealing with the event caused
massive image losses for the company that had issued the call [31].

Besides its practical relevance, the inverse problem also reveals a gap in the body of knowledge. While there was some progress in the research of the theory of CI in recent years, very little progress has been made regarding how theory may inform the design of CIS eventually contributing to a general solution of the inverse problem. This precisely is the research gap that a design science approach to CIS can fill.

When speaking of a general solution to the inverse problem, an algorithm comes into mind that takes a formalized challenge as input and outputs a CIS. Obviously such an algorithm is out of reach because this algorithm would have to perform software engineering which is not algorithmically solved currently and even worse, it would have to perform STS engineering which is even less possible to formalize completely. But because this is in part a software engineering problem, in search of a more promising sub-problem of the inverse problem it is feasible to take inspiration from lessons learned with computer aided software engineering (CASE).

We argue for the potential of similar tools to support human designers in solving specific instances of the inverse problem and that the creation of such a tool is relevant in the same right as the development of a general solution. This consideration also justifies the choice of design science methodology in the first place.

We need to discuss, what makes CIS design “special” in contrast to related problems that are more or less solved with existing tools. Hevner et al. write: “Routine design is the application of existing knowledge to organizational problems […]. The key differentiator between routine design and design research is the clear identification of a contribution to the archival knowledge base of foundations and methodologies.” [21]. We now show that CIS design is more than routine design.

Figure 3 shows how routine design works. According to the business needs of a user, the designer designs what eventually becomes the technical subsystem. Ideally the designer will keep in mind the enclosing STS when designing the user interface of the system but mostly the STS is outside scope of the design. Because the business needs are the users own, they alone suffice to motivate the user to interact with the system and in reverse the interaction has the clear goal to fulfill the business needs. In contrast to the findings about CIS in previous work [19], user motivation is not part of the design task. If this motivational structure for some reason does not work a management solution rather than a design solution is called for. Essentially this is a case of business engineering vs. software engineering [32].

Figure 4: Swarm engineering overview.

Compare routine design to the case of swarm engineering [15] within the swarm ontology [33] as introduced in the related work section and depicted in Figure 4. There are some differences to be noted. In slight stretching of terminology we call the system a multi-agent system (MAS). This seems to be fit because only swarms of artificial agents can be engineered in this manner. Agents in such artificial swarms do nothing more than follow rules. These rules lead to rule-based swarm behavior. The design task is it to formulate rules for which the swarm behavior creates a desired outcome with respect to the initial challenge. Most notably, however, the challenge is detached from the swarm. The swarm is ignorant of the challenge and follows the rules blindly, without understanding their purpose.

Figure 5: CIS design overview.

CIS design, as depicted in Figure 5 suffers from the combined complexity of routine design and swarm
engineering. This is to say that the collective is detached from the underlying business challenge and not intrinsically motivated to overcome it, but it is also reluctant to blindly follow rules. Following [19], instead we assume that the individuals have their own motives that will guide their behavior and that the behavior can only be influenced indirectly by adding incentives.

At the current state of the art, the creation of a tool supporting CIS design is still a major problem which will require further decomposition and much more research than can be conducted within this paper. Nevertheless, the reasoning presented so far makes it easy to defend the relevance of all the related smaller research problems and in particular is sufficient to rationalize the research question which we have put forward at the beginning of this paper.

### 3.2 Objectives of a Solution

Peffers et al. subtitle the second step of the DSRP with the question “what would a better artifact accomplish?” [22]. The wording might be misleading here because currently there exists no similar artifact. Comparing idealized CIS design (Figure 5) with the inverse problem of CI (Figure 2) we have to acknowledge, that not all components of the CIS can equally be influenced by the designer, when transforming a challenge into a CIS. This goes so far that for open CIS even the precise composition of the collective is unknown and out of the designers reach. Nevertheless we now discuss why a supporting software system might be of greater use to a CIS designer in comparison with existing artifacts of much more theoretical character, e.g. models of CI or CIS.

Following work on CASE tools, we argue that a tool can provide systematic means of creating a design model of the future CIS and testing it in simulation. Furthermore on the same grounds we argue that solving the inverse problem requires creativity and intuition to a degree which cannot be substituted by a tool but can also be supported given an interface that allows for the tool to be used in creative processes [34].

Basically we understand the design of the technical subsystem as a means to enable desired ways of interaction and simulation as a way to predict how the collective employs these possibilities in its actual behavior which may sometimes evolve in undesired ways, as the example in the previous section has shown. To achieve this, the capability of case tools to support the design of the technical subsystem can be combined with the capability of MAS tools to simulate the behavior of agent groups.

There exists work on the combination of such tools [35], but a CIS design tool needs to go beyond that: Neither kind of tool currently supports the design of incentives with respect to given motivators and a given task which allow to guide collective behavior into desirable interactions. The design of incentives not independent of the technical design so that the two need to be closely integrated. It is not sufficient to simply add a MAS component to an existing CASE tool. The functionality of existing MAS tools might even turn out to be a “red herring” in the CIS context because they assume a degree of control over the agent’s behavior that a CIS designer does not really have.

Continuing to adhere to the DSG we need to show how our tool is going to be applied in practice. To this end we sketch out a simple iterative CIS design workflow in Figure 6. The workflow contains an analytical step that needs to be performed by the designer and can only be informed by the tool, two productive steps that can be supported by the tool and one simulative step that is performed by the tool alone. These steps can be repeated until the CIS displays satisfactory behavior in simulation. The structure of the workflow again speaks to the need of integrating technical design as represented by a design model and incentive design. Testing only one of them in simulation without taking into account the other would render the whole approach useless. The workflow also helps to identify the need to formalize challenge specific assumptions about the components not under the influence of the designer to that they can be taken into account by the simulation.
As for our more short termed research question, the objective of a conceptual model in the context of the larger objectives is to provide information about the domain within which assumptions, incentives and design model reside. In order to be used in simulation, all three of the aforementioned must be somehow represented within the CIS design tool and a conceptual model is a first step towards formalizing a representation. In addition it may also help the designer to develop her own mental model of the domain.

3.3 A Conceptual Model of CIS

In the interest of research rigor we ground our conceptual model on a theoretical basis consisting of the following:

- A synthesis of the ontological results reported in the related work section of this paper.
- Work on utility functions in CI, stating that the individual seek to maximize their own utility independent of progress towards the actual challenge which can be measured as a distinct global utility [30].
- Work on the simulation of CI, stating that CI evolves in repeated cycles of action and observation [11].
- SeeMe as a theoretically sound modeling language [28].

The resulting model is depicted in Figure 7. We use a very simple model of interaction in which all communication is mediated by the technical subsystem. The collective consists of several individuals that have expectations towards each other and will use these expectations in decision making. Generic interaction involves issuing actions to the technical subsystem and observing the reaction of the system. The crucial point is that unlike the interaction with traditional information systems the individuals do not interact with the goal to create information per se. Instead they act because they find the action itself rewarding (learning, socializing, etc.) or expect a pleasant side-effect unrelated to the creation of information (direct compensation, self-marketing, etc.).

Assumptions and expectations about this can be formalized in utility functions for the individuals, which the individuals will maximize. The design of incentives can be seen as a way to influence the individual utility functions.

What the technical subsystem essentially does is to take the action and generate two things. Data based on the action and an observation to be fed back to the collective. Since the collective is not interested in the data in the first place, it is also crucial here that unlike in traditional information systems, the feedback does not need to be a close representation of the data. Instead the feedback mechanism should be designed as to guide the collective behavior, e.g. by informing about incentives. The generation of information from the data happens behind the scenes and maximizes the global utility function as a formal representation of the business challenge.

Naturally, all interaction passes through the user interface. While the generic mechanism described so far, stays the same for all CIS, the real task is to define a user interface that supports a set of interactions that are helpful to generate the required information while at the same time generating feedback that satisfies the collective without the two having to pursue the same goals. This task can only be accomplished through analysis of the specific problem at hand. Even in absence of a CIS design tool we suggest to use the conceptual model to guide this analysis. At the conceptual level, a CIS can be described by a very small set of functions that formalize the mappings of action to observation, action to data, observation to individual utility, data to information and information to global utility. The simulation of CIS in principle is fully parameterized by these functions. Designing and implementing such simulation is impossible without making some assumptions on the nature of these functions. According to the DSG further research is required to justify these assumptions, but, even more importantly, following the design science approach to
CIS simulation will contribute theory on how the different parts of the system that are represented by these functions interact and together make up the phenomenon of CI.

**4. Conclusion and Outlook**

In the course of this paper we have identified a research gap to the extent that theoretical findings on CI have only found very limited application to the design of new CIS. We have proposed a design science approach to mitigate this problem and have made first steps following this approach. Doing so we have shown that CIS design, though in parts similar to routine design and swarm engineering, is relevantly and considerably different from these related problems and that CIS follow their own motivational mechanism. This finding has proved to be a leitmotif in the discussion of tool support for CIS design where we have shown that existing tools...
do not provide support for designing these motivational mechanisms and that a better tool should focus on this topic. Furthermore we argue that such tool needs to be capable of simulating CIS to be of good use in practice. This leads us to the research question of how a CIS can be conceptualized for later simulation and which concepts need to be encompassed. As a result we present a conceptual CIS model that is grounded in theory and contributes to the body of knowledge a first understanding of how to parameterize such a simulation with appropriate functions. Further research will need to show the precise nature of these functions and their interaction. If this research is continued to be carried out according to design science principles, eventually a CIS design tool will not only aid CIS design in practice but also be an artifact that informs theory about the inner workings of CIS.

5. References


