Elements of Software Ecosystem Early-Stage Design for Collective Intelligence Systems

Juergen Musil
jmusil@computer.org

Angelika Musil
angelika@computer.org

Stefan Biffl
stefan.biffl@tuwien.ac.at

Christian Doppler Laboratory for Software Engineering Integration for Flexible Automation Systems
Institute of Software Technology and Interactive Systems
Vienna University of Technology, Vienna, Austria

ABSTRACT

User-contribution driven software service platforms like crowdsourcing and social media services represent an efficient way of aggregating and distributing knowledge. However, only little research has been reported on early-stage design of software ecosystems (SECOs) for software service platforms, in particular in the collective intelligence (CI) domain. In this work we analyze needs for CI-centered SECOs leading to new research challenges. We have identified self-organization and feedback mechanisms as essential characteristics in CI-centered SECOs and thus introduce design elements for structuring them properly at an early stage of design. We discuss the concept with a real-world use case from a widely used CI-centered SECO, Wikipedia. A major result is the successful mapping of the design elements to the specific SECO elements.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; H.3.5 [**Information Storage and Retrieval**]: On-line Information Services—*Web-based services*; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*Collaborative computing, Web-based interaction*; H.1.2 [**Models and Principles**]: User/Machine Systems—*Human factors, Human information processing*

General Terms

Design, Human Factors, Theory

Keywords

Collective intelligence, human computation, software architecture, software ecosystems, stigmergic information system

1. INTRODUCTION

Software service platforms relying on user contributions like crowdsourcing platforms, social networking services or virtual worlds have proven to efficiently aggregate and distribute knowledge [18, 15]. Examples are posting and sharing activites on Facebook[1] [2], looking up restaurant peer reviews on Yelp[1], or editing a Wikipedia[1] article. Successful platforms create sustainable virtual ecosystems, each supported by a distinct user community, which generates a continuously growing repository of valuable knowledge and data [15]. Today around 50% of the top Alexa[1]-ranked websites are user-contribution driven and each of them has generated a prosperous software ecosystem. Due to their openness and stickiness [4], SECOs are also a good strategy to obtain a hard-to-replicate, competitive advantage [14, 12]. Although some research has been reported trying to understand established SECOs, only little effort has been undertaken so far to understand the early-stage design of SECOs, in particular the SECOs based on a software service platform. Deepening the understanding of early-stage SECO design is thus important, since (1) it is cheaper to change software design in early phases, and (2) many companies which launch service platform SECOs are start-ups and have only one chance to design and build the system right. This work focuses on the early-stage design of software service platform ecosystems in the context of collective intelligence. The paper presents self-organization/ feedback mechanisms and design elements, which can be used by system architects to guide conceptual design for collective intelligence-centered SECOs.

The remainder of this paper will discuss related areas of CI-centered SECOs and their relationship. Based on the identified key characteristics we will present basic design elements used for early-stage SECO design and discuss them with a real-world example use case. Finally, the paper concludes and illustrates future research.

2. RELATED WORK

This section presents an overview of related work on collective intelligence, software ecosystems and self-organization.

2.1 Collective Intelligence

Collective intelligence (CI) is broadly defined by Malone et al. [15] as "groups of individuals doing things collectively that seem intelligent". Depending on the intensity of computational or human tasks, CI systems can be categorized into three groups [17]: crowdsourcing, human computation, and social computing.

[1] www.{name}.com
[2] All URLs referenced in this publication have been last visited at 06/30/2013.

WEA '13, August 19, 2013, Saint Petersburg, Russian Federation

ACM 978-1-4503-2314-7/13/08 ...$15.00.
http://dx.doi.org/10.1145/2501585.2501590

1. Crowdsourcing describes the outsourcing of a task to an undefined group of individuals instead of assigning it to a (single) employee [11], like the cataloging of landforms (NASA Clickworkers[3]) or prediction markets [15]. Successful crowdsourcing platforms are Wikipedia[1], InnoCentive[1] and Amazon Mechanical Turk[4].
2. Human computation aims to replace computing power with human work to "perform tasks that computers cannot yet perform" [1]. According to Quinn et al. [17] this means that the problem is per se suitable for computation, but it is yet not solvable by computers and that the ICT system and process guide overall user participation. An example for human computation would be Captchas [1].
3. Social computing describes the facilitation of human interaction among individuals through an ICT system acting as a mediation hub [17]. Examples would be social networking services (Facebook[1]) and community platforms (YouTube[1], FourSquare[1]).

All three system groups share the commonality of (1) an ICT system as the locus of user coordination and (2) processing tasks that are split between the ICT system and the users. Such a hybrid configuration of human groups and machines addresses new solution spaces which couldn't have been reached by ICT systems (e.g. AI-only-based systems) or human groups alone.

2.2 Software Ecosystems

Software ecosystems (SECOs) are a young field in software engineering, which focuses on collaboration networks of communities and their interrelated business models and motivations surrounding a central piece of software, software service platform, or IT standard. So far multiple definitions of the term software ecosystem exist, depending on the scholastic viewpoint. Selected examples of definitions can be found in Kittlaus et al. [14, p. 25] Bosch et al. [5, p. 68], Jansen et al. [12, p. 2] and Hanssen et al. [9, p. 3]. Though there are multiple views on SECOs, Hanssen et al. [9] have identified five reoccuring aspects in SECO literature, which are illustrated in figure 1:

a. Central reference organization: Each SECO is governed by a coordination organization which is instrumental in overseeing and guiding the evolution of the ecosystem. Jansen et al. [12] note that these organizations are the main "beneficiaries of software ecosystem growth who have instruments available to influence the development of the platform or the surrounding ecosystem". Also such an organization acts as a hub to which the ecosystem members are linked and which is able to exercise some form of control (extensive or partial) upon the members [9]. Examples of coordination organizations are privately owned companies (Apple, Facebook) or consortia (Apache Foundation, OSGi Alliance).

b. Networked character: It has been observed that software ecosystems trend to generate network structures between its actors and organizations. In fact a central expectation of pursuing an ecosystem strategy is to thrive on network effects [13] which arise from these interdependencies [14, 9, 12]. Bosch [5] addresses this aspect from the perspective of openness by arguing that a software ecosystem approach requires a "community-centric way of collaborating and coordinating" relying on external contributors. An approach which stands in strong contrast to intra-organizational

[3]http://nasaclickworkers.com

[4]http://mturk.com

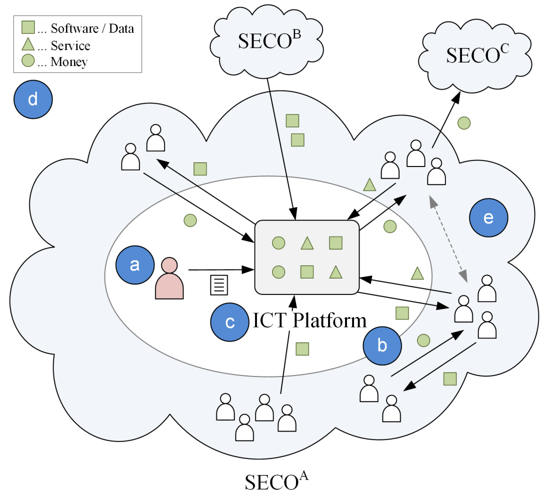

Figure 1: Overview of a generic software ecosystem.

paradigms and solutions like software product lines (Bosch 2010). Figure 1 (b) illustrates the networked character by depicting a network of actor groups exchanging software, data, services and money in various configurations.

c. ICT platform: The ICT platform is the central driver around which the software ecosystem evolves. Gawer et al. [8] define the platform aspect of a SECO as "A foundation technology or set of components used beyond a single firm and that brings multiple parties together for a common purpose or to solve a common problem". ICT responsibilities described by Hansen et al. [9] may include communication, coordination, development, solution deployment and economic transactions. Figure 1 (c) depicts the ICT platform as an abstract element, since it depends on each SECO of what the platform may look like. Examples of a platform are an app store, a social networking service, a technical standard or a programming framework.

d. Shared values: Shared values can be of extrinsic (software product, platform, complementary services, business domain) or intrinsic (motivation of individual actors and organizations) nature [9, 12].

e. Self-regulation: Self-regulation is achieved through feedback and regular interations between actors [9].

It can be observed that matured instances of CI systems like Facebook are consistently related to software service platform SECOs [4, 12]. Bosch notes on the inception of "application-centric, domain-specific SaaS" SECOs, that they do not happen to be SECOs per se, but develop in a 2-phase process [4, p. 114]. At the beginning there is a successful online application, which generates a considerable user/customer base. In the second phase, the application is opened to 3rd party developers and content providers. Though, for CI systems the case is different, since as user-contribution driven software service platforms, they have to be open to external content (via user community) already in the first phase. Openness to external contributors is a key aspect, which CI systems and SECOs share. Instances from both groups are also often mentioned as examples within the OpenInnovation movement [7].

2.3 Self-Organization

This section discusses self-regulatory mechanisms, which have been identified as an important SECO aspect, from the perspective of self-organization in detail.

Self-organization describes a process where a system increases its internal organization by itself [3]. Two main factors for self-organization are emergence and feedback. Emergence describes a phenomenon, where a system property exists on a macroscopic level, but not on a microsocopic level or vice versa [2]. Feedback describes a circular process where the output of a system is returned to its input in order to regulate its further output, also refered as feedback loop [6]. It can be distinguished between positive feedback (higher output increases further input) and negative feedback (higher output reduces further input). Feedback loops have been identified by Brun et al. [6] as an important generic mechanism for self-adaptation in software-intensive systems and argued as first-class entities in the design and development of self-adaptive systems [6]. An interesting control process is stigmergy (from Greek *stigma*: sign, and *ergon*: work), which is a bio-inspired indirect communication mechanism, where agents communicate by modifying their environment [3]. The environment is used as a "shared medium for storing information so that it can be interpreted by other individuals" [10]. Using the example of termites, Bonabeau et al. [3] describe the feedback loop in stigmergy as "a stimulating configuration triggers the response of a termite worker, transforming the configuration into another configuration that may trigger in turn another (possibly different) action performed by the same termite or any other worker in the colony". As stigmergy is subjected to be involved in content-oriented collaborative environments [10, 18] (e.g. Wikipedia), from which most fall also into the previously indentified categories of CI systems, it can be hypothesized, that stigmergy could be a relevant factor in the general process design of CI systems as well.

The previous paragraphs have introduced to CI systems and SECOs and discussed the relationship between both. Since feedback loops and control processes are key elements of any self-organizational system, they also require particular attention in the design of software ecosystems.

3. DESIGNING SECOS FOR CI

The design of SECOs is due to their inherent evolving nature a constant process. Therefore, this work limits its scope focusing on the early-stage conceptual design of SECOs and addressing the following research issues:

- R1 - Identification of a self-organization and feedback mechanism concept for CI-centered SECOs.
- R2 - Identification of design elements, which are necessary for the early-stage design phase of CI-centered SECOs.
- R3 - Mapping the concept with the real-world use case of the CI-centered SECO Wikipedia.

The following section presents the extension of a generic SECO to a generic CI-centered SECO. The extension is achieved by the application of the Stigmergic Information System (SIS) model [16] on the generic SECO. The SIS model is a novel architectural model for software platforms,

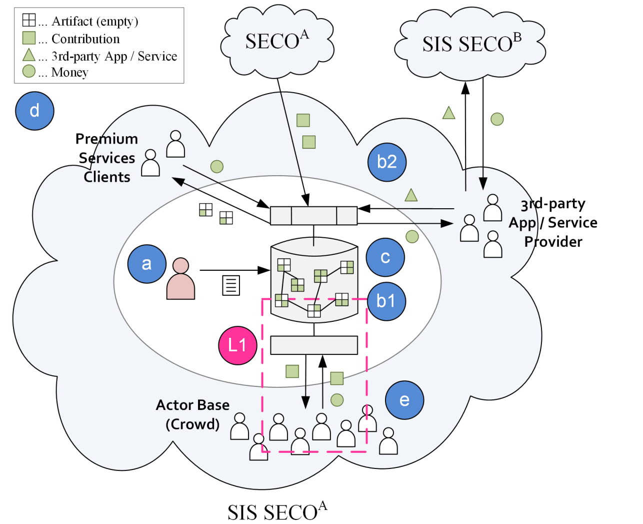

Figure 2: CI-centered SECO, based on the SIS model.

which enables self-organization through environment-mediated indirect communication for human agents. A distinctive feature of the SIS model is that the model supports coordination artifacts and feeback loops as first-class citizens during system design. The SIS model aims to "facilitate the building of an information network by allowing actors to create / modify network elements and thereby share information among each other" [16]. Whereby its main capability is to bottom-up "harness collective intelligence by stimulating, aggregating, leveraging, and distributing user contributions" [16].

The following paragraphs relate the areas of the SIS model [16] actor base, information network, control services and software ecosystem to the elements of the extended CI-centered SECO in figure 2:

Actor base: The collectivity of human agents (e), which engage with the artifact network in a consuming or producing activity (b1).

Information network of artifacts: The artifact network (c) preserves contributions made by the actor base and logs the usage behavior of each single actor. The artifact network follows the structure of a scale-free network [19], which creation follows the two steps of (1) growth: network growth is achieved by introducing new vertices, and (2) preferential attachment [19]: a new vertex is more likely linked to a vertex with many links. The shared values in CI-centered SECO (d) are firstly the coordination artifacts (e.g. Wikipedia articles) and the actor contributions, and secondly complementary services and apps, which build upon the aggregated information within the artifacts.

Control services: The control services maintains the activity level of the feedback loop between actors and artifacts (L1), monitors network behavior and regulates artifact network access (b2) for 3rd-party content providers, premium service clients and other SECOs.

Software ecosystem: The SECO is built upon the artifact network as a resource pool and the ongoing stream of actor contributions in the artifacts. 3rd-party dependent symbiotic services [16] may be able to access the artifact network

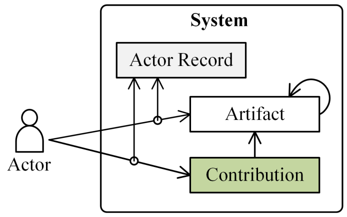

Figure 3: Design elements for CI-centered SECOs.

in order to provide additional services to actors. Normally the reference organization (a) operates the ICT platform (b1, b2, c), which equals to application-centric SECOs [4].

As long as the stigmergy cycle (L1) generates sufficient actor/artifact activity, this center of growth can be exploited, especially in early stages, in a 'Bilbao-Effect'-like manner[5] by building up a software service platform ecosystem, which seeks to either capitalize on the generated contributions (e.g. special third party access, reports) or the stigmergy cycle itself (e.g. transaction fees). System level and software ecosystem level are often interdependent, so that changes on one level affect the other, which makes cause-and-effect prediction of the overall system behavior particularly challenging.

4. BASIC DESIGN ELEMENTS

This section discusses basic design elements for CI-centered SECOs. Figure 3 illustrates the relationships between the five main elements: actor, artifact, contribution, actor record and system.

Actor. An actor is a human agent, who has privileges to manipulate the artifact through contributions (e.g. via a user interface).

Artifact. An artifact is a container with a defined data structure, which is the same for all artifacts. A certain part of the artifact can be filled with content, which can be manipulated through contributions by one or more actors. The artifact is persistent, so that an artifact's content is shared and accessible by the actors. An actor can create (= instantiate a new artifact with no content) and delete an artifact and change its content, but not its data structure. Actors can link artifacts together, thus causing the growth information network of artifacts (artifact network).

Contribution. A contribution is any modification of an artifact's content through an actor either by adding or removing content from the artifact. Typically all actors have the same means of contributing avaiable. Privileged contribution forms may exist (e.g. for admins, editors).

Actor Record. The actor record (AR) is a layer through which the actor accesses the artifact network and its nodes, the artifacts. The AR stores data, which is necessary for the actor to access the artifact network (e.g. credentials) and logs in addition any activity of the actor within the artifact network. Each actor has only one AR. Although the AR stores a considerable amount of an actor's generated data,

[5]http://www.forbes.com/2002/02/20/0220conn.html

Table 1: Design elements mapping in Wikipedia.

Element	Wikipedia Scenario
Actor	A user who is logged in with her Wikipedia account.
Artifact	The artifact is realized in the article concept, which consists primarily of unstructured text that can be enriched with image, audio and video sources. Each article has a 'talk page', where actors are able to exchange their views about the article.
Contribution	Any article manipulation, as well as any discussion on the article's 'talk page'.
Actor Record	A Wikipedia user account.
System	The Wikipedia system is implemented by a scaled-up MediaWiki platform. The ICT platform pushes article changes and discussion posts to other actors and gives them guidance about the activity level of articles and other actors. Also, the system bridges between Wikipedia instances of different languages in order to stimulate the translation/adaption of an article's topic in other languages.

it is not a profile, but it can be used by the system as a data basis for generating an actor's profile.

System. The system layer is the governing environment, which coordinates actors and artifacts, by maintaining a perpetual feedback loop of actor contributions (*stigmergy cycle*). Since actors and artifacts are isolated in virtual space, the system has to act as the medium, which raises new artifact contributions to other actors' attention in order to *stimulate* further contributions. This is a remarkable distinction to stigmergy in the physical world, where the transportation medium (e.g. air) between agents and artifacts 'just exists'.

5. EVALUATION WITH WIKIPEDIA

This section explains the introduced CI-centered SECO and design elements with Wikipedia as example use case. Wikipedia disrupted the encyclopedia business by introducing the wiki concept and providing a free online encyclopedia platform, where articles are created, written and edited by volunteers. This changed the way of knowledge aggregation as since up to that time encyclopedias were exclusively edited and sold by private publishing companies and their articles were written and reviewed by a selected group of subject experts and publisher employees. Table 1 maps Wikipedia elements to the design elements identified in the previous section and the following paragraphs highlights its SECO characteristics based on figure 2.

1. Wikipedia's stigmergy cycle (L1) is located between the actor base (e) and the artifact network (c) enabling the aggregation of contributions from volunteer actors.

2. Openness to 3rd-party developers who provide apps and services on top of the aggregated data. Wikipedia increases its stickiness by providing database dumps of its articles free of charge, so many applications and services are able to rely on Wikipedia's data to increase their product value, thus improving indirectly the outreach of Wikipedia's actor contributions.

3. The MediaWiki platform provides users with a free solution for running their own wiki system. MediaWiki has become one of the leading platforms in self-hosted wiki systems and has generated an extension[6] market on its own.

4. Wikipedia has acted as lead platform around which the Wikimedia software ecosystem has evolved, which includes the platforms: Wiktionary, Wikiquote, Wikibooks, Wikisource, Wikispecies, Wikinews, Wikivoyage, Wikimedia Commons and Wikidata. Therefore a key function of Wikipedia is also the cross-pollination of adjacent Wikimedia platforms as well as enhancing its own artifacts with contributions from these systems.

6. CONCLUSIONS & FUTURE WORK

In this paper we discussed software ecosystems in the CI domain and highlighted mechanisms for self-organization and process control as important elements of a CI-centered SECO's self-regulation capabilities. The example of Wikipedia has illustrated that blending together a SECO approach with a feedback loop-centered perspective like the SIS model has the potential to provide a wider and more detailed viewpoint of the system.

Future research will focus on the design and analysis of software service platforms for CI. Next steps include (1) conducting a comprehensive survey of existing CI systems and their characteristic features as well as (2) interviews with domain experts to discuss and refine the CI-centered SECO and SIS model. (3) Also it is planned to extend the design elements with CI-specific quality attributes and a measurement framework to support a quality-assured architecture design process. The investigation of domain-specific SECOs, like CI-centered SECOs, is promising since it deepens the understanding of the differences between generic and specific SECO features, thus providing better strategic guidance for architects of the next generation of software ecosystems.

7. ACKNOWLEDGMENTS

This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria.

8. REFERENCES

[1] L. V. Ahn. *Human Computation*. PhD thesis, Carnegie Mellon University, Pittsburgh, Dec. 2005.

[2] M. A. Bedau and P. Humphreys. *Emergence: Contemporary Readings in Philosophy and Science*. MIT Press, 2007.

[3] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[4] J. Bosch. From Software Product Lines to Software Ecosystems. In *Proc. of the 13th Int'l Software Product Line Conference (SPLC '09)*, pages 111–119. Carnegie Mellon University, 2009.

[5] J. Bosch and P. Bosch-Sijtsema. From integration to composition: On the impact of software product lines, global development and ecosystems. *Journal of Systems and Software*, 83(1):67–76, Jan. 2010.

[6] Y. Brun, G. Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw. Engineering Self-Adaptive Systems through Feedback Loops. In *Software Engineering for Self-Adaptive Systems*, pages 48–70. Springer, 2009.

[7] H. Chesbrough. *Open Service Innovation*. Jossey-Bass, 2011.

[8] A. Gawer and M. Cusumano. *Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation*. Harvard Business School Press, 2002.

[9] G. K. Hanssen and T. Dyba. Theoretical foundations of software ecosystems. In *Proc. of the 4th Int'l Workshop on Software Ecosystems (IWSECO '12)*, pages 6–17. CEUR-WS, 2012.

[10] F. Heylighen. Collective Intelligence and its Implementation on the Web : Algorithms to Develop a Collective Mental Map. *Computational & Mathematical Organization Theory*, 5(3):253–280, Oct. 1999.

[11] J. Howe. Crowdsourcing: A Definition, 2006. Available: http://www.crowdsourcing.com/cs/2006/06/crowdsourcing_a.html (last visited 05/08/2013).

[12] S. Jansen and M. Cusumano. Defining Software Ecosystems: A Survey of Software Platforms and Business Network Governance. In *Proc. of the 4th Int'l Workshop on Software Ecosystems (IWSECO '12)*, pages 41–58. CEUR-WS, 2012.

[13] M. L. Katz and C. Shapiro. Network externalities, competition, and compatibility. *The American economic review*, 75(3):424–440, 1985.

[14] H.-B. Kittlaus and P. N. Clough. *Software Product Management and Pricing: Key Success Factors for Software Organizations*. Springer, 2009.

[15] T. W. Malone, R. Laubacher, and C. Dellarocas. Harnessing Crowds : Mapping the Genome of Collective Intelligence. Working paper no. 2009-001, MIT Center for Collective Intelligence, Feb. 2009.

[16] J. Musil, A. Musil, D. Winkler, and S. Biffl. A First Account on Stigmergic Information Systems and Their Impact on Platform Development. In *Proc. of the WICSA/ECSA 2012 Companion Volume (WICSA/ECSA '12)*, pages 69–73. ACM, 2012.

[17] A. J. Quinn and B. B. Bederson. Human Computation: A Survey and Taxonomy of a Growing Field. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*, pages 1403–1412. ACM, 2011.

[18] H. Van Dyke Parunak. A Survey of Environments and Mechanisms for Human-Human Stigmergy. In *Proc. of the 2nd Int'l Conference on Environments for Multi-Agent Systems (E4MAS '05)*, number 2005, pages 163–186. Springer-Verlag, 2006.

[19] M. van Steen. *Graph Theory and Complex Networks: An Introduction*. Maarten von Steen, 2010.

[6] http://www.mediawiki.org/wiki/Manual:Extensions