Pragmatic Software Innovation

Ivan Aaen, Rikke Hagensby Jensen

Pragmatic Software Innovation

Ivan Aaen, Rikke Hagensby Jensen

Department of Computer Science, Aalborg University, Denmark
ivan@cs.aau.dk, rjens@cs.aau.dk

Abstract. We understand software innovation as concerned with introducing innovation into the development of *software intensive systems*, i.e. systems in which software development and/or integration are dominant considerations. Innovation is key in almost any strategy for competitiveness in existing markets, for creating new markets, or for curbing rising public expenses, and software intensive systems are core elements in most such strategies. Software innovation therefore is vital for about every sector of the economy. Changes in software technologies over the last decades have opened up for experimentation, learning, and flexibility in ongoing software projects, but how can this change be used to facilitate software innovation? How can a team systematically identify and pursue opportunities to create added value in ongoing projects? In this paper, we describe Deweyan pragmatism as the philosophical foundation for *Essence* – a software innovation methodology – where unknown options and needs emerge as part of the development process itself. The foundation is illustrated via a simple example.

Keywords. Software innovation, software development, pragmatic philosophy, Deweyan pragmatism, Essence

1 Introduction

Innovation has been a recurring theme in public as well as academic debate for decades. Despite the widespread interest, there seems to be little advice on how to make innovation more likely to happen in software development – at least at the team or project level.

The classic commitment for software development is to ensure quality and efficiency [9]. Therefore, software engineering – whether traditional or agile – focus on delivering quality solutions in a predictable and effective way. But today we need more than just meeting requirements effectively. To stay competitive, we must create high value solutions.

Software innovation addresses a number of challenges including the development of innovative software products, designing software support for innovative business processes, transforming known solutions to innovative uses in new contexts, and stimulating paradigmatic changes among developers and customers concerning the framing of problem domain and the discovery of potential game changers on the market [45].

In a global world where ICT increasingly becomes commoditized to provide a shared and standardized infrastructure [11], there is a need to move from mainly operational considerations towards more strategic ones: Software development in high-cost countries is under pressure to deliver more valuable results than development overseas [4].

To remain competitive in this environment there is a need for software development to move beyond the traditional efficiency and quality focus: How can software teams deliver high value solutions?

Essence evolves as part of an ongoing effort to develop a software development methodology that facilitates a team in producing high value software solutions [1-3]. Essence sees software innovation as a reflective practice [38-40] where options and needs emerge as part of ongoing team-based efforts. At any given time, the team may work towards a goal and employ the means deemed relevant to attain it, but while working on a problem and on solutions to it, the team and other stakeholders discover needs and options that were unknown at the start.

The purpose of this paper is to outline a philosophical foundation for Essence – and team-based software innovation – based on Dewey's pragmatic philosophy [17-20]. The aim is to investigate core concepts of pragmatic philosophy, to find ways to help a software team develop, mature, and implement ideas.

The paper starts out by outlining software innovation as a concept and surveys contributions within the field (Section 2). Section 3 presents main ideas and concepts in Deweyan pragmatism. Section 4 is a simple illustration of reflective practices in incremental software innovation. Section 5 describes how this illustration was facilitated by key concepts in Essence and how these concepts reflect core ideas in Deweyan pragmatism. Section 6 concludes the paper.

2 Software innovation concepts and contributions

Software innovation is not an established term yet, and contributions to the field are scattered over organizational levels and project stages. Some contributions are generic and have little focus on either organizational level or particular stages [35]; some focus on the company level [33]; some on picking and improving promising ideas [22]; some on ideation in the requirements stage [28, 29]; and some on innovation as part of an ongoing project [1-3, 10, 12].

This paper is aimed at the methodology level. It is part of building a foundation to help software teams increase the value of what they build as they go about building it. There are numerous techniques and tools for creativity, and many insights on how to stimulate creative thinking and innovative work, but very little work has been done on methodology for software teams.

We focus on innovation as part of the software development project. We see software innovation as part of everyday life in a team, and thereby as part of what designers, users, and stakeholders in the project do. We focus less on what happens before a project is decided and more on what takes place from the decision to start a project until the end of it. We assume that modern development techniques are used to

allow for iteration and experimentation within reasonable levels of risk. In other words, we see software innovation as a reflective process where experiences and insights during a project may change its course.

We understand software innovation as concerned with introducing innovation into the development of *software intensive systems*; i.e. systems in which software development and/or integration are dominant considerations. Our focus is on innovations that offer something new to known users or customers, or something known to new users or customers. Specifically, software innovation here refers to the contribution of innovation to the user or customer side only. Therefore, we do not include changes in the software development process itself into our understanding of software innovation in this context.

Innovation usually extends creativity in the sense that ideas are developed and matured in the context of implementation. Basically ideation is concerned with the generation of socially acceptable ideas [42], whereas innovation by definition implies change in the real world [45].

While the interest in software innovation is fairly recent, there has been some interest in ideation and creativity since the early and mid 90ies. J. Daniel Couger worked on creative problem solving [14] and creativity techniques [15, 16] for information systems development, and Ben Shneiderman worked on creativity support in the same field [41]. Within software engineering, Neil Maiden has worked on creativity workshops [30] and stakeholder collaboration [29].

Contributions with a direct bearing on software innovation are still relatively few. Within information systems development, innovation research tends to focus on the business context and adoption of innovations. For example Burton Swanson [43] advocate mindfulness when innovating with IT. Within software development, there is some interest in innovation as a goal for software development. Jim Highsmith and Alistair Cockburn point to the potential for agile development to support innovation [25], but as Conboy et al. observe based on a number of studies on the relationship between agility and innovation or improvisation: *These have tended to focus more on the agile practices themselves as the innovation and not the extent to which the practices facilitate agility and innovation* [13].

In the last few years a growing number of writers have published very varied work on software innovation.

Jeremy Rose [35] proposes eight work-style heuristics for software developers, and Pikkarainen et al [33] offer eight fundamental practice areas for innovation with software, each containing a number of activities at the company level to master that particular practice.

Misra [32] presents a goal-driven measurement framework for software innovation processes linking these metrics to business goals. The processes per se are not part of this framework. Also at the business level, Gorschek [22] suggests Star Search, an innovation process using face-to-face screening and idea refinement for software-intensive product development. This process has particular focus on ideation and selection prior to actual development.

Focusing on the team level, Aaen discusses ways to facilitate software innovation [1-3]. At a similar level, Conboy and Morgan [12] discuss the applicability and

implications of open innovation in agile environments, and Mahaux and Maiden [28] suggest using improvisational theater as part of requirements elicitation.

These contributions generally focus on methods and normative principles, whereas work on philosophical foundations for software innovation still seems to be missing in the field.

Based on pragmatic philosophy, Donald A. Schön discussed reflective practices in design. In many ways, Schön's work can serve as inspiration for software innovation at the level of the individual or the team. To Schön, design is a reflective inquiry into a problem, what it is about, and possible ways to address it.

Schön completed his doctoral thesis at Harvard University in the 1950's, which dealt with John Dewey's theory of inquiry [39]. Schön's most notable work depicted in *The Reflective Practitioner* [38] and *Designing as reflective conversation with the materials of a design situation* [37] reworks Dewey's *theory of inquiry* into *reflective practice* [39].

Schön gives an insightful perspective on how to understand the design process, by illustrating the reciprocal interplay between the designer and the design situation. Schön describes this process in three steps: *Seeing-moving-seeing*. These steps are small iterations, where the designer with actions and the materials at hand allows the situation to talk back and thereby sees the design situation anew [40]. Schön compares this reflective conversation with a Deweyan inquiry, where human actors inquire into a problematic situation [37].

Lim et al. brings Schön's reflective conversations into the world of interaction design of information systems. In their article *The Anatomy of Prototypes* [27], the authors discuss the nature of prototyping and how the production of prototypes can induce reflection. They formulate a framework to help designers frame and refine design ideas by means of prototypes.

Biskjaer & Dalsgaard instigate a connection between design creativity and Deweyan pragmatism [8], while Goldkuhl discusses the philosophy of pragmatism in relation to information system design research in *Design Research in Search for a Paradigm: Pragmatism Is the Answer* [21] . By examining different pragmatic epistemic types of design research and relating these to four aspects of pragmatism, the author proposes pragmatism as an appropriate paradigm for information system design research.

Having presented some examples on how Dewey has inspired work on information systems design, we will present his original ideas in more detail.

3 Deweyan Pragmatism

Pragmatism originates from the United States. The philosophy emerged in the last decades of the 19th century as a response to experiences acquired by migrants settling the American frontier in search of the American Dream [36]. To cultivate and survive in this rogue and demanding environment, the migrants were constantly exposed to practical problems necessitating creative solutions. To choose among alternatives

these choices had to somehow be judged. This judgment was based on the practical consequences each choice induced.

One of the main contributors to the pragmatic philosophy tradition is John Dewey (1859-1952). His work covers a vast variety of topics including logic, ethics, education, politics and technology. In his lifetime, Dewey saw many innovations come to life: Power plants, automobiles, airplanes, the telephone, radio, AI and the first robot only to mention a few [24]. Dewey was an active part of this society and the innovation of technological artifacts was a catalyst to his work. His philosophic views are a reaction to the opportunities and problems that occur in a technological society [24].

One of Dewey's points of interests was how technological tools and instruments come to be, how they change the way we experience the world, and how they become part of shaping and building our own future [24]. One of Dewey's core ideas is that the problem solving process of producing of artifacts - physical (technology) as well as mental (theories/ideas) - are occurrences of the same creative process [36].

Dewey's critique of technology is a central topic in his work and becomes synonymous with his *theory of inquiry* [24]. The theory of inquiry is pivotal in Dewey's perspective on pragmatism, most noticeable depicted in *Logic: The Theory of Inquiry* [17]**.** This particular subject is central to this article, as we see software innovation as a recurring process of inquiry among team members and stakeholders.

We will discuss four core concepts of Deweyan pragmatism*: Problematic situation, inquiry, means, and ends (ends-in-view)*. These highly interconnected concepts are all presented in Dewey's *theory of inquiry*. To illustrate the social implications of inquiry in teams, we also present a fifth concept: *Community of inquiry*.

3.1 Problematic Situation

A situation can be seen as an environment wherein we live, experience, and most importantly act, reason, and reflect. The subjects, the environment, means, artifacts, and social constructs, and the relations among them, determine how we experience and understand a situation [17]. This means that our thoughts and actions can only be understood in the context of the unique situation we find ourselves in.

We may experience a situation as stable and determinate where we fully comprehend the implications of our actions. However, when encountering something uncertain and doubtful, an *indeterminate situation* supervenes. At this stage, the situation will appear *problematic,* and to actively engage in its resolution entails venturing into a process of inquiry [17].

Dewey describes the problematic situation as *open* [17]. The openness has two dimensions: The situation is open, as the elements that constitute the situation are in imbalance and dissociated. Therefore, the situation is also open for new actions and interpretations as we examine what caused it to become unsettled.

3.2 Inquiry

Inquiry means to make an investigation into matters of interest or problems in search of knowledge. Dewey sees inquiry as a process throughout life and in all aspects of living. To live means facing daily challenges. Meeting these challenges requires intelligent *inquiry*. In Deweyan pragmatism, this entails examining matters, inferring conclusions, and evaluating and reflecting upon these conclusions. The evaluation follows from the practical consequences the inquiry induces.

For Dewey, inquiry starts with the problematic situation. The objective is to intelligently transform the problematic situation into a stable and determinate one:

Inquiry is the controlled or directed transformation of an indeterminate situation into one that is so determinate in its constituent distinctions and relations as to convert the elements of the original situation into a unified whole [17].

To transform the problematic situation implies not only to understand what caused the situation to become doubtful in the first place, but also – through exploration and experimentation – to actively seek means *in* the situation for settling it [19]. Dewey describes the inquiry process as a sequence of iterative and intertwined states [17]:

A) The indeterminate situation: An inquiry is initiated by an indeterminate situation – an uncertain, unsettled, and disturbed situation. Doubt may arise from the confusing, obscuring, or conflicting in the situation.

B) Institution of a problem: By doubting, the problematic in the situation becomes the focal point of the investigation. Framing the problem entails identification and examination into the subject matters that causes the situation to become problematic. This knowledge will also direct the transformation of the situation to its resolution.

C) The determination of a problem-solution: In the process of framing the problem, observations are made concerning the facts affecting the situation. Based on these observations a possible solution is drawn, represented as one or more ideas. An idea forecasts the consequences to be expected from the employed activity. Thereby the idea instigates a course of direction. The more facts are observed, the more enhanced the perception of the idea and possible solution becomes.

D) Reasoning: The idea is developed and matured through experimentation and exploration. In the experiment, facts and ideas are tested against the problem at hand. The employed activity is maneuvered in the direction that makes the most common sense. Hence, inquiry combines both action and mental reasoning to make the situation determinate.

An inquiry should be understood as *transactional***.** Dewey sees human activity as a transaction between an actor and the environment [20]. Here actors are not bare observers from the outside but an active part of resolving the situation, as they act and reason *within* the situation [19].

However, inquiry does not only resolve doubt found in the situation. The outcome of the inquiry is likely to bring on new surprises and doubts, as the actor will create new environmental conditions, producing a new unique situation. This reciprocal relation will continue in a highly iterative manner. While transforming the problematic situation, the process will bring on new situations with new problems,

because every settlement introduces the conditions of some degree of a new unsettling [17].

3.3 Means

Confronted with a problematic situation we face choices among actions. To evaluate the consequences of our actions, we need valuation criteria to determine if our actions are successful in transforming the problematic situation into a stable and determinate one. Dewey sees criteria as something that arises from inquiry. Dewey forms this something in the concepts of means and ends and the inseparable relation between these [17].

Means are tools waiting to be employed in an activity. Tools can be external materials or bodily and mental organs. However, they only become means when they are employed. One way to understand the concept is to look at the distinction between materials, tools/techniques, and means according to Dewey [18].

Materials are objects an actor utilize to produce an artifact. Tools/techniques are instrumentals that have the potential of becoming means. When these are not employed in an activity they are just passive tools/techniques. They become means only when they are employed in conjunction with our bodily or mental organs. As means they become an active part of the actual situation and context, the actors find themselves in.

Where means can be seen as intermediates – a series of acts with an end – *ends* help to elaborate on what acts to perform and look at the next act in perspective of the context [18]. Ends give a clear direction of the course of action an actor should take to solve a problem. Hence, means and ends are inseparable. They coexist and one cannot be defined without taking the other into consideration - they are two names of the same reality [18].

3.4 Ends and Ends-in-View

Ends provide an activity – an adapted human action – with a purpose that suits the current situation. If an action has no meaning or purpose, the action becomes blind and disorderly. Having an aim for an activity is what Dewey calls ends or ends-in-view [18].

To Dewey, an *end-in-view* is conceived and tested in a process of inquiry. For this reason, it also indicates the aim of an activity is framed and bound within the problematic situation. The end-in-view thus establishes a course of direction for our activities to produce the solution we are committed to in the given situation. Ends-in-view define and deepen the meaning of activities, as they trigger evaluation and reflection. When we evaluate and reflect upon the results of our activities, we also learn new things about the end and the experiences we gained to get here.

Ends-in-view are dynamic and alive entities in the process of inquiry. Therefore they can change the course of activity to suit the current situation in connection to resolving the problem [17]. Inquiry elucidates both ends to be achieved and necessary means to achieving them.

3.5 Community of Inquiry

Dewey saw a *community of inquiry* as a group of individuals inquiring into problematic situations together [39].

A problematic situation causes the community to form and undertake inquiry. Any inquiry is socially contingent and has cultural and social consequences [17] as humans live in communities and take pleasure in the culture the association with others brings [17].

Knowledge emerges and exists within a social context and therefore requires agreement among those involved in the inquiry for legitimacy. *An inquirer in a given special field appeals to the experiences of the community of his fellow workers for confirmation and correction of his results* [17].

The social and cultural aspects of the environment will influence the perception of the problematic, means, and ends, utilized to transform idea into solution. If an idea and its meaning cannot be communicated and understood, the idea is nothing else but fantastic beyond imagination [17]. As a social activity, inquiry depends not only on the physical environment, but also on the traditions, organizations, customs and common practice embedded in the situation [17].

4 Illustration: Telerehab

Software innovation is about the exploration of problem domain needs and possible software-based ways to approach them. For this, we need not only to focus on how to develop solutions, but more importantly inquire into why we develop them and what we want to achieve.

We will illustrate our pragmatic approach via a very simplified example. The example is partly based on a thesis project where software engineering students worked on a system to support rehabilitative physiotherapy for post-surgery patients in their own home [26]. The idea was to improve rehabilitation effects while spending fewer resources on transportation at the same time.

Could a system be designed to help patients and therapists collaborate virtually? A system where patients exercise in their home under therapist supervision using an Internet connection?

We will use this case to illustrate how innovative solutions emerge as technological expertise is combined with problem domain insights. The gradual development is described in a series of prototypes, where each prototype reflects a deeper understanding of problems, means, and ends.

The project team consists of three people: Charlie, Ralph and Alex. Charlie is a physiotherapist and she represents the customer-side and flow of resources into the project. Charlie has agreed with Pat, a patient recovering from shoulder surgery, to discuss and try out various ideas, scenarios, and prototypes aiming to facilitate therapeutic sessions with Charlie in her office and Pat in his home.

The two other team members, Ralph and Alex, are software developers. Both of them are experts in software technologies whereas their a priori knowledge of the problem domain is limited. Being the senior, Alex is responsible for facilitating and

enacting an agile and iterative development process, and she is also the interface between the team, management, and external stakeholders.

We will describe the first iteration in the project, which results in the first prototype and ideas for a second prototype.

4.1 1st Iteration: X-ray

Physiotherapeutic rehabilitation requires the patient to do prescribed exercises correctly. Performed incorrectly, these exercises might be less effective or even worsen the condition. Furthermore, problems with doing an exercise correctly might indicate sideeffects from surgery, or recovery complications such as joint, tendon, and muscle problems related to fatigue and stress.

Based on such concerns the team could start out discussing how a system could support Charlie in instructing Pat on the movements of an exercise and deciding whether these steps are performed correctly.

Charlie knows of existing telerehab systems using web cameras and two-way video links. She suggests building a similar system where Pat will see her demonstrate the individual movements of an exercise, while she can see how he follows her lead. The audio would be used to explain and discuss issues.

The team agrees that this might work but Ralph asks Charlie if a two-dimensional image on her screen will be enough for her to evaluate Pat's moves. Charlie replies that it might for example be hard to see how much a limb twists in the joint during a particular movement. It might also prove difficult to see if Pat uses compensation movements to make the exercise easier.

After some discussion Ralph suggests using a Microsoft Kinect camera. A Kinect would compensate for some of the limitations of an ordinary video-feed by highlighting bones, joints, and movements of the patient. This would help the therapist to see how the exercise is performed and give precise instructions to the patient.

Alex and Ralph research the Kinect SDK and decide that the platform offers some interesting possibilities. The API provides easy access to the Kinect's sensors and supplies enhanced features such as skeleton tracking and gesture recognition. Also, as it integrates nicely with the .Net platform, it would be easy to interface with external software components. Building a system for a Kinect-based platform therefore seems to be viable design.

The team agrees to base a first prototype on the Kinect and they nickname this first prototype *X-ray* to reflect the primary motive for using a Kinect: To compensate for limitations in patient-therapist interaction based on a conventional video feed via human pose estimation. The Kinect is used to enhance those parts of the feed that are most important for a therapist in order to assess the movements of an exercise.

4.2 Testing and further development

Having built the prototype, the team invites Pat to test-drive it from his rural home. Charlie successfully instructs Pat, and she is able to monitor and correct his movements with sufficient precision.

Charlie asks Pat to carry on with the exercise to see if the movements change over time as Pat tires or simply forgets exactly how to do the exercise. Unfortunately, this part of the session suffers from a series of communication breakdowns ranging from bandwidth reductions and time lags to complete disruptions. Although the connection is reestablished after 5-15 seconds or less, these recurring disruptions make it impossible for Charlie to monitor Pat let alone offer him meaningful feedback.

After the test, the team and Pat discuss the prototype with mixed feelings.

Charlie is not enthusiastic about the system, although it does what she asked for – except for the connection problem. Obviously, the system allows the therapist to instruct the patient, but longitudinal monitoring of the patient requires stable high-quality connections. If breakdowns were frequent she could not see much use of the system.

Ralph and Alex are disappointed. The system does what it should but the connection problems are ISP-related and clearly out of scope for this project. Moreover, the system is merely a medium between the therapist and the patient, and seen as a tool it offers little support to the users compared to its potential. The platform is seriously underutilized and could offer much more. Ralph wonders if the system could be more stand-alone.

Pat jokingly regrets that Charlie could not be automated. Before the surgery, he used to play table tennis on his son's Kinect and enjoyed the thrill of competing against the machine. If Charlie were 'inside the box' somehow, he would be able to carry on exercising even if connections broke down. It could even be more fun compared to just being controlled by Charlie like a puppet on a string.

Alex replies that building a Charlie-avatar for the system would probably be out of scope for the project, but perhaps they could use some of the unused resources in the platform to copy parts of what Charlie would do. Ralph agrees and says that it should be fairly simple to have the system monitor Pat's movements and compare them to previously registered 'ideal' movements. After all, the skeleton data points from the Kinect could easily be converted to vectors and used in calculations where actual movement vectors are compared with 'ideal' vectors.

They decide to start working on a new prototype extending the first one. This time the system should autonomously monitor Pat's exercises and show on his screen how his movements compare with how they are supposed to be performed. Reflecting the addition of such features, they nickname the second prototype *Biofeedback*.

After a similar second iteration the team decides to continue on a third prototype – *Trend Analyst* comparing Pat's progress with other patients' data – and then a fourth one – *Timestretch* aggregating data to save time for Charlie. We will not describe these iterations in detail, but in the next section we will discuss how this work was facilitated by Essence – a methodology for software innovation – and how this methodology stands on pragmatic ideas.

5 Essence: Pragmatic software innovation

The pragmatic ideas presented in this paper can be linked to an ongoing effort to develop *Essence* – a methodology for software innovation [1-3].

In Essence, software innovation is a recurring process of inquiry in software teams. Essence therefore is aimed at facilitating such inquiry among team members. A basic assumption in Essence is that an innovative software team integrates software expertise and problem domain expertise. This combination requires experts from different domains to combine their knowledge in constructive ways.

Essence is based on three types of elements: Values, views, and roles. There are four values, four views, and four roles (see Table 1).

Value	**View**	**Role**
Reflection over requirements – working on ends gives a deeper understanding than found in requirements	*Paradigm* – underlying mental models of the problematic situation shared in the team	The *Child* is the main idea-developer for the problematic situation – not bounded by convention
Affordance over solution – every solution affords new options and potentials to be discovered via inquiry into the problem domain	*Product* – means and ends seen from the 'inside' with a focus on architectures, data, components, etc.	The *Responder* sees the problematic situation from a developer perspective, creates solutions and explores options
Vision over assignments – it is more important to share the end-in-view under conditions of learning and change than listing tasks	*Project* – plans, status, and priorities in the project based on a project vision that represents the end-in-view for the team	The *Challenger* prioritizes ends and approves the end-in-view in the problematic situation from a customer perspective
Facilitation over structuration – it is more important to facilitate inquiry and valuation of means and ends than following standard procedures	*Process* – facilitating idea generation and indeed idea evaluation and maturation via identifying potentials and supporting decision-making	The *Anchor* is liaison with outside stakeholders, and facilitates inquiry and valuation in the team

Table 1. Essence elements

Values are normative statements encouraging the team to focus on valuable choices at all times. The values in Essence seek to push the values in the agile manifesto [5] explicitly towards software innovation [2].

Views are analytical perspectives encouraging the team to view problems and possible solutions from several perspectives. The inspiration to use analytical perspectives comes from 3 directions: (1) the more than two thousand year old philosophical tradition of portraying all worldly structures as made from four elements thereby offering a generic totality consisting of mutually exclusive elements;

(2) the four perspectives on Software Engineering [7, 34]; and (3) Tidd et al.'s [45] distinction between four categories of innovations.

Roles are personal and seek to install a sense of personal responsibility for creating innovative solutions. Every team member has a permanent role, but they may temporarily take the only fleeting role in Essence: Child. The Child role is for creating new ideas even if the new ideas conflict with ideas and doctrines previously agreed upon in the project.

Usually a particular role is closely related to a particular view and value. The rows in Table 1 represent such triads. The following four sections therefore describe these triads with illustrations from the Telerehab case.

5.1 Reflection, Paradigm View, and the Child

The *Child* role is fleeting. Any team member has a permanent role but may temporarily take this role. Even people outside the team – for example the patient Pat – may act as Child on occasion and bring new insights and ideas to the team.

This role encourages anyone to propose new ideas even if in direct conflict with the principal shared understanding in the team about the problematic situation faced by the project. The Child role is one way to overcome indeterminate situations and suggest means and ends that may restore a situation to be determinate and thereby manageable for the team.

The *Paradigm* view is where the Child works. This View is used to inquire into the problematic situation, i.e. the challenges that brought the project into existence in the first place. The view is used to share insights from the problem domain, for example by describing users, needs, situations, scenarios, etc. When the first iteration started, Charlie – using the Child role – saw the paradigm as centered on instruction: How could Charlie show Pat the right way to do an exercise? The problem domain was explored at the Paradigm view and centered on how to exchange factual information between two people in different locations using a software-based system.

Such insights typically reflect determinate situations where the team feels confident in the problems to solve and how. Yet inquiry at this and the other views might bring about indeterminate situations. This was what happened when the test of the first prototype was obstructed by connection problems and Pat –as Child – pointed to an alternative paradigm for the project: *Learning*.

The Child role and the Paradigm view are both for *Reflection*. Reflection mirrors the inquiry into the problematic situation where insights regarding problems and needs emerge and bring about new ideas. The connection problems and Ralph's dissatisfaction with unused potential in the platform were two indeterminate situations that caused the team to inquire deeper into the situation.

As a value, Reflection encourages the team to consider who the users are, what to expect from them, how to scope the project, what value a solution could bring to customers and users, etc. The problems faced in this case could be understood from different paradigms: *Instruction* (showing how), *learning* (user empowerment), *coaching* (comparing with other patients), or *quality time* (spending less time observing and more time talking with the patient). Each paradigm reflects

fundamentally different notions of the situation at hand and which ends would be called for, and each paradigm is a product of our inquiries.

5.2 Affordance, Product View, and the Responder

The *Responder* role is permanent. This role is for developers and their main responsibility is to respond and design solutions to the challenges that the project confronts. In the case, both Ralph and Alex were responders, although Alex had extra responsibilities being the Anchor as well.

Responders work primarily at the *Product* view. This view is for designing architectures, deciding on components, algorithms, and data – always with a view to create and pursue potential wherever possible.

Their work is driven by *Affordance* – the aspiration to always ask if any component, algorithm, database, etc. could bring more to the situation than originally anticipated. A design at a given time might afford more potential later than was recognized when the design actually took place.

Ralph suggested using a Kinect to enable Charlie to see Pat's movements better. When the connection problems emerged, he suggested using the unused potential of the platform to make Pat's side of the system more stand-alone. This technically motivated move led to a change in the paradigm for the project to focus on learning and empowering Pat.

At a later time, Charlie might mention that she wants data from the system to be exported to the hospital database for research on treatments and complications. Alex might suggest using data from the database to compare Pat's progress with similar records in the database.

Alex and Ralph's continuous inquiry into the problematic situation entails hands-on experience with the means at their disposal. This experience leads to a deeper understanding of the technology and what the technology affords of new possibilities in the situation they find themselves in. Affordance thereby might lead to a new paradigm for the next prototype: *Trend Analyst*. This way, the data requested by Charlie turns from ends (deliverables) wanted from the system into means used in the system to service Charlie and Pat better.

5.3 Vision, Project View, and the Challenger

The *Challenger* role is a permanent role for a person representing the problem domain. Charlie is the Challenger in our case. Acting on behalf of the customer she not only contributes with knowledge and resources, but also with the challenge as well as acceptance of solutions as they emerge.

The challenge is the problematic situation the team is thrown into by the Challenger. Charlie started out by asking for support for collaboration between therapist and patient in separate locations. As the team inquired into this challenge in iterations, the vision and the project scope developed into more and more valuable solutions.

The Challenger works at the *Project* view where features are prioritized and the project vision is represented, maintained, and shared within the team as well as with external stakeholders. The vision represents the *end-in-view* for the project at a given time. In the case described here, each prototype name represents end-in-views as they change over time from X-ray over BioFeedback and Trend Analyst to TimeStretch.

The value for the Project view and the Challenger is *Vision*. It is important to share the end-in-view among the members of the team. A list of requirements can hardly reflect the overall idea of the project let alone help the team spot subtle changes in project goals as problems and solutions emerge from inquiry. Visions – perhaps expressed as metaphors – are well-known ways to share this overview [6, 44].

5.4 Facilitation, Process View, and the Anchor

The *Process* view is for *facilitating* the team itself in making the best out of the work at the three other views. It is used for idea generation and in particular for evaluations. Criteria for acceptance, for determining the qualities sought for in a solution are examples of contributions to this view.

Evaluations are when visions are tested in the process of inquiry. Core elements in evaluations include the choice of criteria and how they are used. Developing criteria is as much a reflective activity as is the design of solutions.

The *Anchor* is liaison between the team and outside stakeholders and responsible for facilitating productive working in the team. As such, the Anchor insists on fair and sober discussions in the team to ensure sensible decisions and choices.

Alex was Anchor and helped the team evaluate strengths, weaknesses, opportunities, and threats for each prototype. Ralph pointed to unused potential in the first prototype and this helped inspire an answer to the communication problems. Later, when Charlie required that data about Pat's progress be registered in a hospital database, Alex suggested using this opportunity to improve the system itself for analyzing Pat's progress compared to similar patients. A proposal that lead to the *Trend Analyst* prototype.

6 Conclusion – Essence and Pragmatic Software Innovation

This paper suggests a pragmatic approach to software innovation. Using Essence as methodological foundation – based on agile principles and in particular incremental development – Essence was used to illustrate inquiry into a problem domain identifying technological options.

In incremental projects, sprint deliverables serve as prototypes. A prototype reflects the perceived scope of and needs in a problem domain and also a vision for how to answer these needs. The prototype can be evaluated in connection with a sprint review meeting, and scope, needs, vision, and configuration may change as the team inquires deeper.

Our illustration started out with a plain problematic situation and a team of technology and problem domain experts. The problematic situation offered a

challenge with known scenarios and needs, but after testing the first prototype, new scenarios and needs were discovered. The inquiry changed the problematic situation.

The values, views, and roles in Essence highlight key characteristics of pragmatic software innovation.

Values encourage the team to inquire into the problematic situation and study the problem domain and its needs (reflection); to consider if there is untapped potential in the technological answers (affordance); to establish an end-in-view allowing the team to work determined under uncertainty (vision); and to constantly evaluate the fit between the problems and the answers under conditions of constraints (facilitation).

Views focus on artifacts – shared representations offering common ground for inquiry. Are problems well represented, consistent, and meaningfully scoped on the Paradigm view? Are design options represented on the Product view in a way where they can be assessed, and does the design offer flexibility to embrace such options? Is the vision on the Project view represented in a format easy to understand and maintain? Are tools and techniques for idea development available on the Project view and are evaluation criteria visible and representative for the current end-in-view?

Roles focus on the personal reflection and responsibility. The Challenger represents problem domain knowledge in the team and maintains the project vision. Responders represent technological insight and suggest answers for the problematic situation. The Anchor is responsible for the process and for ensuring fair evaluations and timely criteria. Finally, the Child – the fleeting role at the Paradigm view – is key for inquiring with a free hand.

Inquiry in Essence is both theoretical – using the expertise of the team members – and experimental – using every artifact to reflect and learn. Evaluation in Essence therefore is an important way to improve the design, and serve as a stimulus to creativity in software design [31]. Evaluations and judgments may be according to predefined or ad hoc criteria, and they may even be tacit and intuitive [40].

In Essence any design forms the basis for a new beginning. A prototype is a transaction with the problematic situation [37]. Testing the prototype in the illustration necessitated the system to be stand-alone. This new design created opportunities – new ends – that were not anticipated when the project started. As the vision develops, it will usually grow increasingly stable as the product is tested with customers and users. Still, the vision remains changeable and allows the project to adapt to changes by guiding without excessive detail [23].

In this paper we have tried to show how Essence facilitate a pragmatic approach to software innovation – an approach characterized with learning across knowledge domains and with incremental development of not only ideas – but also solutions – throughout the span of the project. We believe that pragmatic software innovation offers a way to opportunistically pursue innovation in everyday projects.

References

1. Aaen, I.: Essence: Facilitating Software Innovation. European Journal of Information Systems 17, 543-553 (2008)

2. Aaen, I.: Software Innovation - Values for a Methodology. In: Aanestad, M., Bratteteig, T. (eds.) 156. pp. 72-86. Springer Berlin Heidelberg. 2013
3. Aaen, I.: Roles in Innovative Software Teams – A Design Experiment. Human Benefit through the Diffusion of IS Design Science Research, IFIP WG 8.2 + 8.6 International Working Conference, Perth, Australia 73-88 (2010)
4. Aspray, W., Mayadas, F., Vardi, M. Y.: Globalization and Offshoring of Software. Association for Computing Machinery, Job Migration Task Force (2006)
5. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: Manifesto for Agile Software Development. (2001)
6. Beck, K.: Extreme Programming Explained: Embrace change. Addison-Wesley, Reading, MA (2000)
7. Bernstein, L., Yuhas, C. M.: People, Process, Product, Project - The Big Four. In: Bernstein, L., Yuhas, C. M. (eds.) Trustworthy Systems Through Quantitative Software Engineering. pp. 39-71. John Wiley & Sons, Inc. New York 2005
8. Biskjaer, M. M., Dalsgaard, P.: Toward A Constrating Oriented Pragmatism Understanding Of Design Creativity. The 2nd International Conference on Design Creativity (ICDC2012) 65-74 (2012)
9. Bourque, P., Fairley, R. E. (Eds.): *SWEBOK v3.0 – Guide to the Software Engineering Body of Knowledge*. Washington: IEEE Computer Society. (2014)
10. Campos, P.: Promoting innovation in agile methods: two case studies in interactive installation's development. International Journal of Agile and Extreme Software Development 1, 38 (2012)
11. Carr, N. G.: IT Doesn't Matter. Harvard Business Review 81, 41 - 49 (2003)
12. Conboy, K., Morgan, L.: Beyond the customer: Opening the agile systems development process. Information and Software Technology 53, (2011)
13. Conboy, K., Wang, X., Fitzgerald, B.: Creativity in Agile Systems Development: A Literature Review. In: Dhillon, G., Stahl, B., Baskerville, R. (eds.) IFIP Advances in Information and Communication Technology. pp. 122-134. Springer Boston. National University of Ireland Galway Ireland 2009
14. Couger, J. D.: Creative problem solving and opportunity finding. Decision making in operations management series. Boyd & Fraser Pub. Co., Hinsdale, Ill. (1994)
15. Couger, J. D., Dengate, G.: Measurement of Creativity of I.S. Products. Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences 4, 288-298 (1992)
16. Couger, J. D., Higgins, L. F., McIntyre, S. C.: (Un)Structured Creativity in Information Systems Organizations. MIS Quarterly 17, 375-397 (1993)
17. Dewey, J.: Logic: The Theory of Inquiry (1938). 1953, 1-549 (1925)
18. Dewey, J.: Human nature and conduct: An introduction to social psychology. The Modern Library, New York (1957)
19. Dewey, J.: Essays in experimental logic. SIU Press, (2007)
20. Dewey, J., Bentley, A. F.: Knowing and the known. 111. Beacon Press Boston, MA, (1960)
21. Goldkuhl, G.: Design research in search for a paradigm: Pragmatism is the answer. In: M, H., Donnellan, B. (eds.) Practical Aspects of Design Science. pp. 84-95. Springer. Berlin 2012
22. Gorschek, T., Fricker, S., Palm, K., Kunsman, S. A.: A Lightweight Innovation Process for Software-Intensive Product Development. Software, IEEE 27, 37-45 (2010)
23. Hey, J. H. G.: Framing innovation: negotiating shared frames during early design phases. Journal of Design Research 6, 79-99 (2007)
24. Hickman, L. A.: John Dewey's pragmatic technology. The Indiana series in the philosophy of technology. Indiana University Press, Bloomington (1990)

25. Highsmith, J., Cockburn, A.: Agile software development: the business of innovation. Computer 34, 120-127 (2001)
26. Jensen, R. H., Brodersen, K. H.: Responder's Inquiry - Spikes. Department of Computer Science Master Thesis, 83 (2013)
27. Lim, Y.-K., Stolterman, E., Tenenberg, J.: The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas. ACM Trans. Comput.-Hum. Interact. 15, 1-27 (2008)
28. Mahaux, M., Maiden, N.: Theater Improvisers Know the Requirements Game. Software, IEEE 25, 68 - 69 (2008)
29. Maiden, N., Ncube, C., Robertson, S.: Can Requirements Be Creative? Experiences with an Enhanced Air Space Management System. Software Engineering, 2007. ICSE 2007. 29th International Conference on 632-641 (2007)
30. Maiden, N., Robertson, S.: Integrating Creativity into Requirements Processes: Experiences with an Air Traffic Management System. Proceedings of the 2005 13th IEEE International Conference on Requirements Engineering (RE'05) 105-114 (2005)
31. McCall, R.: Critical Conversations: Feedback as a Stimulus to Creativity in Software Design. Human Technology 6, 11-37 (2010)
32. Misra, S. C., Kumar, V., Kumar, U., Mishra, R.: Goal-Driven Measurement Framework for Software Innovation Process. Journal of Information Technology Management XVI, 30-42 (2005)
33. Pikkarainen, M., Codenie, W., Boucart, N., Alvaro, J. A. H. (Eds.): *The Art of Software Innovation - Eight Practice Areas to Inspire your Business*. Berlin, Heidelberg: Springer Berlin Heidelberg. (2011)
34. Pressman, R. S.: Software engineering: a practitioner's approach. McGraw-Hill Higher Education, Boston, Mass (2005)
35. Rose, J.: Software Innovation: Eight work-style heuristics for creative system developers. Software Innovation, Aalborg University, Department of Computer Science, (2010)
36. Samuelsen, R.: A Pragmatist Contribution to Science, Technology and Innovation (STI) Studies. (2013)
37. Schön, D. A.: Designing as reflective conversation with the materials of a design situation. Knowledge-Based Systems 5, 3-14 (1992)
38. Schön, D. A.: The Reflective Practitioner: How professionals think in action. Basic books, (1983)
39. Schön, D. A.: The Theory of Inquiry: Dewey's Legacy to Education. Curriculum Inquiry 22, 119-139 (1992)
40. Schön, D. A., Wiggins, G.: Kinds of seeing and their functions in designing. Design Studies 13, 135-156 (1992)
41. Shneiderman, B.: Creativity support tools: accelerating discovery and innovation. Communications of the ACM 50, 20-32 (2007)
42. Sternberg, R. J., Lubart, T. I.: The concept of creativity: Prospects and paradigms. In: (eds.) Handbook of Creativity. pp. 3-15. Cambridge University Press. Cambridge 1999
43. Swanson, E. B., Ramiller, N. C.: Innovating Mindfully with Information Technology. MIS Quarterly 28, 553-583 (2004)
44. Takeuchi, H., Nonaka, I.: The new new product development game. Harvard Business Review 64, 137-146 (1986)
45. Tidd, J., Bessant, J. R., Pavitt, K.: Managing innovation: integrating technological, market and organization change. Wiley, Hoboken (2005)