ENGAGED SCHOLARSHIP: ACTION DESIGN RESEARCH FOR NEW SOFTWARE PRODUCT DEVELOPMENT

Completed Research Paper

Maria Moloney
Trinity College Dublin
Ireland
Mmolone2@tcd.ie

Liam Church
Escher Group Ltd
Dublin 8
Liam.Church@EscherGroup.com

Abstract

Action Design Research (ADR) has been termed a new research method for generating prescriptive design knowledge through building and evaluating ensemble IT artifacts in an organizational setting. This paper demonstrates the use of the ADR methodology in a New Software Product Development (NSPD) environment. The main argument of this paper is to investigate whether ADR facilitates engaged scholarship among IS academics and practitioners in a high pressure environment such is NSPD. Due to increased pressure on organizations to reduce costs and their time to market, the NSPD environment is increasingly not conducive to engaged scholarship. Finding an arrangement where researchers and practitioners can successfully develop both theory and artifact concurrently and in relative harmony is often crucial to the success of the new product and/or to the rigor of the research outcomes. The paper makes a contribution by providing a demonstration of successful research-practitioner engagement through the use of ADR.

Keywords: Engaged scholarship, Action Design Research, software development

Introduction

Across most industries, the survival of businesses is largely determined by their success in New Product Development (NPD) (Cooper, 2001; Ulrich and Eppinger, 2004). NPD is the process of bringing a new product to market, including idea generation, concept development and testing, business analysis, prototype and market testing, and plans for product commercialization and launch (Pavlou and Sawy, 2006). NPD is a strategic process wherein firms integrate disparate inputs from research and development, engineering, and marketing to jointly develop and launch new products (Clark and Fujimoto, 1991). The importance of NPD is demonstrated not only by the vast amount of academic and business literature that refers to the field (Pavlou and Sawy, 2006; Krishnan and Ulrich, 2001), but also by the extent of interest it attracts from the large and diverse sets of management, engineering, and manufacturing specialists in any given industry (Nambisan, 2003).

The software industry is no exception. In fact, given the fast-paced and ever evolving IT marketplace, NPD is of particular strategic importance to software companies. New Software Product Development (NSPD) needs to be dynamic in order to remain competitive in such an unpredictable marketplace. Ironically, research on the facilitation of innovative software development seems still in its infancy (Aaen, 2008). A study by Lee and Xia (2005) suggests that systems development teams are inefficient and reactive in dealing with business changes and therefore have problems supporting business adaptations to shifting demands. Admittedly, this study was conducted over seven years ago and the environment may be considerably different by now, but more recent research indicates that this no longer needs to be the case as current software development technologies are opening up new opportunities for innovation by allowing for product changes and adaptations even late in the new product development lifecycle (Aaen, 2008). This dynamic new NSPD environment, in itself, also gives rise to some challenges for NSPD companies. An example is the need to create a supportive eco-system to surround the dynamic software development lifecycle. Additionally, this supportive ecosystem should in turn enable the results produced by the NSPD function to become consistent and reproducible. This paper looks at an NSPD project that set about putting in place such a supportive eco-system by ensuring the project was supported and guided by both practitioners and academics throughout the entire project lifecycle. In so doing, the software development company sought to achieve consistency in (1) developing design principles for new software products, and (2) being effective and successful in high pressure environments.

Similarly, from an academic perspective, IS research has been criticized for having little influence on practice (Cole et al., 2005). One approach to achieving more relevance for IS research is to conduct research using appropriate research methods that balance the interests of both researchers and practitioners. The Action Design Research (ADR) methodology has been shown to facilitate the reconfiguring of a company's IS function into an efficient and effective function that adequately supports company processes (Sein et al., 2011). However, ADR has not yet been applied in an NSPD setting. This paper seeks to ascertain whether the ADR methodology facilitates engaged scholarship during an NSPD project. In other words, this paper seeks to answer the following question. *Is the ADR research methodology an effective methodology for engaging researchers and practitioners in a high pressure environment such is new software product development?*

The structure of this paper is as follows. First, the new software development lifecycle is examined, along with the concepts of engaged scholarship and action design research. This examination is followed by an overview of a case study, which used the ADR methodology to guide the research effort during the development of a new software product entitled RiposteTrEx™. Finally, the findings of the case study are outlined and discussed.

Literature Review

The New Software Development Lifecycle

The last decade (2012-2002) has witnessed an increasing conflict between traditional and modern approaches to software development. This conflict is most evident in the debates over Software Process

Improvement (SPI) and Agile Software Development (ASD) (Aaen, 2008). At the centre of these debates are distinctions between predictable manufacturing and new product development (Larman, 2004), which according to Aaen (2008) reflect recent developments in software technologies that appear to be having an effect on software innovativeness. The traditional software development paradigm aims for predictable and documented software production, while agile software development aims for innovative and dynamic software development (Aaen, 2008). While this may be a fairly strong statement by Aaen, and it is debatable whether traditional methods for NSPD ever really achieved predictability or complete documentation. Nevertheless, improved methods, tools and techniques for software development have indeed allowed for greater flexibility in the development of software products in recent years.

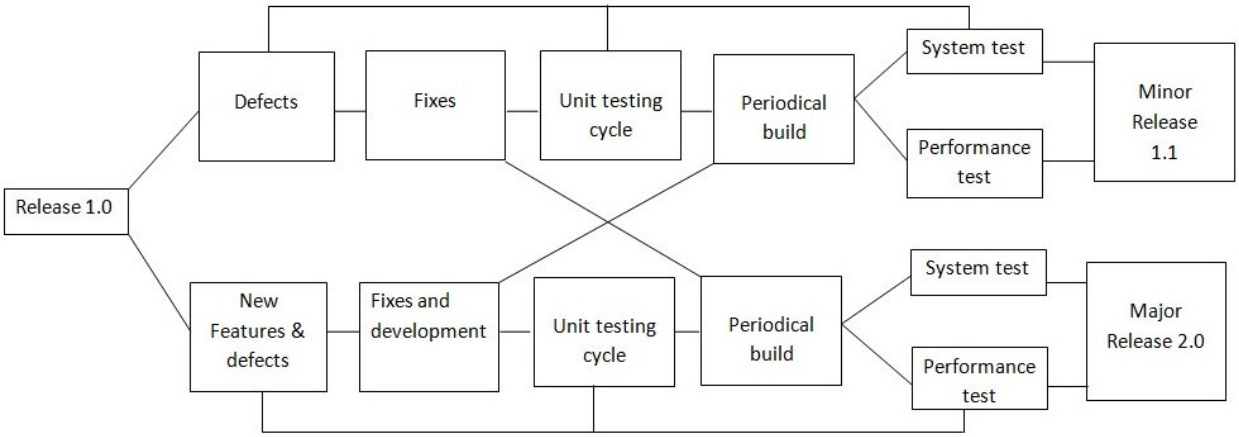

Figure 1 Product Development Life Cycle, adapted from Dayani-Fard (2001)

Figure 1 illustrates a typical product development cycle plan, while agile processes employ user feedback, rather than planning, as their primary control mechanism. This feedback is driven by regular tests and releases of the evolving new software product (Larman, 2004). Figure 2 shows the testing lifecycle for agile software development.

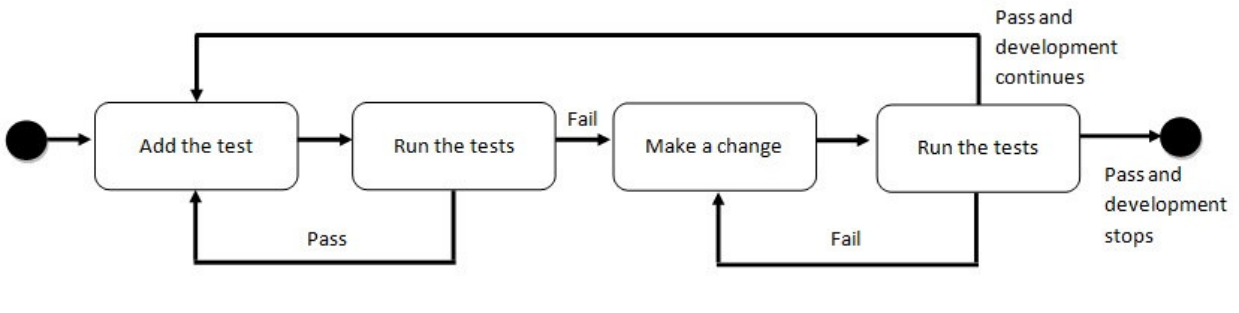

Figure 2 Testing lifecycle for agile software development

Traditional software product development places great emphasis on initial elicitation of product requirements and design in order to get the architecture right from the start. It also relies heavily on product verification and validation to ensure that project deliverables are consistent and comply with requirements. Agile development focuses on customer involvement (Beck and Andres, 2005), emergent design (Fowler, 2003), test-infected development (Beck and Gamma, 2000) and sees testing as confirming that user and customer needs are met (Aaen, 2008). Figure 3 shows how the start and initiation of an agile software development lifecycle is immediately followed by the construction of the software.

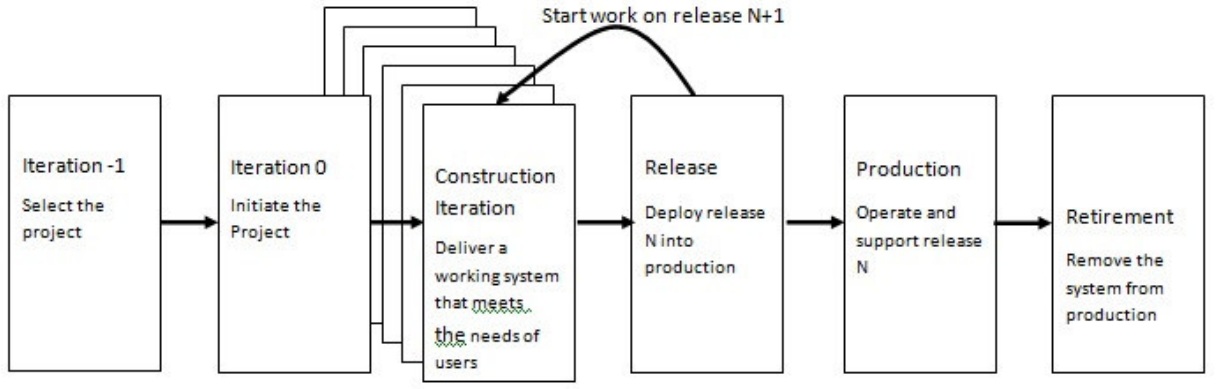

Figure 3 Agile software development lifecycle taken from (Ambler, 2011)

In traditional project management, focus is often placed on cost and schedule, which includes resource allocation, risk planning, quality planning, and plan management and more (Abran and Moore, 2004). In summary, collecting comprehensive information up-front is of primary importance. Agile development is more fast-paced and focuses on adaptive planning with built-in feedback loops, and is based on incremental development with more frequent releases (Larman, 2004).

Agile development enthusiasts argue that building stable software processes stands at the centre of the traditional paradigm. They point to the strategy of software process improvement as proof of this (Abran and Moore, 2004). In agile development, processes are not independent standalone objects, but rather they are practices that evolve dynamically with the team as it adapts to the particular circumstances (Aaen, 2008). This could be as a result of the increased pressure on software development companies to keep up with growing consumer demands and increased competition in the marketplace. Regardless of the cause, such companies have had to speed up their time to market considerably in the last number of years. This often results in an increasingly pressurised environment where getting the new product to market supersedes all other concerns. Such an environment makes it difficult to carry out research and achieve the research goals of the company. At the same time, there is increasing interest among academics surrounding how to conduct research that can be directly applied to organizations, practitioners, and policymakers (Cole et al., 2005). This interest grows not only from a desire by scholars to conduct research that has greater applicability and responsiveness to contemporary social issues but also results from external pressures from funders, university administrators, and taxpayers for institutions of higher education to become more accountable to the needs of the local communities in which they are located (Cole et al., 2005).

So, for the benefit of both practitioners and academics, new research methodologies need to be explored and tested for their suitability to engaged scholarship for NSPD. The next section firstly gives a brief overview of the notion of engaged scholarship and then shows how the principles of engaged scholarship can be employed to guide action design researchers during an NSPD project.

ADR as a form of Engaged scholarship

ADR is an *action-oriented* research methodology. The purpose of any action-oriented research is to generate knowledge that can be used to address practical concerns of organizations (Small and Uttal, 2005). Action-oriented research projects can focus on any issue or concern and usually have some type of change (e.g., individual, social, organizational) as an ultimate goal. Common to nearly all action-oriented research approaches is that they involve some type of collaboration between researchers and practitioners.

A guiding principle of action-oriented research is the idea that research can never be value free. Action-oriented research calls for heightened attention to how power and trust shape the relationship between partners and how this relationship shapes the findings of the project. The practitioners' perspective is

kept at the forefront to ensure that the researcher's view does not dominate the way knowledge is created and defined. For the research findings to be useful, they must be constructed in a way that is meaningful to the industry partner.

Action-oriented research such as ADR assumes that valid knowledge comes from a variety of sources and that academic and practitioners each possess unique and valuable knowledge, practices, and insights that can contribute to the success of a research project (Strand et al., 2003). DeVault (1999) observed that researchers need specialized skills and must devote more sustained attention to any investigation than is usually practical for practitioners. Conversely, using research results effectively to promote change requires the pragmatic evaluative and strategic skills of practitioners, honed through more daily participation in front-line work than most researchers experience. Together, researchers and practitioners can combine their different kinds of knowledge and skills to produce insightful and usable findings (Small and Uttal, 2005).

Van de Ven's (2007) description of engaged scholarship reflects the values of action-oriented research. He describes it as a participative form of research for obtaining the views of key stakeholders to understand complex problems. By exploiting differences between these viewpoints, he argues that engaged scholarship produces knowledge that is more penetrating and insightful than when researchers work alone. According to Costello et al. (2011) engaged scholarship has a number of facets:

- a form of inquiry where researchers involve others and leverage their different perspectives to learn about a problem domain;
- a relationship involving negotiation, mutual respect, and collaboration to produce a learning community;
- an identity of how scholars view their relationships with their communities and their subject matter.

According to Mathiassen, et al. (2008) engaged scholarship undertakes the challenge of conducting research that advances science while at the same time enlightens professional practice. In Van de Ven's view, in order to achieve this, researchers must increase the likelihood of advancing knowledge for science and practice by engaging with practitioners and other stakeholders in four ways;

- *Informed basic research* that is undertaken to describe, explain, or predict a social phenomenon;
- *Collaborative basic research* is similar to informed basic research, but it entails a greater sharing of power and participation between researchers and stakeholders;
- *Design and evaluation research* focuses on normative knowledge related to design and evaluation of policies, programs, and models for solving practical problems within a profession; and
- *Action research* applies intervention to address a problem of a specific client while at the same time contributing to academic knowledge.

ADR engages practitioners predominantly through both design and evaluation research and action research. ADR focuses on normative knowledge related to design and evaluation of policies, programs, and models and uses this knowledge to apply interventions to address a design problem of a specific client, while at the same time contributing to academic knowledge. It also entails considerable sharing of power and participation between researchers and stakeholders.

Mathiassen & Nielsen (2008) argue that the degree of collaboration with stakeholders is an important distinguishing feature of engaged scholarship in IS research. However, variation in stakeholder collaboration does not only help distinguish different forms of 'basic research', it applies equally well to distinguishing forms of 'design and evaluation research' and 'action research'. They propose a similar way to classify forms of engaged scholarship within IS research which is based on their underlying knowledge interests:

1. *Practice research* which focuses on understanding IS practices with the purpose of informing or advising relevant stakeholders.
2. *Design research* which focuses on designing various forms of software products with the purpose of supporting stakeholders engaged in IS practices.

3. *Action research* which focuses on changing IS practices through problem solving in response to specific client needs.

From this classification, ADR straddles the latter two classes, design research and action research. From this perspective, ADR can be said to focus on changing IS design and development practices through problem solving in response to specific client needs. Sein et al. (2011) argue that ADR is a research method that explicitly recognises software products as being shaped by the interests, values, and assumptions of a wide variety of communities of developers, investors, and users, without letting go of the essence of design research i.e. innovation and dealing with a class of problems and systems. These interpretations of the ADR methodology link it to the concept of engaged scholarship i.e. "a participative form of research for obtaining the views of key stakeholders to understand complex problems" (Van de Ven, 2007: p. 9). Figure 4 shows Van de Ven's representation of engaged scholarship for problem solving. According to him, engaged scholarship involves four stages. These stages can happen in any sequence:

1. Problem formulation – situate, ground, diagnose, and infer the research problem by determining who, what, where, when, why, and how the problem exists up close and from afar.
2. Theory building – create, elaborate, and justify a theory by abductive, deductive, and inductive reasoning.
3. Research design – develop a variance or process model for empirically examining the alternative theories.
4. Problem solving – communicate, interpret, and apply the empirical findings on which alternative models better answer the research question about the problem.

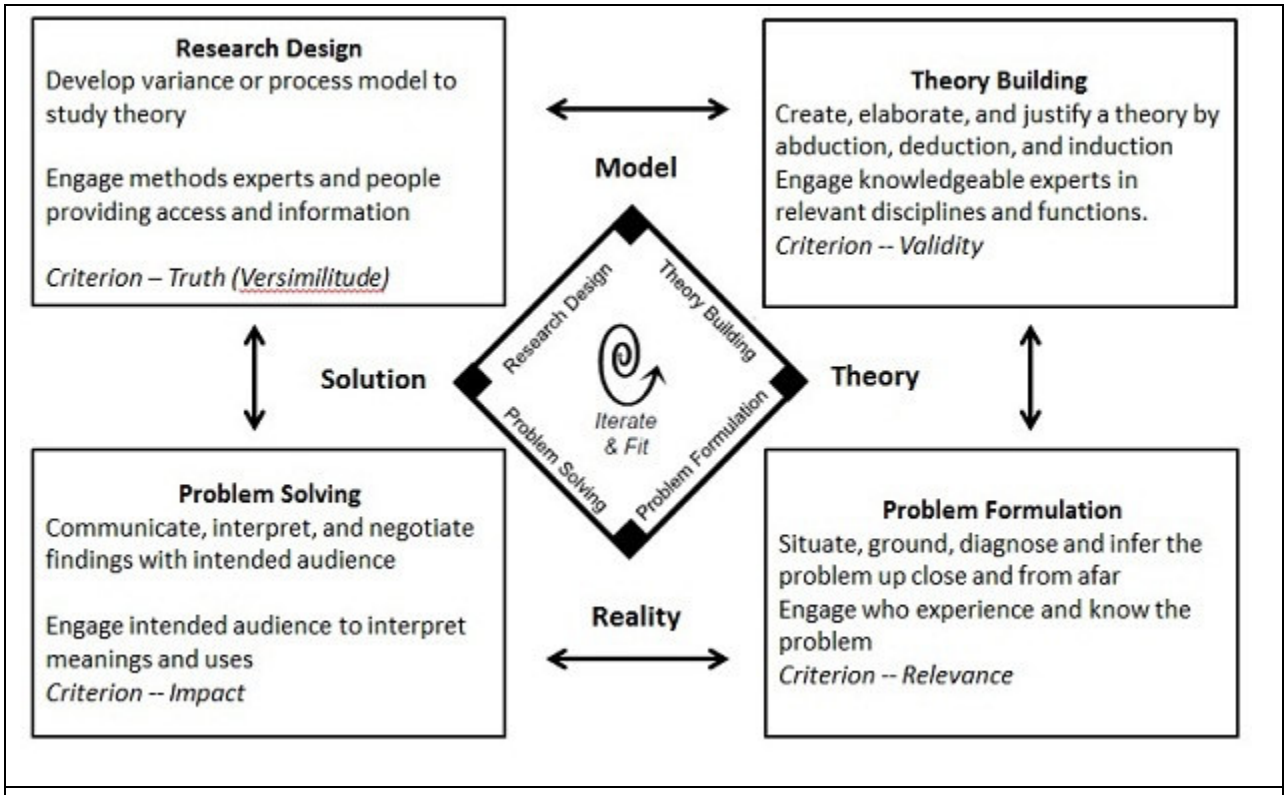

Figure 4 Engaged scholarship for problem solving (Van de Ven, 2007)

When engaged scholarship takes place within the context of a software design and development environment, Sein et al, (2011) claim ADR as an appropriate research methodology. ADR effectively combines two rich research traditions from the action research domain and the design science domain.

Choosing ADR for New Software Product Development

The case study described in the next section initially used a combination of action research and insights from design science research to guide the NSPD process. In mid 2011, it was decided to interpret the project through the lens of the ADR methodology. The researchers found that retrospectively fitting the project into the ADR structure was relatively easy. On the contrary, once permission was acquired from top management to use the ADR methodology going forward, structuring the project around the methodology requirements ran into some problems, which are discussed in the 'Findings' section of this paper.

Typically, ADR deals with two IS challenges: (1) it addresses a problem situation encountered in a specific organizational setting by intervening and evaluating; and (2) it constructs and evaluates an software product that addresses the class of problems typified by the encountered situation (Sein et al., 2011). When ADR is used in the NSPD environment for producing new software products, the challenges differ slightly. In this new environment, ADR: (1) addresses a need or service requirement encountered in a specific "real-world" or practical setting that has not yet been met by information technology; and (2) constructs and evaluates a software product that addresses the need or requirement typified by the encountered setting, until and after the software product goes to market. The ADR process must continue, for a short time and on a less intense level, after the software product initially goes to market as it is generally accepted in agile software development that the first version of a software product often does not meet requirements in terms of functionality and user expectation. The feedback from the first release can be used to test feature design and help improve marketing and customer support (Jiang, Scheibe and Nilakanta, 2011).

This paper interprets ADR as complementing the NSPD process in the following ways: (1) by providing a structured method that focuses on the building, intervention, and evaluation of a commercial software product during the entirety of its turbulent development lifecycle, and (2) by facilitating the expression and validation of the theoretical intent of the researchers during this turbulent lifecycle, and (3) by assisting the validation of the software product by super users and end users before and after commercialisation. ADR facilitates the effective and simultaneous building of a software product and the derivation of its underlying theory. In this way, it provides for the possibility of engaged scholarship in turbulent environments such as NSPD.

The ADR approach used in the RiposteTrEx™ product development project follows the four stage structure outlined by Sein et al. (2011) of which the ADR method consists:

- Stage 1: problem formulation;
- Stage 2: building, intervention, and evaluation;
- Stage 3: reflection and learning;
- Stage 4: learning formalisation.

Overview of the Escher Group Holdings Plc Case Study

Founded in 1989, Escher Group has been developing software applications for over 20 years. They have offices in five countries worldwide. Traditionally, they provided software solutions to postal operators and today their software is licensed for use in 230,000 workstations across thirty three countries including Germany, Norway, Denmark and Ireland. Escher's message based division delivers RiposteTrEx™, a digital mailbox solution designed to encourage citizens, businesses, governments and international agencies to collaborate securely online, on mobile devices and/or through social networking channels. The RiposteTrEx™ platform enables commercial business to communicate securely and deliver e-services to their customers. It enables governments to develop their e-government strategies by delivering citizen-centric e-services and it enables the citizen to interact in the digital world commercially, socially and politically and all from a central location. Their goal is to use RiposteTrEx™ to empower citizens and enable them to interact and transact safely with anyone or any organization linked up to the secure RiposteTrEx™ network.

Table 1 shows a breakdown of the full software product lifecycle for RiposteTrEx™ using the four stages of

the ADR methodology. The lifecycle of the RiposteTrEx™ development project is then described in detail.

Table 1 Breakdown of ADR Method		
Problem formulation (Stage 1)	Data collection and documentation through interviews and meetings with key stakeholders • Senior public service personnel • Prospective users	20 months
April 2009 – Nov 2010	Days on site	28 days
Building, Intervention, and Evaluation (Stage 2 or BIE stage)	Iterative prototype development, testing & evaluation Articulation of design principles for public ICTs (tentative) Focus groups with a cross section of the public	7 months
Dec 2010 – June 2011	Days on site	52 days
Reflection and learning (and ongoing BIE) (Stage 2&3)	Ongoing iterative development, testing & evaluation Ongoing articulation and formalisation of design principles for public ICTs Submission of research to workshops for academic feedback.	5 months
July – November 2011	Days on site	24 days
Formalisation of learning (and ongoing BIE) (Stage 2&4)	Ongoing iterative development, testing & evaluation Generalisation of design principles for public ICTs (ongoing articulation of new design principles)	
Jan 2012 – ongoing	Days on site	18 days
Outcomes and reiteration of stages 2-4	Software product transition to production stage of the product lifecycle. Conclusion of iterative development, testing & documentation Wrap up meeting Start-up of the next development lifecycle using the data gathered during the NPD lifecycle for version 2 of the product. Days on site	8 days
	Total days of research project	130 days

Stage 1: Problem Formulation

The trigger for the first stage of the ADR method occurs when a problem is perceived in practice or anticipated by researchers. It provides the impetus for formulating the research effort. The problem

formulation stage identifies and conceptualizes a research opportunity based on existing theories and technologies (Sein et al., 2011).

The RiposteTrEx™ project was triggered by a problem anticipated by the CEO, Liam Church, of Escher Group Holdings Plc. regarding how citizens' personal data online will need to be increasingly protected from third party exposure but available to the individual citizen in future public e-services. This insight was the product of his career of working closely with various public sector organisations worldwide. Based on this insight, the company developed a prototype software infrastructure, which safeguards citizens' personal information when interacting on the Internet and when availing of electronic public services but which also makes a citizen's data available to him/her for their own personal use. Details of this product can be found at (Church and Moloney, 2012). He consequently successfully sold a licence for this prototype to a national governmental agency, resulting in funding to further develop the prototype to the 'proof of concept' stage. As a result of this funding and after almost 2 years of in-house development, the *RiposteTrEx™ project for commercialisation* started in earnest.

At the beginning of the RiposteTrEx™ project and during the latter stage of the problem formulation process, an IS researcher was embedded in the company to work closely with the project team and to both impart knowledge to the team and learn from the teams experience, expertise and tacit knowledge of their chosen field. The researcher collaborated with two other academics in a university during the RiposteTrEx™ project. However, only one researcher was embedded in the organisation.

Principle 1: Practice-Inspired Research

This principle emphasizes viewing field problems (as opposed to theoretical puzzles) as knowledge-creation opportunities. ADR seeks these opportunities at the intersection of technological and organizational domains, although the degree of novelty can vary across the two. The intent of the ADR team should not be to solve the problem per se as a software engineer or a consultant might. Neither should it be only to intervene within the organizational context of the problem. Instead, the action design researcher should generate knowledge that can be applied to the class of problems that the specific problem exemplifies. As a result, the research activity is problem inspired (Markus, Majchrzak and Gasser, 2002).

Even though the problem formulation stage within the company started around April 2009, practice-inspired research commenced when the IS researcher join the team in August 2010. The IS researcher was called upon to generalize the problem and to help inform the future design of a software product aimed at resolving the particular class of problems which the specific problem represented, for the RiposteTrEx™ case study outlined in this research, this was a privacy-enhanced, secure software product for providing public e-services (Church and Moloney, 2012).

By embedding the researcher in the world of the practitioner at such an early stage, an interface between the scientific world of the researcher, marked by theoria, and the practical world of the practitioner, marked by praxis, was created (Mårtensson and Lee, 2004; Costello, Conboy and Donnellan, 2011). This encouraged the mixing of praxis and theoria throughout the rest of the project stages, with the ultimate aim of producing a theory-ingrained artefact.

Principle 2: Theory-Ingrained Artefact

This principle emphasizes that the ensemble artefacts created and evaluated via ADR are informed by theories. To define what constitutes a theory, Sein et al (2011) used Gregor's (2006) criterion of "the power to generalize." Gregor considers systems of statements that allow generalization and abstraction to be theories. The level of predictive power can vary, and theories can range from universal laws to ones with more restricted scope, such as the technology acceptance model (TAM). Sein et al (2011) believe that Gregor's theories of Type IV (explanation and prediction theories) or Type V (design theories) are likely candidates for ADR. This project developed a Type V, design theory for informing the design of artefacts for providing public e-services (Church and Moloney, 2012).

In late August 2010, a weeklong brainstorming session took place at the company headquarters to kick-off both the development effort for the software product and the collaboration between industry and

academia. At this meeting, senior software developers and designers, security experts, IS researchers, and consultants, senior management, and senior public servants came together to conceptualise the proposed product and to tackle the expected challenges to be faced by the project team. Following the brainstorming week, the momentum of this stage was continued into the next stage through the use of Computer Mediated Communication (CMC) channels where everything relating to the product such as documentation, the development environment and prototype testing environments was shared across four countries.

Table 2 shows a breakdown of the data collection and analysis that took place during the RiposteTrEx™ development project. It also outlines the length of time, in terms of days, it took to complete both the data collection and analysis phase and the ADR and iteration phase. There are currently 3 pilot studies being conducted at local government level in Ireland.

Table 2 Data Collection and Project Summary

Phase 1 Aug 2010 – Nov 2010	Number of Formal Interviews with Key Stakeholders - Managers in public organisations (9 interviews) - End users (20 interviews) Estimated hours	30 35
Aug 2010 – March 2011	Literature review of public service provision, software development, new product development, ADR, and engaged scholarship. Estimated hours	120
Jan 2011 – March 2011	Number of focus groups with the end users (8 individuals in each group) Estimated hours	8 12
	TOTAL DATA COLLECTION HOURS	167 hours
Phase 2 2010-2012	ADR and iteration Estimated days on site	 112 days
	In-house testing and evaluation (conservative estimation) Estimated days (x 4 testers + 4 developers)	 3,840 days
Phase 3 March 2011 – Nov 2012	Number of pilot projects conducted for systems evaluation Estimated days of evaluation (1 pilot will last 2 years)	3 600 days

Stage 2: Building, Intervention, and Evaluation (BIE)

The BIE stage for a new software product differs slightly from the BIE stage outlined by Sein et al. (2011). For NSPD, this stage is carried out as an iterative process in a target environment, but it interweaves the building of the software product, with intervention and evaluation in a *real-world setting* as opposed to an organisational setting described by Sein et al. (2011). The outcome of the BIE stage is the realized design of the software product. During BIE, the problem and the software product are continually evaluated, and the design principles are articulated for the chosen class of systems, which the software product represents.

This initial BIE stage for RiposteTrEx™ lasted seven months, during which time a number of iterations of BIE cycle took place. Senior management, the development team, the product testers and the IS

researcher worked together to define, refine, develop and deploy prototypes which tested various features of the system and observed how users responded to the features and iterated, until finally all features of the product were implemented, tested and functioning correctly. At the seven month point, the software product had reached the 'proof of concept' stage of the development lifecycle. At this point, the product was demonstrated to the governmental body that had commissioned the project and a series of focus groups with end users were conducted. These demonstrations and trials determined whether the software product would be further developed or whether the product would be postponed indefinitely. Fortunately, the product feedback was for the most part, positive. However, considerable reworking of the product was carried out as a result of the feedback from the governmental body representatives and the end user focus groups. This reworking embodies the reciprocal shaping principle described next.

Principle 3: Reciprocal Shaping

This principle originally, in Sein et al. (2011), emphasized the reciprocal influences exerted by the two original domains: the software product and the organizational context. With NSPD, the two domains consist of the software product and the real-world context where the need or requirement satisfied by the software product exists. Using the feedback from the focus groups carried out at the proof of concept stage, the ADR team engaged in recursive cycles of decision-taking and used a combination of the user feedback and design constructs to shape their interpretation of the real-world environment. They then used this increasing understanding of the real-world environment to influence the selection of design constructs for the product. The starting domain, however, for these iterations is different for NSPD and reflected the company's and the researcher's perception of the specific need that the software product was being designed to meet.

Principle 4: Mutually Influential Roles.

This principle points to the importance of mutual learning among the different project participants. During the RiposteTrEx™ project the mutually influential roles of researcher and practitioner were clearly defined, however, no role was mutually exclusive. A good working relationship was formed among both researchers and practitioners and all project members demonstrated a genuine curiosity and interest in the roles of their counterparties. The practitioners understood the reason for the researcher taking part in the project and were eager to impart their knowledge for the benefit of the final product. Likewise, the researcher understood that the practitioners were under pressure to meet deadlines and project milestones. Any recommendations given by the researcher to improve the design of the product were formally imparted to the project team during project meetings so minimal disruption to the team occurred. Likewise, any ideas proposed by practitioners to improve the product were imparted to the researcher through weekly progress meetings.

Principle 5: Authentic and Concurrent Evaluation

This principle emphasizes a key characteristic of both ADR and agile software development: evaluation is not a separate stage of the research process that follows building. Instead, decisions about designing, shaping, and reshaping the software product and intervening in real-world settings should be interwoven with ongoing evaluation. For the RiposteTrEx™ project, the BIE stage continued long after the proof of concept stage was complete. In fact, the BIE stage overlapped with stage 3 and 4 of the ADR method. Table 1 clearly shows this overlapping. Ongoing iterative development, testing & evaluation of the software product took place right up to the final testing of the product before it went live.

It was found during the project that controlled evaluation is difficult to achieve in both an ADR project and a NSPD project due to the emergent nature of the software product. Therefore, evaluation opportunities should be sought following natural controls. Consequently and similar to the agile software development philosophy, authenticity is a more important ingredient for ADR than controlled settings (Sein et al., 2011, Ambler, 2011).

During the RiposteTrEx™ project, evaluation opportunities ranged from formative cycles such as in-house functional and UI testing at the earlier stages of the project, to conducting summative evaluation cycles such as focus groups with end users and user trials and pilot projects at the latter stages. Eight

focus groups were carried out at the proof of concept stage of the product and the feedback from these groups was fed back into the product development. Version 1.0 of the product was then installed and piloted in the offices of three different public bodies. Installation and administration of the product during these pilots was handled by company project managers, software engineers and administration staff from the public bodies. The ADR researcher was in charge of collecting feedback from the participants of the pilots to help in the refinement of the product before it was officially released to the users. Two out of the three pilots are ongoing, one of which is due to last for two more years. The third pilot is now complete and was successful. The government body involved in the pilot is now investigating ways to roll out the system across the department.

Stage 3: Reflection and Learning

The reflection and learning stage moves conceptually from building a solution for a particular instance to applying that learning to a broader class of problems. For RiposteTrEx™ this stage was continuous and interweaved with stage four. In fact, conscious reflection on the problem framing, the theories chosen, and the emerging product occurred through all four stages of the ADR process. It was critical to constantly reflect and learn from the development process of the new software product in order to ensure that all contributions to knowledge were identified and rigorous. The research process was regularly adjusted based on early evaluation results to reflect the increasing understanding of the software product and to constantly develop the product with a view to the problem it needed to solve.

Principle 6: Guided Emergence

This principle captures a vital trait of ADR, which is the interplay between two seemingly conflicting perspectives: that of design and emergence. It emphasizes that the software product will reflect not only the preliminary design (see Principle 2) created by the ADR team but also its ongoing shaping by end users, perspectives, and participants, and by outcomes of authentic, concurrent evaluation. This guided emergence makes the NSPD lifecycle dynamic and the lines between the different stages of the ADR method begin to overlap. This principle emphasizes that the ADR team should be sensitive to signals that indicate the need for ongoing refinement and react accordingly by continually updating and refining the design principles for the specific class of systems they are made to address (Sein et al., 2011).

Stage 4: Learning Formalisation

The objective of the fourth stage of ADR is to formalize the learning. Following Van Aken (2004), the situated learning from the RiposteTrEx™ project was further developed into general solution concepts for the design of public e-service systems (Church and Moloney, 2012). The ADR researcher outlines the design accomplishments realized in the software product in a previously published paper and describes in the present paper the organizational outcomes, which formalize the learning. The former outcomes can be characterized as design principles and with further reflection, as refinements to theories that contributed to the initial design.

Principle 7: Generalized Outcomes

Generalization is challenging because of the highly situated nature of ADR outcomes that include organizational change along with the design and development of a software product. The resulting set is, by definition, a bundle of properties in different domains. This set represents a solution that addresses a problem (Sein et al., 2011). Both solution and problem can be generalized. This move from the specific-and-unique to generic-and-abstract is a critical component of ADR. Sein et al. (2011) suggests three levels for this conceptual move: (1) generalization of the problem instance, (2) generalization of the solution instance, and (3) derivation of design principles from the design research outcomes. All three levels were attempted during the RiposteTrEx™ project and constitute papers that are either published (Church and Moloney, 2012; Moloney and Church, 2012), or currently works in progress.

Findings

The ADR method provides a new approach to the challenge of engaged scholarship. It is suitable when the practitioner seeks to identify guidelines for the future development of a specific class of systems. However, ADR is a new method and relatively untested and this study is intended to contribute to the debate on the approach.

The first finding was that the RiposteTrEx™ project members found that after the project was retrospectively re-structured, by the research academics, according to the guidelines of the ADR methodology (around mid 2011) a significant improvement occurred in relation to organizing the project and emphasizing focus points during the project. However, this result did not happen easily. It became clear early in this 'restructuring phase' that the ADR stages needed to be more cyclical and iterative in nature than anticipated. As new features were being added right up to the final stages of the project, stage 2 frequently needed to be revisited. In fact, stage 2 continued up until the end of the project and stages 3 and 4 were also iterative in nature and alterations to the outcomes in all three of these stages were constantly being made. The dividing line between the learning and reflection stage (stage 3), and the formalization of learning stage (stage 4) was never clear. This caused some negotiation to take place among the researcher and the practitioners regarding the need for timely sign-off for newly developed features.

After the addition of any new feature, the embedded researcher recommended further cycles of gathering users' feedback via focus groups or at least by filling in a five minute survey of their user experiences, and the practitioners, as a result of resource and budget constraints were urging immediate sign-off once in-house testing showed no functionality failures in the new features. Further data analysis did not take place and sign-off occurred once all in-house testing of the new features was complete. The need for such a dynamic research process, and the complexity of sharing ownership of the research project with industry partners was a challenge for the researcher in that her need for rigor conflicted with the practitioners need for meeting project deadlines and adequately serving their customers. The transition to using the structured ADR methodology became a process of negotiation and compromises between the academic and her industry partners. Occasionally, the knowledge building goals of the academic partner were subordinated to the practice and change-related goals of the industry partner.

Yet, once these transitional issues were resolved, all parties agreed that the resulting project outcomes were more rewarding than before the implementation of the ADR methodology. This is seen to be because of the heightened efficacy and efficiency that the research methodology provided for achieving the research and practical aims of the project. One of the attractions of the ADR methodology for the academic was that it provided an opportunity to use her research skills effectively in an applied research scenario and to see the results of her skills quickly.

In this project, the practitioners found it beneficial that the researcher had published the project outcomes in academic conferences and journals. They felt it added credence to their new product and showed their dedication to developing software grounded in best practices and state of the art technology.

From an academic perspective, the ADR methodology facilitated dyadic learning, i.e. when one member of a dyad develops knowledge then the other member develops that same knowledge also. The practitioners learned a great detail about engaged scholarship during the project and the researcher learned a lot about new software product development. This type of knowledge sharing breaks down barriers and facilitates increased collaboration among industry and academia.

There was a learning curve for the embedded researcher in the project in that she found that she needed to be more reflexive in such an applied environment. Being reflexive meant that the researcher had to critically examine how her perspective, position, and presence had some bearing on the research process as well as on her relationship with her industry partners. As the project progressed, the researcher developed more expertise in being reflexive and she observed that the RiposteTrEx™ project experienced a reduced number of conflicts as a result. This reduction in conflict enhanced the validity, rigor, and credibility of the findings.

The ADR method provided the industry partner with an explicit method of transforming their product ideas into a clear reality. By articulating and exploring opinions in an open environment with colleagues and researchers, ideas were more easily captured, developed and transformed into design constructs. This

type of analysis has been emphasised by Lester and Piore (2004) as aiding the innovation process.

It was found that a critical part of using the ADR methodology for an NSPD project was the timely dissemination of the findings back to the industry partner. For the academic partner to maintain a reputation in the IS industry as a reliable collaborator, the data needs and products for the industry partner must be given priority. After the data was collected, the researcher took care to provide information to the industry partner first before focusing on her own academic goals. The data was primarily analyzed according to the current needs of the industry partner, which frequently changed during the project. When used in such a fashion, the increased use of ADR may not only serve as a way to make the activities of IS academics more meaningful and useful to the IS Industry but could also enhance the industry's view of IS research.

In addition to providing findings to industry partners in a timely manner, it is important to present them in a style and format that is easily understood and applicable. The traditional training of most social scientists often does not prepare them to be creative in how they disseminate the findings of a study. Scholarly papers filled with dense narratives, and disciplinary jargon are common in the social sciences (Small and Uttal, 2005). Traditional academic training focuses on preparing scholars for technical expertise and communicating almost exclusively with other scholars (Small and Uttal, 2005). Very often, little attention is given to selecting, analyzing, translating, and targeting research findings for non-academic audiences. A number of techniques were devised by the researcher during the RiposteTrEx™ project to enhance the likelihood that the findings from the project would be read and used by the industry partners and other stakeholders:

- First, the researcher had to be selective about what was presented in formal documents and presentations. Decision makers and practitioners are busy and do not have the time to read through long reports and decide which findings are most relevant (Small and Uttal, 2005).
- The researcher, with the help of the project members had to prioritize the findings. The meaning of the findings and implications for practice needed to be clearly stated and not left to the reader to decipher. An executive summary at the beginning of a report was found to be an excellent way for imparting the main findings.
- The researcher had to be proactive in disseminating the findings. Rather than waiting for people to seek out the findings, it was more effective to identify the audience and bring the findings to them in ways that were consistent with how they typically received information. Working closely with the RiposteTrEx™ management team, the researcher came up with some creative ways of sharing research findings, by giving PowerPoint presentations to specific teams. These presentations were then re-used by the industry partner for audiences such as funders and customers. A central repository on the intranet was created where all research findings and papers are now accessible to all members of the company. A monthly newsletter featuring the study's findings was sent to interested parties. The findings were also shared in a one-day workshop on RiposteTrEx™ given to internal staff and to potential customer (on two separate occasions).

Escher Group Holdings Plc received a strong response to the new software product from their customers and they view the design and development project of RiposteTrEx™ as a complete success. Top management and the project team witnessed firsthand, the benefits of collaborating with researchers for improving their product offering. Consequently, the company is now engaging more enthusiastically with research institutions in the region. They are currently members of a consortium collaborating with a group of research institutes to establish an applied research centre. This centre will conduct similar ADR projects for the benefit of industry and academia.

The researcher involved in the project gained a diverse and multifaceted understanding of the ICT industry from the perspective of both experienced management and development teams involved in the project. The tacit knowledge of the project members, built up over a lifetime in the industry, provided a fresh perspective to the research findings. Their perception of the market place and its future direction was perceptive and enriched the research findings.

The researcher also found that the ADR methodology when used in such contexts helps to broaden the academic function, focusing less narrowly on the scholarship of discovery and expanding the academic research mission to more fully embrace the scholarship of application and dissemination (Small and

Uttal, 2005). It is believed that the adoption of ADR as a legitimate research methodology will not only enhance the standing of academic institutions in the eyes of funders and the public, but more importantly, it will increase the potential of academics to impact positively on the IS industry.

Conclusions

This paper applies the ADR research method in a NSPD environment. The ADR method was used in a high pressured environment like the NSPD environment to ensure sustained and comprehensive practical and research outcomes for NSPD projects.

When referring this ADR study to the engaged scholarship diamond model in figure 4, it can be argued that this study meets all the necessary criteria for an engaged scholarship study:

- Problem formulation: The length of the research study (over two years) ensured there was adequate engaging of researchers and practitioners and diagnosing of the problem.
- Theory building: A design theory for the development of ICTs for public services was developed from this project. The details of this design theory are published in another academic paper and outside the scope of this research (Church and Moloney, 2012).
- Research design: the ADR methodology was used for this project. This is a new method but one highly regarded among the IS community, given the quality of the journal in which the method was first published.
- Problem solving: The research engaged the intended audience in solving the problem in hand, i.e. Escher Group Holdings and various government bodies through pilot studies. The findings were disseminated through direct engagement and academic publications.

The study initially provides an overview of the new software product development environment and how it is evolving with ever increasing pressure on companies to bring new products to market. It then briefly describes engaged scholarship and how to conduct research by engaging with practitioners. The RiposteTrEx™ case study, which was conducted on the premises of a software development company and which employed the ADR methodology to conduct research during the development lifecycle is then presented. Lastly the findings of the research are outlined and discussed.

It can be argued that a limitation of this research is that is it is based on a single case and thus it is difficult to argue that the findings are generalizable (Lee and Xia, 2005; Mintzberg, 1996). Future research hopes to test these findings with other NSPD projects as they are formed within the company and elsewhere.

Bibliography

Aaen, I. (2008) 'Essence: facilitating software innovation', European Journal of Information Systems, vol. 17, p. 543–553.

Abran, A. and Moore, J. (2004) Guide to the Software Engineering Body of Knowledge: 2004 Version, Washington: SWEBOK IEEE Computer Society.

Ambler, S.W. (2011) The Agile System Development Life Cycle (SDLC), [Online], Available: HYPERLINK "http://www.ambysoft.com/essays/agileLifecycle.html" http://www.ambysoft.com/essays/agileLifecycle.html [5th April 2012].

Beck, K. and Andres, C. (2005) Extreme Programming Explained: Embrace Change, Boston, MA.: Addison-Wesley.

Beck, K. and Gamma, E. (2000) 'Test-infected: programmers love writing tests', in Deugo, D. (ed.) More Java gems, New York, NY, USA: Cambridge University Press.

Church, L. and Moloney, M. (2012) 'Public Value Provision: A Design Theory for Public e-Services', Annual Services Research and Innovation Institute Global Conference, San Jose, California, USA.

Clark, K.B. and Fujimoto, T. (1991) Product Development Performance, Cambridge, MA: Harvard Business School Press.

Cole, R., Purao, S., Rossi, M. and and Sein, M. (2005) 'Being Proactive: Where Action Research Meets Design Research', International Conference on Information Systems Proceedings, University of Nevada, Las Vegas, USA.

Cooper, R.G. (2001) Winning at New Products, Cambridge: MA: Perseus Publications.

Costello, G., Conboy, K. and Donnellan, B. (2011) 'Dialogical Action Research as Engaged Scholarship: An

Empirical Study', The Thirty Second International Conference on Information SYstems, Shanghai, China.
Dayani-Fard, H. (2001) A Framework for Managing Software Product Development, [Online], Available: HYPERLINK "http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.5082" http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.5082 [11 November 2011].
DeVault, M. (1999) Liberating method: Feminism and social research, Philadelphia, USA: Temple University Press.
Fowler, M. (2003) 'Design – who needs an architect?', IEEE Software, vol. 20, no. 5, pp. 11-13.
Gregor, S. (2006) 'The Nature of Theory in Information Systems', MIS Quarterly, vol. 30, no. 3, pp. 611-642.
Jiang, Z., Scheibe, K.P. and Nilakanta, S. (2011) 'The economic impact of public beta testing: the power of word-of-mouth', Thirty Second International Conference on Information Systems, Shanghai, China.
Krishnan, V. and Ulrich, K.T. (2001) 'Product Development Decisions: A Review of the Literature', Management Science, vol. 47, no. 1, pp. 1-21.
Larman, C. (2004) Agile and Iterative Development: A Manager's Guide, Boston, MA.: Addison-Wesley.
Lee, G. and Xia, W. (2005) 'The ability of information systems development project teams to respond to business and technology changes: a study of flexibility measures.', European Journal of Information Systems, vol. 14, no. 1, pp. 75-92.
Lester, R.K. and Piore, M.J. (2004) Innovation: The Missing Dimension, Cambridge, massachusetts. USA. : Harvard University Press.
Markus, M.L., Majchrzak, A. and Gasser, L. (2002) 'A Design Theory for Systems That Support Emergent Knowledge Processes', Management of Information Systems Quarterly, vol. 26, no. 3, pp. 179-212.
Mårtensson, P. and Lee, A.S. (2004) 'Dialogical Action Research At Omega Corporation', MIS Quarterly, vol. 28, no. 3, pp. 507-536.
Mathiassen, L. and Nielsen, P.A. (2008) 'Engaged Scholarship in IS Research', Scandinavian Journal of Information Systems, vol. 20, no. 2, pp. 1-18.
Mintzberg, H. (1996) 'Managing government, governing management', Harvard Business Review, May-June, p. pages 75–85.
Moloney, M. and Church, L. (2012) 'Informational Privacy Preservation through Universal Service Obligations', Journal of Internet Technology and Secured Systems, vol. 1, no. 1/2.
Nambisan, S. (2003) 'Information Systems as a Reference Discipline for New Product Development', MIS Quarterly, vol. 27, no. 1, March, pp. 1-18.
Pavlou, P.A. and Sawy, O.A.E. (2006) 'From IT Leveraging Competence to Competitive Advantage in Turbulent Environments: The Case of New Product Development', Information Systems Research, vol. 17, no. 3, September, pp. 198-227.
Sein, M.K., Henfridsson, O., Purao, S. and Rossi, M. (2011) 'Action Design Research', MIS Quarterly, vol. 35, no. 1, pp. 37-56.
Small, S.A. and Uttal, L. (2005) 'Action-Oriented Research: Strategies for Engaged Scholarship', Journal of Marriage and Family, vol. 67, no. November, p. 936–948.
Strand, K., Marullo, S., Cutforth, N., Stoecker, R. and Donohue, P. (2003) Community-based research and higher education., San Francisco, CA, USA: Jossey-Bass.
Ulrich, K.T. and Eppinger, S.D. (2004) Product design and development, Fifth Edition edition, Irwin/McGraw-Hill.
Van Aken, J.E. (2004) 'Management Research Based on the Paradigm of the Design Sciences: The Quest for Field-Tested and Grounded Technological Rules', Journal of Management Studies, , vol. 41, no. 2, p. 219–246.
Van de Ven, A.H. (2007) Engaged scholarship: a guide for organizational and social research, London: Oxford University Press.