Agile Research in Information Systems Field: Analysis from Knowledge Transformation Perspective

Ilia Bider

DSV, Stockholm University/IbisSoft, Stockholm, Sweden
Ilia@{ibissoft|dsv.su}.se

ABSTRACT

Due to the relative success of agile methods in software development, the idea of having agile processes started to be tested in other areas, for example, agile business process development. This trend already reached the research community and there have appeared some materials that suggest using agility in research projects. Analysis of these suggestions, however, shows that they do not go beyond finding superficial analogy between the concepts of the software development and research projects. The paper presents a deeper analysis of the concept of agile research in Information Systems (IS) based on the analysis of the research projects from the knowledge transformation perspective. As a basis for analysis, the SECI model of Nonaka is used. Based on this analysis, several suggestions are made on how to conduct agile research in IS, e.g. prioritize relevance over vigor, test early for a practical purpose, use own experience and reflections, etc. It is also shown that some research types, like action research and design science, are more suitable for conducting agile research than others. The paper also gives analysis of risks of non-agile research, and presents an example where they are revealed.

KEYWORDS

Agile development, IS research, action research, design science, knowledge creation, tacit knowledge, SEKI, Nonaka

1. Introduction

As Agile Software Development (ASD) becomes fashionable in the field of software development (Agile Alliance, 2001) attempts are being made to transfer the ideas behind ASD to other fields. An example of such transfer is presented in (Bider & Jalali, 2014), where we applied the agile principles to business process development that in both practice and theory. The relative success of such transfer comes at no surprise as business processes nowadays are heavily supported by software systems, thus process development is tightly connected to support systems development. Recently, a number of attempts have been made to transfer the agile development principles to the areas that are not directly connected to system development. Some of this suggestions concern the academic world. For example, (Twidale & Nichols, 2013) re-drafts the agile manifesto (Agile Alliance, 2001) for agile university teaching. Other authors do such redrafting to define Agile Research (AR), see, for example, (Hazzan & Tozik, 2014; Amatriain, 2009).

The main risk when translating the agile manifesto (Agile Alliance, 2001) to another field without deep analysis is getting a superficial definition that does not lead to new results in the targeted field. The main problem here, in our view, is adequate translation of term “working software” used in the agile manifesto in
the principle regarding “Working software over comprehensive documentation”. In the field of software development, working software means two principles:

1. The software is functioning and is free from the major bugs
2. The software is actually used in the context for which it was intended

The two principles above are not mentioned in the agile manifesto, as they are considered to be common knowledge. However, both are important when understanding the ideas behind ASD. The main idea with ASD is aimed at the second principle of working software; the first one can be achieved even with the traditional, phase-based, development. Thus, when translating term “working software” to another field, one needs to take care of both principles above, as well as on how to verify that the “software” is working.

When translating the agile manifesto to the field of university teaching, (Twidale & Nichols, 2013), translates term “working software” into “demonstrable student achievements”, which implicitly incorporates analogous to both principles stated above, i.e.:

1. The student have passive knowledge in the subject
2. The student can use this knowledge for solving practically-oriented tasks

Attaining both can be tested through the exams, thus the translation suggested in (Twidale & Nichols, 2013) seems reasonable.

If we look at existing attempts of translation of term “working software” to the field of research, they show to be quite vague. For example, (Amatriain, 2009) translates “working software” into “real-world working solutions” that should be preferred to “comprehensive theories”. It is not clear what real-world working solutions are, and why they should be preferred to “comprehensive theories”. The latter definitely contradict Kurt Lewin’s statement “There is nothing more practical than a good theory” (Lewin, 1952). Another translation of the agile manifesto to the field of research (Hazzan & Tozik, 2014) implicitly translates “working software” into a document, such as research project report, a paper, or a PhD thesis. Even here, it is not clear what “working” means – acceptable for the reviewers and examination committee? This would be too thin to be accepted.

So far, we have not found materials on agile research that properly discuss analogues of term “working software” in the research world. In our view, without finding a proper analogue to “working software”, any translation of the agile manifesto to the realm of research is bound to be superficial and not leading to better quality or speed of the research projects.

The goal of this paper is to find a proper meaning of “working software” in the realm of research in the Information Systems (IS) discipline, and give a possible interpretation to agile research in IS. While pursuing this goal, we neither try to translate the agile manifesto (Agile Alliance, 2001) to the realm of research, nor try to tune the existing ASD project methodologies, like XP or SCRUM, to the needs of research projects. Instead, we perform a deeper analysis of research activity in order to find out what would be the essence of the agile project in the research world. To complete this task, we create two models of research projects, one that can be called Traditional Research (TR), and the other one which can be called Agile Research (AR), and show the difference between them.

The following two main sources are used for achieving the goal set above:

- Michael Polanyi’s ideas on the nature of knowledge (Polanyi, 1969), more exactly his statement that all knowledge is personal and tacit.
- SECI model of Nonaka (Nonaka, 1994) that explains how knowledge is created in organizations, where SECI stays for Socialization – Externalization – Combination – Internalization. More exactly, we use our approach of using SECI for analysis of agile software development (Bider, 2014) and agile business process development (Bider & Jalali, 2014).

The rest of the paper is structured as follows. In Section 2, we establish the meaning of “working software” for IS research. In Section 3, we give an overview of the works on which our investigation is based. In Section 4, we present models of traditional and agile IS research. In Section 5, we discuss the results and draw plans for the future.
2. Defining “working software” for research

As a starting point for defining an analogue for “working software” for research, we use the idea that the goal of any research is creating new knowledge. Therefore, the term software for the realm of research can be substituted with term knowledge. Furthermore, according to Michael Polanyi (Polanyi, 1969), the knowledge can exist in two forms: explicit and tacit. Explicit knowledge is represented using external media, paper or electronic media in the form of texts, formulas, drawings, etc. Tacit knowledge is in the head of people who can use it in practice consciously or unconsciously. According to Polanyi, in the end all knowledge needs to become personal and tacit, otherwise it cannot be used. At the same time, explicit knowledge plays a great role, especially in science, for both transferring knowledge between different people, and creating new knowledge through the known formal mechanisms, e.g. deduction.

Based on the deliberation above, we suggest:

- To consider scientific knowledge, e.g. a theory, in its explicit form to be analogous to “software” in the realm of software development.
- To consider this explicit knowledge to be “working” if there is at least one individual for whom this knowledge became tacit so that he/she can use it for some purpose having advantages of solving his/her tasks better with this knowledge than otherwise.

To finish the construction of an analogue of “working software”, we need to define for what purposes the scientific knowledge can be used when it becomes tacit. Here, we borrow the list from (Gregor, 2006) that states that theories have several purposes including to explain, to describe, to predict, to analyze and to design or take action. The list is open and can be extended or detailed if needed. For example, the purpose of analysis can be split into analysis of past experience for attaining organizational learning, and analysis of the current situation in order to plan actions.

Summarizing our discussion we can consider that “working software” in the realm of research could be translated into “working knowledge” defined as:

1. scientific knowledge, e.g. a theory, that exists in an explicit form, and in addition
2. there is at least one individual for whom this knowledge became tacit and he/she has successfully used it for some purpose

3. Background

As was already mentioned in the Introduction, we build our models of traditional (TR) and agile research (AR) based on the SECI model of Nonaka and its usage for analysis of agile development in (Bider, 2014; Bider & Jalali, 2014). As we cannot presume all readers being familiar with these works, in this section, we present a short overview of them.

The SECI model, where SECI stays for Socialization – Externalization – Combination – Internalization, by Nonaka (Nonaka, 1994) explains the ways of how knowledge is created in an organization while being transformed from the tacit form (in the heads of the people) to the explicit one (e.g. on the paper) and back, see Figure 1. (Nonaka, 1994) defines the cycle of knowledge creation as consisting of four steps or phases:

1. The cycle starts with Socialization (right top corner of the figure), where tacit knowledge is transferred from the heads of one group of people to others via informal means, conversations during the coffee breaks, meetings, observations, working together, etc.
2. The next phase is Externalization, which is the conversion of knowledge from the tacit form into the explicit one, e.g. a model of situation (right bottom corner of the figure).
3. The third phase is Combination, which is transforming the externalized (explicit) knowledge in a new form using existing knowledge, e.g. solution design principles (left bottom corner of the figure).
4. The last phase is Internalization, which converting the explicit knowledge, e.g. a solution, in the tacit knowledge of people that are ready to apply this knowledge to any situation that warrants it (left top corner of the figure).

The cycle of SECI can be repeated indefinitely reflecting constant creation of new knowledge.
Applying ideas from SECI to software development, (Bider, 2014) designed two models of knowledge transformation in software development projects, one - for Traditional Software Development (TSD), and another - for Agile Software Development (ASD). Both are presented in Fig. 2. Analysis of these models helped to understand the difference, advantages and drawbacks of each approach, and define the area of applicability for ASD. In the next section, we will use the same approach to model traditional and agile research and analyze their differences, advantages and drawbacks.

Using models in Fig. 2 as an example, we created similar models of knowledge transformation in traditional Research (TR) and Agile Research (AR), as presented in Fig. 3. For both models, we assume that the object of scientific investigation for IS is technology enabled human-activity systems, or work-system in terminology of (Alter, 2006). Furthermore, we consider that some knowledge about IS is already uncovered so it exist in an explicit form, as shown on the left-hand side of both model in Fig. 3. Some of this knowledge became personal and tacit for the community of researchers and practitioner, as shown on the left-top side of both models in Fig. 3. Some of the knowledge about IS remains hidden, we consider such knowledge to be embedded in existing IS and those that will come in the future. Such knowledge is embedded in:

- systems artifact, for example, policy documents, IT-systems in use, logs produced by such systems, and

4. Models of traditional and agile IS research


Figure 1. SECI diagram of knowledge creation (adapted from the original one to be more in line with the figures below)

Figure 2. Left – ECEA model (Externalization-Combination-Embedment-Adoption) for TSD; right - SEA model (Socialization-Embedemnt-Adoption) for ASD. Adapted from (Bider, 2014)
exist as tacit knowledge of human participants of the system
A goal of a research project, independently of whether it is a TR or AR project, is to convert to an explicit form some part of the hidden embedded knowledge. This explicit knowledge will later be adopted and used in practice thus becoming tacit knowledge of IS researchers and practitioners. However, AR and TR are trying to reach this goal in different ways, as discussed in the sub-sections that follows.

Figure 3. Knowledge transformation in research projects: left – ECCI model (Externalization-Combination-Combination-Internalization) for TR; right - AEI model (Appreciation-Externalization-Internalization) for AR

4.1 Knowledge transformation in traditional IS research
TR is based on the idea that the researcher is an external observer of the system of interest. Simplified TR model on the left-hand side of Fig. 3 has four main phases:

**Phase 1 – Data collection** - consists of gathering and categorizing information embedded in information system(s) of interest (right top corner of the model). This phase can consists of gathering and labeling documents, interviewing human participants of the system and setting tags on the interview questions and answers, preprocessing logs of IT-systems (if such logs exist). The procedure, for example, can follow recommendations from the ground theory, or other known methods used for data collection in IS. This phase corresponds to **Externalization** from SECI model.

**Phase 2 – Data processing** – consists of processing data gathered in the previous phase with the help of known methods, e.g. statistical methods (right bottom corner of the model). This phase corresponds to **Combination** of SECI model when new explicit knowledge is obtained by applying known methods to already explicaded knowledge.

**Phase 3 – Induction (theory building)** – consists of using processed data for building a theory with the help of logic and creative thinking. This phase also corresponds to **Combination** of SECI model.

**Phase 4 – Adoption (testing theory in practice)** – consists of using theory for some purpose thus converting explicit theoretical knowledge into personal tacit knowledge. This phase corresponds to **Internalization** of SECI model.

Note that the model of TR in Fig. 3 is an idealization and simplification. It presents TR as a sequence of phases which is not mandatory true, especially in longer complex research projects. All phases can run in
parallel or/and with loops. New data collection can be done after the preliminary processing shows that more data is needed. Also, in our TR model, we do not differentiate inductive and deductive research styles, attributing everything to induction. The difference will show itself in how the data collection is performed. Having a specific hypothesis that one wants to prove or disprove will affect the way data are collected. After collection the data still need to be processed. The difference is in the phase of theory building. In case of testing a hypothesis, the induction will be substantially reduced to the results is compatible with the hypothesis or not. In case of having no hypothesis, the theory building phase constitutes a more complex and creative process than in case of testing an existing hypothesis.

The advantages of using TR is that in case of success, the results are considered to be rigor in principle gives the theory more scientific weight. However, the TR process has the following built-in risks:

1. The cycle can be lengthy and does not guarantee results, as the success depends on selecting the right type of data for collection, and therefore on the experience of the research team. An experienced researcher has intuition on what data to collect, which might be lacking with less experience researchers. In addition, there might be no intermediate confirmation on that the team is on the right track.

2. The result, even if correct, might be formulated in a way that gives no clue to how it can be used, which will hinder adoption, especially by the practitioners. In other words, the theory produced, while being rigor, may lack relevance. For example, the theory can just state the fact of dependencies without providing a method of removing these dependencies. There is a risk that such theory will be forgotten, and reinvented later in another form, the one that promotes its usage for, at least, one purpose and by, at least, some members of the community.

3. The traditional cycle can be applied only to already existing information (work) systems. It is not aimed at suggesting completely new ways of organizing human activity with new technology.

To stress the importance of the second risk, consider the development in the field of LAP (Language Action Perspective). This field, quite popular in research circles for a time, could not find the area where it would be useful, and thus could not be tested in practice. The urgent needs to find an area of applicability for LAP were stated in (Lyytinen, 2004):

"(1) observe real challenges in practice where LAP related ideas can be effectively applied so that they can show significant economic benefits, (2) build a focus on a few and prominent areas in which LAP related solutions can be developed that demonstrate user value, (3) strive towards areas where ideas can be softwired into platforms that enable continued learning and codification of knowledge. A good example would be e-commerce platforms that are reflective and capable of reasoning around ongoing transactions, (4) build alliances with critical members of the knowledge transformation networks including platform providers, solution integrators and different communities of practice."

It looks like the plan above has never been executed, and the interest in LAP died, including the LAP working conference ceased to exist.

Three risks as above correspond to the risks of TSD from Fig. 2 left listed in (Bider, 2014): (1) instability – lack of the negative feedback loop (small errors in any phase could go undetected and result in a wrong system); (2) uncertainty – difficulty to imagine how the system will work in the context intended; (3) evolving context – the system becomes outdated before being released. These risks can be mitigated with the agile cycle like the one in Fig. 2 right, which is discussed in more details in the next sub-section.

### 4.2 Knowledge transformation in agile IS research

As follows from Fig. 2, the main ideas of ASD are: (1) shortcut through Requirements Engineering and Design directly to building software, and (2) multiple small cycles of software development and introduction of software in practice instead of one big cycle. Applying the same ideas to research in IS, we get a model of AR represented in Fig. 3 right. The model had a shortcut to theory building, and multiple cycles.

Having a shortcut to theory building in no ways excludes the need to have knowledge on which to build a theory. The difference is that in TR this is explicit knowledge obtained by processing collected data, while in AR, this is tacit knowledge of the researcher(s) engaged in building a theory. There are several ways of obtaining this tacit knowledge, all of them requiring involvement of the researcher(s) in the system(s) they
investigate, instead of being an external observer(s). This is exactly the idea presented in the dialog of a Zen master and a pupil from the epigraph to this paper. Involvement can, for example, be arranged through:

- Socialization with the human participant of the system(s) under investigation through informal talks meetings, message exchanges, etc. In this case, tacit knowledge is obtained in a way similar to ASD as represented in the model of Fig. 2 (right-hand side model).
- Researcher(s) being part of the system under investigation. This is often the case when the system under investigation belongs to the domain of university teaching.
- Researcher(s) being part of the project of developing a new IT system and introducing it in operational practice of the system(s) of interest.

As we can see from Fig. 3, Theory building in AR is of different nature than in TR. In the latter it is of inductive nature, while in the former it is of reflective nature. Another difference is the nature of the theory. TR set high demands on rigor, while AR sets high demands on relevance. A theory should be formulated in a way that makes it possible to apply it in the next iteration (this demand does not exist in TR). For example, it could be a method for improving collaboration in the work system under investigation. There is no demand on the method being rigor, or optimal, it can be of heuristic nature. The rigor can be obtained later through multiple iterations that show how well the method actually works.

Again, as in case of our model for TR, the AR model in Fig. 3 is an idealization and simplification. Besides obtaining tacit knowledge through the involvement, the researcher(s) may also compile some diary records, user stories, diagrams, etc. The purpose of collecting these documents is however different from data collecting in TR. This documentation serves as a supplementary material that helps the researcher(s) to grasp the holistic picture of the system(s) under investigation. Note also that the reflective nature of theory building in AR does not exclude the usage of already existing scientific knowledge independently whether it was obtained via TR or AR.

AR helps to mitigate all three risks of TR listed in section 4.1. Risks 1 and 2 are mitigated via AR’s iterative nature with smaller cycles. Risk 3 is mitigated through the researcher(s) initiating a project of introducing innovative and disruptive changes in the system(s) of interest, via, for example, developing and introducing a new type of IT-system. While helping to mitigate the risks of TR, AR sets certain requirements on the researchers engaged in AR: (1) being able to become involved, and (2) having ability to reflect on own experience. Both are difficult to obtain through the university classroom, and may be best attained through the apprenticeship.

5. Discussion and plans for the future

The goal of this paper was to try to find out the meaning of “agile research in IS”. This investigation was initiated due to appearance of a number of interpretations of agile research that we consider superficial. To complete the investigation, we used knowledge transformation perspective in the form that we have already used for analysis of agile projects in other fields, i.e., agile system and business process development.

Our investigation resulted in building models of traditional and agile research that show the ways knowledge is created and transformed in these two kinds of research projects (depicted in Fig. 3). These models helped to point out the risks built-in into TR, and how they could be mitigated with AR. The models also helped to understand the requirements on the researchers engaged in AR.

After defining the essence of AR through building the models, we can check which of already existing approaches to IS research can be qualified as agile. We see at least there examples:

1. Action Research, which is already in use in IS (Baskerville & Myers, 2004). For example, Kolb’s four stage learning cycle: Concrete experience -> Reflective Observation -> Abstract Conceptualization -> Active Experimentation -> Concrete Experience ->…, practically implements the AR model of Fig. 3 (Kolb, 1984).
2. Design Science (DS), which also started to become popular in IS (Hevner et al., 2004). DS is directed at developing new innovative solutions and testing them in practice, having dissemination of results an important part of the research project.
3. Reflective theory building (Mott, 1996), though this direction is not much used in IS. These examples show that, actually, AR is already in use in IS research, though it is far from constituting the mainstream. Papers related to Action Research and Design Science are rare and are not easily accepted by major IS-related journals.

The next question to ask is what potential usefulness of the models from Fig. 3 could be. We see several areas where they can be useful, including: (1) analysis of successful and unsuccessful past research projects (learning from success and failures), (2) decision making (whether to use TR or AR), (3) project planning for AR, (4) research education. Testing the models in, at least, one of these areas is in our plans for the future.

Finally, this paper is an example of agile research that exemplifies reflective theory building. The models in Fig. 3 and their analysis comes from reflections on the author’s (1) long practice in the fields of software and business process development, e.g., (Andersson et al., 2005), (2) experience in the research projects of the action research, e.g. (Bider et al., 2015) and design science, e.g. (Bider et al., 2012) types.

References