AIS SIGPRAG
Pre-ICIS Workshop 2016
"Practice-based Design
and Innovation of
Digital Artifacts"

Software Embedded Evaluation Support in Design Science Research

Leona Chandra Kruse (1) Jonas Sjöström (2)
Amir Haj-Bolouri (3) and Per Flensburg (4)

(1) University of Liechtenstein, Liechtenstein
(2) Uppsala University, Sweden
(3) University West, Sweden
(4) University West, Sweden

Abstract

Even though the practice of integrating evaluative features into software has long been applied in commercially available software, it is still underrepresented in IS community. This paper presents a framework for built-in evaluation support. We seek to incorporate the idea of 'the self-evaluating artifact' into the design science research (DSR). We are aware of the challenges of evaluation of socio-technical systems and take this issue into consideration in our framework. Our framework is the result of conceptualizations drawing from the evaluation discourse in DSR. We also demonstrate our ideas through a built-in evaluation support mechanism designed and used in a DSR project in the Swedish healthcare sector.

Keywords: built-in evaluation, socio-technical evaluation, design science research, design guidelines

1 Introduction

IS design in general emphasizes the process of defining, designing, implementing and evaluating architecture, components and features of an information system. Evaluation is one of the crucial components in IS design (Venable, Pries-Heje, & Baskerville, 2012). This is due to the nature of design and implementation, where needs and requirements are expected to be fulfilled through evaluative forms of activities. But evaluation is not only relevant in terms of IS design. In this paper, we focus on evaluation as a core activity in design science research (DSR). DSR researchers need to demonstrate the efficacy and utility of their designed artifacts (Hevner, March, & Park, 2004; Venable et al., 2012), where artifact quality implicitly signals the value of the abstract knowledge embodied in the artifact (Venable et al., 2012). Throughout the course of DSR, there has been a vivid discourse about evaluation in the DSR community. Evaluation frameworks have emerged and been recommended for conducting sufficient DSR (Pries-Heje, Baskerville, & Venable, 2008; Sonnenberg & vom Brocke, 2012; Venable et al., 2012; Venable, Pries-Heje, &

Baskerville, 2014). Hence, evaluation is an essential part of IS design in general, and the overall DSR approach in particular, where evaluation may be conducted in different manners depending on the artifact at hand, its use context, and the underlying research questions.

Available frameworks for evaluation in DSR provide thorough guidelines for planning and executing evaluation, and for interpreting evaluation results. While covering evaluations for both social and technical aspect of IS artifacts, many of these DSR evaluation frameworks have not sufficiently addressed the prospect of embedding evaluation features into the software. The fact that built-in evaluation support is underrepresented, if not underexplored, in DSR is indeed unfortunate.

The idea of self-evaluating artifact – that is, an artifact that has been modified to support its own evaluation – has long been applied in commercially available software (e.g. operating systems and word processors that sends error reports to developers when exceptions occur in the software). Given the objective to evaluate an artifact, it is intuitively appealing to use that artifact as a means for evaluation; i.e. as a means to collect data about its use and to facilitate interaction between its users and its designers. Evaluation activities typically include the need to study user behavior, and to collect the viewpoints of users. Our basic idea is that the software artifact - in actual use - may thus be an excellent instrument to support the collection of data in evaluation activities.

However, we need to be aware of the challenges in integrating evaluation mechanisms into software given the socio-technical character of design science research (Lee, Thomas, & Baskerville, 2015; Robey, Anderson, & Raymond, 2013). Recent discourse on the notion of IS brings about potential to refresh our way to conceptualize and eventually to evaluate artifact. On the one hand it is suggested to unpack an IS artifact into its technical, social, and information components (Lee et al., 2015), on the other hand the plea to retire the mention of the notion of "IS artifact" is articulated in order to become specific and precise about the goal, form, and function of the information systems under scrutiny (Alter, 2015). Either way the challenges remain as to how we can evaluate socio-technical systems by incorporating evaluation mechanisms into software.

It is indeed apparent that the needs to automate some facets of the evaluation practice in DSR has not been satisfied due to the lack of framework to guide designers or design science researchers[1] to embed evaluation support into their software articats. With regards to this identified tension, we pose the following questions:

1. How can information systems (IS) evaluation be supported?

and specifically:

2. How can evaluation support be built into the artifact?

[1] In this paper we follow the definition proposed by Hevner, et al. (2004) that states design science "creates and evaluates IT artifacts intended to solve identified organizational problems" (p. 77). Given this definition, we use the terms "design science researchers", "designers", and "researchers" interchangeably, as they refer to the same idea within the scope of this paper.

In addressing the research questions, we disclose an underexplored dimension of evaluation in DSR: the integration of evaluation mechanisms into artifact instantiations. We synthesize various elements of the DSR evaluation discourse into a framework for software-embedded evaluation support (SEES) with focus on the typology of SEES. We provide an expository instantiation of the framework within a DSR project, and discuss implications for both practice and research.

The paper proceeds as follows. In section 2, we provide a general overview of types of evaluation in IS DSR and proceed to some examples of self-evaluation healthcare information systems. In section 3, we account for our framework of built-in evaluation support. In section 4, we provide an expository instantiation of the framework, followed by a concluding discussion.

2 Supporting Information Systems Evaluation

2.1 Evaluation in Design Science Research (DSR)

There are a number of different purposes for evaluating IS artifacts in Design Science Research (DSR), as well as there is a variety of different methods, strategies and activities for conducting the actual evaluation process. In this paper we provide a rough overview of the discourse on the evaluation of artifact in DSR literature and later on focus on topics relevant to the purpose of building evaluation support into the artifact. Therefore, this section provides a literature review that is neither complete nor comprehensive.

Remenyi et al. (1999) identify two common and important categories of evaluation: (1) formative vs. summative evaluation, and (2) ex ante vs. ex post evaluation. A distinction between the two categories lies in the nature of reasoning and strategies for evaluation. Venable et al (2014) have developed and explicated a framework for evaluation in design science (FEDS), together with a four-step evaluation design process, which is based on an analysis and synthesis of work on evaluation in DSR and other domains of IS (Remenyi & Sherwood-Smith, 1999; Smithson & Hirschheim, 1998; Stufflebeam, 2003). The framework aids DSR researchers by offering a strategic view of DSR evaluation. The strategic view is based on two different dimensions: (1) functional purpose, which incorporates formative vs. summative evaluation, and (2) evaluation paradigm, which incorporates naturalistic vs. artificial evaluation.

The formative perspective of FEDS captures the possibility to reduce risks by evaluating early, before undergoing the effort of actually building and strictly evaluating an instantiation of a design. The summative perspective offers a possibility for evaluating the instantiated artifact in reality, and not just in theory. In contrast to the formative and summative perspectives, naturalistic evaluation methods offer the possibility to evaluate the real artifact in use by involving real users solving real problems. Artificial evaluation methods offer the possibility to control potential variables more carefully, and prove or disprove testable hypotheses, design theories, and the utilization of design artifacts (Venable et al., 2014). Figure 1 summarizes and illustrates the constitution of FEDS.

Venable et al's (2014) framework with evaluation strategies highlights the idea of choosing a present, or building an own, DSR evaluation strategy that maps certain criteria for evaluation. For instance, the DSR researcher needs to identify whether the evaluation shall be conducted in a realistic environment together with real users, or in an artificial environment together with prospective users.

Other DSR evaluation methods such as the ones presented by Peffers et al (2012) emphasize the distribution of evaluation method types, based on the artifact type (e.g. method, instantiation, construct). They provide guidance for how to conduct DSR evaluation, based on the purpose of artifact use and utility. However, Venable et al' (Venable et al., 2014) FEDS incorporates a similar notion of evaluation guidance. Furthermore, Venable et al's (2014) FEDS extends the previous ideas of Pries-Heje et al (2008), Venable et al (2012) and Peffers et al (2012). Hence, the FEDS approach is a suitable source of inspiration for a framework for built-in evaluation support.

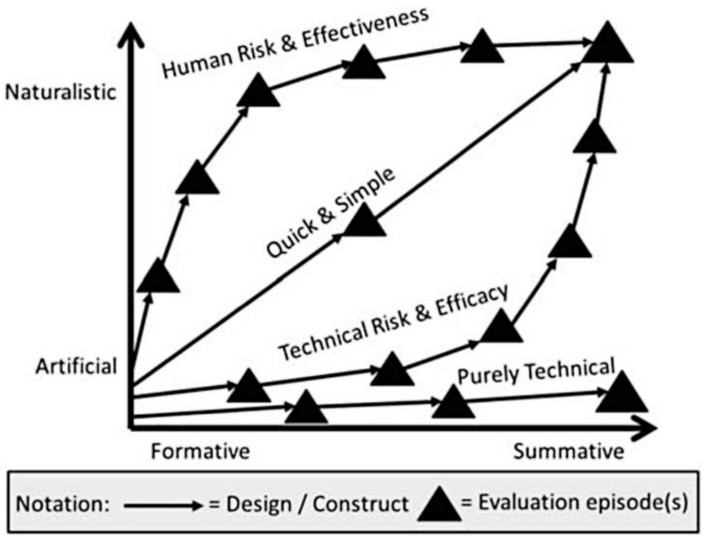


Figure 1: FEDS (Framework for Evaluation in Design Science) with evaluation strategies (Venable et al., 2014)

2.2 Supporting Formative and Summative Evaluation

Let us clarify the scope of our evaluation support by positioning it in relation to a set of evaluation dimensions derived from the DSR evaluation discourse. In the attempt to automate the evaluation of artifact, we mainly deal with naturalistic ex-post evaluation (Pries-Heje et al., 2008; Venable et al., 2012) which has the character of real users, real problems, and real systems. While such evaluation is considered to be the best evaluation of effectiveness and identification of side effects, it also comes with the highest cost and a potential risk for participants (Venable et al., 2012). Naturalistic ex-post evaluation can be carried using various methods, including (but not limited to) well-known research methods such as action research, case study, focus group, participant observation, ethnography, phenomenology, and qualitative or quantitative surveys. We position our discussion within the practice approach to DSR (Iivari, 2015) as well, where we focus on the effects an artifact has in in its use context, and the meaning which is ascribed to it by its various stakeholders.

We propose automated data collection during the lifetime of an artifact. Thus it may be used in a formative manner, e.g. as part of an action design research cycle of building-intervening-evaluating (Sein, Henfridsson, Purao, Rossi, & Lindgren, 2011), or for summative purposes to demonstrate the qualities of an artifact after a period of use. However, despite the idea that evaluation mechanisms are embedded in the IS artifact, evaluation does not necessarily concern only artifact in the sense of a software instantiation.

Despite the focus on evaluation of a software instantiation, we emphasize that the ultimate goal of DSR evaluation is to demonstrate the value of abstracted design knowledge, e.g. in the form of design theories, design principles, or technological rules. Artifact-centric evaluation is a means to demonstrate qualities of abstracted concepts. We subscribe to the view that "*when an artifact is evaluated for its utility in achieving its purpose, one is also evaluating a design theory that the design artifact has utility to achieve that purpose. From the point of view of design theory, a second purpose of evaluation in DSR is to confirm or disprove (or enhance) the design theory*." (Venable et al., 2012)

In a similar vein, the discussion on the nature of artifact to be evaluated becomes indispensable. The idea of supporting evaluation in DSR concerns primarily socio-technical artifacts, even though – to some extent – it can be applied to pure technical artifact as well. While a socio-technical system indeed involves human actors, there are many examples for artifacts that do not involve a human actor, for instance, algorithms used as part of a software system. What differentiates such agents from human agents is the process underlying their task execution. These so-called automata execute certain operation only if certain pre-determined condition is met, therefore the entire system is algorithmic in its nature.

In fact, a similar distinction has been pointed out by Bunge (2009), between human and what he called 'automata,' that is the non-human agent:

> "*Although automata can store 'theories', as well as clear-cut instructions to use them, they lack two abilities: (1) they have no judgment or flair to apply them, i.e. to choose the more promising theories or to make additional simplifying assumptions, (2) they can't invent new theories to cope with new situations, unpredicted by the designer and to which the stored theories are relevant*." (p. 160)

The implication of this definition of scope can be seen in the section that follows, where we derive the methods to support evaluation in DSR and describe them using socio-technical vocabularies, and identify the justificatory knowledge that is drawn from both social and technical foundation.

2.3 Evaluation of Information Systems in Healthcare Context

Later on in this paper we use an example of an IS in healthcare context to demonstrate our idea of built-in evaluation support. For this reason, we deem it important to begin the application of each topic or concept into the healthcare context early in this paper. In health-related fields, various information systems have been implemented in order to support particular processes. Nonetheless, the evaluation mechanisms mostly place emphasis on summative evaluation; for instance the Total Evaluation and Acceptance Methodology (TEAM) for IS in biomedicine (Grant, Plante, & Leblanc, 2002), and guidelines to tackle possible challenges in evaluation of health care IT (Ammenwerth, Gräber, Herrmann, Bürkle, & König, 2003).

Shaw (2002) argued that "*information technology is not a drug and should not be evaluated as such* (Heathfield, Pitty, & Hanka, 1998). *Secondly, there is very little evidence to support the view that health care information technology will, in itself, improve patient care*" (p. 210). Based on his review of the practice of health care IS evaluation, Shaw (2002) then put forward the Clinical, Human and Organizational, Educational, Administrative, Technical, and Social (CHEATS) framework for a holistic and generic evaluation of IS in health care sector. The available framework for evaluation of information systems rely mostly on post-implementation evaluation with external tester. Given this tendency, we believe it is fruitful and even necessary to explore other avenues to evaluate IS artifact in general and health care IS in particular that promise better efficiency and effectiveness.

3 Towards a Framework for Software-Embedded Evaluation Support

3.1 Purpose and Objectives of SEES

The purpose of design principles for SEES is to help design science researchers to systematically reflect about features to collect and analyze data about the software artifact, its qualities in practical use, and its implications. We argue that SEES has the potential to radically strengthen evaluation in DSR and explicate the objectives of SEES in terms of four propositions as follows:

- SEES – due to built-in data collection features that primarily require an initial investment – requires fewer resources than non-automated evaluation methods, i.e. criteria-based, goal-based or open-ended evaluation

- SEES – due to the facilitation of a direct line of communication to people – is equally effective as or more effective than non-automated evaluation methods, i.e. criteria-based, goal-based or open-ended evaluation

- SEES – due to the continuous built-in data collection – reduces time spent on evaluation than non-automated evaluation methods, i.e. criteria-based, goal-based or open-ended evaluation

- SEES – through the potential to collect massive amounts of log data and to collect user impressions *in situ* – results in less subjectivity and therefore higher reliability than non-automated evaluation methods, i.e. criteria-based, goal-based or open-ended evaluation

3.2 The Core Typology of SEES

Essentially, our proposition about design for software-embedded evaluation support concerns two core activities in evaluation: data collection and data analysis.

The first issue to be addressed is what data to collect. Following Cronholm & Goldkuhl (2003), we distinguish between three types of evaluation: (i) Criteria-based, (ii) goal-based and (iii) goal-free (open-ended). Thus, we suggest i – iii as three classes of data collection for evaluation purposes. The choice of which one(s) to use is context-dependent. In addition, there is the distinction between types (a) fully automated data collection conducted by the system and (b) self-reported data from users or external testers (Ågerfalk & Sjöström, 2007). Type (a) data collection corresponds to logging mechanisms of software use. Type (b) data collection refers to software features that permit and promote users to provide feedback to developers and DSR researchers. Combining (a,b) and (i, ii, iii) produces a data collection method matrix as shown in Table 1. Note that in our discussion we put emphasis on the evaluation of socio-technical artifacts, even though SESS may to some extent also apply to the evaluation of pure technical artifacts. Observe the mélange of social or business goals (e.g., employee's satisfaction and financial indicators) and technical goals (e.g., automatic forwarding of technical failure messages with time stamp) that reflect the sociotechnical nature of information systems.

Table 1: Software-enabled data collection strategies for ex-post naturalistic evaluation

	Auto-collected data	**Self-reported data**
Criteria-based evaluation	Automated collection of data to support evaluation based on pre-defined generic critera.	Criteria-based questionnaires filled out by users, artifact-in-use observation
Goal-based evaluation	Automated collection of data to support evaluation based on critera derived from the business context, e.g. number of logins, returning clients, sales, performance times, et cetera	Goal-based instruments filled out by users, financial indicators, other quantitative indicators
Goal-free evaluation	Adoption of generic logging framework to enable rich retrospective analysis of business action conducted through the software.	Open-ended questions to users asked via the software

#1 Criteria-Based SEES

Description:

This type of evaluation requires a set of pre-defined criteria that do not necessarily reflect the goal of designing the system. These criteria may be related to particular features or material properties of the systems in particular, or certain generic quality criteria for the functionality of artifact in general (Cronholm & Goldkuhl, 2003). The generic quality criteria may include utility/effectiveness, efficiency, efficacy, and ethicality of the designed system (Venable et al., 2012). These generic criteria cover chiefly the evaluation of what Cronholm and Golkuhl termed "*IT-systems as such*" (Cronholm & Goldkuhl, 2003).

<u>Self-reported data:</u>
Artifact evaluation based on pre-defined criteria requires users to fill out a questionnaire (that could also free-text answers as well as questions with options) with criteria-related domains or items. Another possibility is through an observation of how users work with the IS artifact or how they make use of the artifact – what Cronholm and Goldkuhl termed "*IT-systems in use*" (Cronholm & Goldkuhl, 2003). The behavioral checklist contains items related to the pre-defined criteria and is filled out by the ones who are in charge for the evaluation.

<u>Auto-collected data:</u>
We propose a built-in or automated manner of this kind of evaluation through a software-enabled automatic collection of usage data related to the pre-defined criteria. This way, the collected data is less exposed to the subjectivity of users' self-report and can be completed in a more efficient manner.

#2 Goal-Based SEES

<u>Description:</u>
Goal-based evaluation is conducted with reference to a set of pre-defined goals of designing the system (Cronholm & Goldkuhl, 2003). These pre-defined goals may be related to business or organizational goals that represent financial and social objectives. While financial objectives are usually clear-cut and well defined, social objectives need further definition. It can be defined as employees' satisfaction with their work procedures or even in a much broader term, such as public perception of company image – any of which needs to be further operationalized into measurable variables.

<u>Self-reported data:</u>
Similar to the non-automated criteria-based evaluation, self-reported goal-based evaluation rely on users' and/or external testers' account on the extent to which the goals of designing the system have been fulfilled. When involving quantitative indicators that can be generated post-implementation, such reliance is even stronger. In addition to quantitative indicators, qualitative data (e.g. customer opinions and goal-based questions to staff and customers) may be self-reported by people using the system.

<u>Auto-collected data:</u>
What we propose is, however, an automatic or built-in mechanism of collecting data that reflect the fulfillment of certain goals. In order to allow for automatic data collection, the goals can be operationalized as internal organization variables as number of numbers of login, the amount and frequency of returning clients, performance speed, and sales level and amount or projected into external variables such as numbers of published news articles and median of rating on review websites.

#3 Goal-Free SEES

Description:
Goal-free evaluation is defined as "*gathering data on a broad array of actual effects and evaluating the importance of these effects in meeting demonstrated needs.*" (Cronholm & Goldkuhl, 2003, p. 66). Without any pre-defined goal or criterion, goal-free evaluation has the potential to enable a broader understanding of the function and effects of IS artifact – even to the extent that unintended positive or negative effects can be discovered. We will delve deeper on this notion of unintended effects in the next section.

Self-reported data:
Goal-free evaluation that relies on users' self-report or external testers' report requires these parties to keep an open mind in order to take into account various possibilities of the effects resulting from both IT-systems as such and IT-systems in use. It poses a challenge to ensure that relevant data is collected that can also account for unforeseeable or unintended effects. Goal-free self-reported data would also include qualitative data from (e.g.) a user-group discussion forum where they can speak about the software in general.

Auto-collected data:
By automating the data collection, the availability of possibly relevant data can be ensured and the goal of data analysis – or evaluation – can be decided as needed. Therefore, we propose adopting generic logging framework to enable rich retrospective analysis of business action conducted through the software.

3.3 Easily Changeable Instantiation of SEES

The SEES framework is intended to support the design of artifacts with built-in evaluation features. Since the features may be implemented in many ways, we propose neither detailed implementations nor generic tools to support data collection. Instead, we point out the features that may be valuable as well as other points to guide designers to reflect systematically about how and if such features should be implemented in their context of design and research.

Since most IS artifacts require input from users, we can assume that it undergoes continuous changes – hence, the mutating feature. With automated data collection and automated evaluation, it can be challenging to have a clear-cut representation of what the artifact looks like at the time of certain evaluation. Therefore, it is important to have a clear record of the form of the artifact at the time of the evaluation. Technically speaking, it involves building a repository of – a sort of time machine from which information can be retrieved about the status of artifact, of automated data collection, and of automated evaluation at a particular time. Essentially, any piece of evaluation data needs to be time stamped - in order to interpret a particular dataset, we need to know what the artifact looked at the time of data collection.

Finally regarding the possible instantiation of SEES, we propose the following examples of features for automatic data collection:

- Log user behavior to facilitate retrospective analysis of use.
- Log technical problems, and report them back to the designers/developers.
- Facilitate an open feedback channel for end-users to designers/developers.
- Facilitate a criteria-based feedback channel for end-users to designers/developers
- Facilitate a goal-based feedback channel for end-users to designers/developers.

3.4 Rationale for Software-Embedded Evaluation Support (SEES)

The idea of integrating evaluation mechanism into the artifact itself is not new. Decades ago, the concept of built-in evaluation has been introduced in the field of education (Bhola, 1984) (Dave, 1980) and in the so-called Design for Testability (DFT) and Built-In Self-Test (BIST) for industrial and electronic artifacts (Jervan, Peng, Ubar, & Kruus, 2002; Nagle, Roy, Hawkins, McNamer, & Fritzemeier, 1989), to name a few. The rationale behind BIST is related to, among others, the effort to reduce dependability on external tester as well as to increase efficiency, speed, and the possibility for hierarchical testing through an integration of test infrastructure onto the artifact (Agrawal, Kime, & Saluja, 1993).

In the field of Human-Computer Interaction, the idea of built-in evaluation tools is not new either. One example is the integration of built-in evaluation for each iteration in their prototyping of “CLASP” – a digital artifact that supports adults with Autistic Spectrum Disorder (Simm, Ferrario, Gradinar, & Whittle, 2014). Another example that is familiar to every personal computer use is the power-on self-test (POST) that is integrated in almost all operating systems. POST is a routine that is executed as soon as the device is turned on in order to detect any error in the system. Even in large-scale online systems integrated evaluation tools are not unusual, as reflected in the availability of various rating features, open-ended comment boxes, automatic error reporting, and many other features.

The examples above are just a small fraction of the available instantiations of built-in evaluation. It is not our intention to provide a comprehensive review of ideas similar to SEES in other fields, but rather to illustrate the broadly used logic of self-testing artifact that we can adopt in doing DSR in the field of IS. In addition to that, we present an overview of underlying consideration for each type of SEES in Table 2 and continue with a brief discussion afterwards.

Table 2: Underlying consideration for each type of SEES

SEES Type	**Underlying consideration**
Criteria-based evaluation	Design theory, design principles, kernel theory/ propositions
Goal-based evaluation	Pre-defined business goals, intended uses/ affordances of system
Goal-free evaluation	Randomly occurring events, unforeseeable errors, unintended uses/ affordances of system

The first question that comes to mind when reading about built-in *criteria-based* evaluation would be, where do these criteria come from and in how far do they differ from those in non-automatic evaluation? Even though the criteria can simply be chosen or decided upon by designers prior to the evaluation or, in the best case, when first designing the artifact, the criteria choice can still have their grounding in the previous works. This kind of grounding may relate to design principles based on similar design projects or even further to kernel theory or propositions in behavioral or engineering disciplines.

The same question would also come to mind when discussing built-in *goal-based* evaluation: where do these goals come from? Automatic or not, the goals are related to the pre-defined business or organizational goals when deciding to implement a new IS artifact or to improve the existing ones. Another interesting aspect of the goals used in evaluation is the designerly aspect itself – what the designers intend the artifact to afford users or how they foresee the users using the artifact. Clearly, as in any DSR endeavor, there is a need for design science researchers or designers to reflect about upcoming evaluations during research planning. The choice of criteria and/or goals to guide evaluation indeed impacts the design of built-in evaluation in the artifact – either by designing instruments for self-reporting or by adapting logging functionality to be able to provide measurements of goals and criteria.

In contrast to the other two evaluation categories, *goal-free* evaluation has neither predefined criteria nor predefined goals. Rationale behind this type of evaluation is the awareness that there are always unforeseeable errors and unanticipated uses of the designed artifact due to an emerging social practice or random events. Formalized evaluation based on criteria and business goals may constrain our attention from interesting aspects of the artifact and its use, e.g. unanticipated side effects and spontaneous reactions from users. By conducting goal-free evaluation, we may thus enhance our understanding of the artifact and its meaning to its stakeholders.

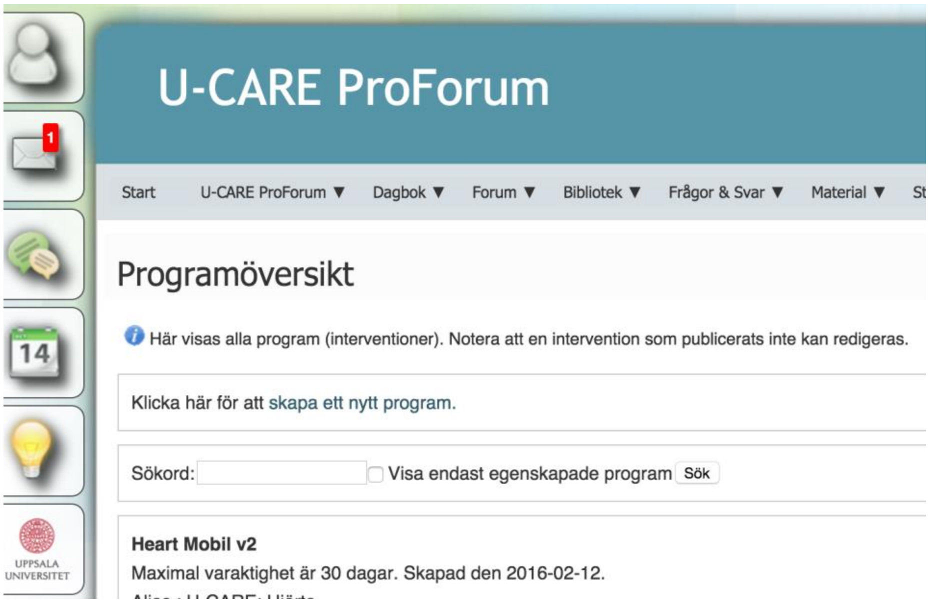
4 A SEES Example: The Case of U-CARE

4.1 The Purpose of U-CARE

In this section, we provide a brief look at built-in evaluation features from U-CARE, a DSR project in the Swedish health care sector. Within the project, a web-based software to provide online cognitive behavioral therapy was developed. The overarching goal of research was to conduct randomized controlled trials (RCTs) to determine treatment efficacy and health economic aspects of various psychosocial support protocols for individuals who suffer from somatic disease. For example, one trial was designed to study the effects on treatment and depression of online psychosocial care for patients who suffered a myocardial infarct. The software system has evolved into a medium-sized software product, consisting of three subsystems, ~40,000 lines of code and ~100 database tables. Nine active research groups currently use the software to deliver (and research the delivery of) online psychological support. So far, more than 500 patients have participated in studies using the software.

4.2 Instantiation of SEES in U-CARE

The design process, largely conducted following Scrum, was based on collaboration between various stakeholders, including patient representatives, psychologists, medical doctors, nurses, economists, software developers and design science researchers. At an early stage in the design process it turned out to be problematic to manage the continual feedback from stakeholders testing the software on the beta server. In order to improve communication, a feature was built in the web portal to allow any user of the software to provide direct feedback about their user experience. Similar features exist in bug-reporting software, and have been used on commercial web sites collect feedback about web site design and service quality from customers.



Programöversikt

Här visas alla program (interventioner). Notera att en intervention som publicerats inte kan redigeras.

Klicka här för att skapa ett nytt program.

Sökord: ☐ Visa endast egenskapade program

Heart Mobil v2

Maximal varaktighet är 30 dagar. Skapad den 2016-02-12.

Figure 2: The ‘light-bulb’ feature to promote stakeholder feedback

Figure 2 shows a screen shot from an arbitrary page on the web site. A click on the light bulb icon (signifying ‘ideas’) opens a dialogue window, in which a free text comment can be written. The light bulb is available on every page on the web site. The comment is stored in an idea backlog, along with a screenshot of the current page, browser information, and some other information derived from the logged in user’s use context. The user can also select a category for the comment. A rephrasing of usability criteria, a simple ‘I found a bug’ category, and a pre-selected category named ‘I’ve got this great idea’ comprises the categories. The basic design idea is as easy as possible to provide feedback on the design of the software, and the feedback should be easily interpretable for both software developers and DSR researchers. The reported ideas were factored into the product backlog in the development process. All stakeholders also had access to the product backlog, where they could discuss ideas, and see the status of development work addressing their idea. The rationale for the design was to provide transparency, to motivate people to submit new ideas continually.

4.3 The ‘Light Bulb’ in SEES Typology

The ‘light bulb’ feature was also integrated into production version of the system for all staff users. It was decided that it should not be available for patients, due to ethical concerns and that it might take their focus away from the treatment protocol. The feature was extensively used, rendering ~1000 comments from users since its introduction in early 2013. In terms of SEES, we characterize the ‘light-bulb’ as a built-in feature for open-ended self-reported data (users only categorized their ideas on a few occasions). The comments represent a variety of ideas, including but not limited to:

- New ideas about how the software should support interaction between psychologists and patients (written by psychologists)
- New ideas about how to design a ‘helicopter view’ for researchers to monitor ongoing RCT activity (written by researchers)
- Suggestions on how to revise existing features, e.g. usability and user experience related design ideas (written by various stakeholders)
- Bug reports (written by various stakeholders)
- Technical issues and software refactoring ideas (from developers)

The characteristics of the ideas thus concern both social aspects (interaction between stakeholders in the practice) and purely technical aspects (internal software design issues). The idea backlog as such is a comprehensive repository with various stakeholder impressions of qualities of the artifact at hand. Clearly, such a repository is a rich source for artifact evaluation in DSR research. In addition to the light bulb feature, there was also automated data collection functionality built in to the artifact, which will not be elaborated here due to space limitations. Nevertheless, it is important to note her that the other functionalities represent also goal-based SEES as well as criteria-based SEES.

5 Concluding Discussion

We began this paper by asking two questions: (1) *How can information systems (IS) evaluation be supported?* and specifically (2) *How can evaluation support be built into the artifact?* Addressing the first question, we provided a brief overview of evaluation in information systems in general and in the DSR stream in particular. In the overview we also touched upon the distinction between formative and summative evaluation and what it means for artifact to be self-evaluating in each of these evaluation stages. Additionally, we contextualized the examples of self-evaluating artifacts and discussed those related to healthcare IS. It was done in order to prepare the ground for further discussion.

Addressing the second question, we developed SEES framework with emphasis on the naturalistic ex-post evaluation (Pries-Heje et al., 2008; Venable et al., 2012), which focuses on evaluation together with real users, real problems, and real systems in real environments. DS researchers may use SEES framework to focus on the actual effects an artifact has in its use context, without losing direct interaction with the end-users of the artifact. Hence, we position our framework within the practice of DSR and information systems design in general (Iivari, 2015), where features for evaluation can be used as built-in features of an artifact (e.g. through automated data collection) at different stages of a DSR-cycle (e.g. design and development stage). Furthermore, SEES is primarily intended to be implemented for socio-technical artifacts, which involve real human actors (e.g. stakeholders) and not abstractions and/or representations of real actors (e.g. personas). However, we also distribute SEES through three classes of data collection, which emphasize strategies for ex-post naturalistic evaluation (shown in Table 1) (Ågerfalk & Sjöström, 2007; Cronholm & Goldkuhl, 2003). Thus, SEES aims to offer a rich feature for DSR evaluation, which can not only serve and support the end-users while making use of the artifact, but also generate input for DSR-researchers throughout their cycles of DSR. Finally, we believe that current frameworks and methods for DSR-evaluation (Peffers et al., 2012; Pries-Heje et al., 2008; Venable et al., 2012, 2014) could benefit from adopting the general idea of SEES, which emphasize a notion of continuously improving artifact quality and utility. Doing so, SEES may enhance the prescriptive nature of designing sufficient and efficient artifacts in DSR (Hevner et al., 2004), by offering in situ features that captures and distributes substantial data for built-in evaluation (shown in Fig 3).

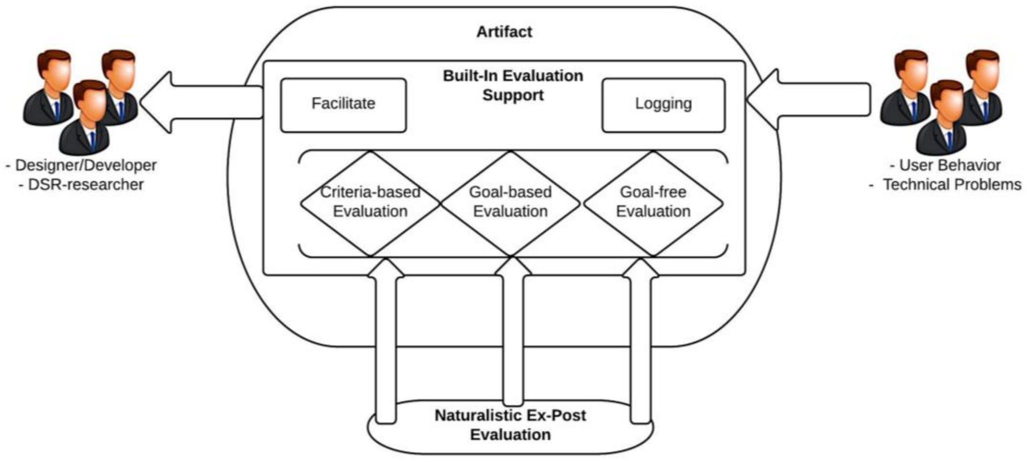


Figure 3: A Conceptual View of SEES

Figure 3 depicts the notion of incorporating three classes of data collection that is provided through end-user interaction, which gets logged, processed, maintained and facilitated for designers/developers and DSR-researchers. The backbone of SEES relies on the idea of naturalistic ex-post evaluation. Thus, the testable hypotheses of SEES address features for automatic data collection, which emphasizes the aspects of logging and facilitating data. However, there may be some issues regarding data quality in the process of automatic data collection. For instance, the collected data may not always be relevant for further analysis. Hence, the question arises: how can the features for logging and facilitating determine the relevance of data for further processing? In other words, if data gets collected automatically, then there may be a need for certifying the relevance of the data through a certain kind of criteria. One idea would be to use the three classes for automatic data collection (e.g. criteria-based evaluation) to define a set of pre-defined rules, which act as filters for reducing noise data (e.g. irrelevant data) throughout the process of data collection.

Another issue delving the notion of automatic data collection emphasizes the idea of incorporating adaptability and flexibility into the built-in evaluation features. An adaptability and flexibility could provide a sense of freedom for both designers/developers, and end-users, in terms of modifying the criteria for data collection filter. For the designers/developers, an adaptability of features would go in line with the idea of design for testability (Dave, 1980; Jervan et al., 2002; Nagle et al., 1989), where features are evaluated through actual use. For the end-users, a flexibility of features would mean that end-users have the freedom to not only provide feedback on actual flaws of the artifact (e.g. software bugs), but to also provide feedback on the actual built-in evaluation features. For instance, it wouldn't be impossible to adopt underlying principles from general rating features and/or chat-mechanisms that establishes a continuity of collecting and facilitating a chain of relevant data. However, such cases of feature adaptability and flexibility could generate a great amount of data, which in turn need to be processed (logged and facilitated). Hence, incorporating adaptable and flexible automatic data collection features is not only relevant for practitioners and end-users, but also to DSR in terms of implementing formative evaluation features (Sein et al., 2011).

To sum up the discussion, we restate our expected contributions as follows:

1. Bringing forward the idea of self-evaluating artifact in the DSR community in IS.
2. Assembling a high-level typology of built-in evaluation support that covers the common methods of data collection for the purpose of evaluation.
3. Providing some preliminary guidelines for design science researchers or IS designers who are interested in embedding evaluation support at either formative or summative evaluation stage.

References

Ågerfalk, P. J., & Sjöström, J. (2007). *Sowing the seeds of self: a socio-pragmatic penetration of the web artefact.* Paper presented at the Proceedings of the 2nd international conference on Pragmatic web.

Alter, S. (2015). The concept of 'IT artifact'has outlived its usefulness and should be retired now. *Information Systems Journal, 25*(1), 47-60.

Ammenwerth, E., Gräber, S., Herrmann, G., Bürkle, T., & König, J. (2003). Evaluation of health information systems—problems and challenges. *International journal of medical informatics, 71*(2), 125-135.

Bunge, M. (2009). *Philosophy of Science: From Explanation to Justification* (Vol. 2). New Brunswick, NJ: Transaction Publishers.

Cronholm, S., & Goldkuhl, G. (2003). Strategies for information systems evaluation-six generic types. *Electronic Journal of Information Systems Evaluation, 6*(2), 65-74.

Dave, R. H. (1980). A built-in system of evaluation for reform projects and programmes in education. *International review of Education, 26*(4), 475-482.

Grant, A., Plante, I., & Leblanc, F. (2002). The TEAM methodology for the evaluation of information systems in biomedicine. *Computers in Biology and Medicine, 32*(3), 195-207.

Heathfield, H., Pitty, D., & Hanka, R. (1998). Evaluating information technology in health care: barriers and challenges. *bmj, 316*(7149), 1959 - 1961.

Hevner, A. R., March, S. T., & Park, J. (2004). Design Science in Information Systems Research. *MIS Quarterly, 28*(1), 75–105.

Iivari, J. (2015). Distinguishing and contrasting two strategies for design science research. *European Journal of Information Systems, 24*(1), 107-115.

Jervan, G., Peng, Z., Ubar, R., & Kruus, H. (2002). *A hybrid BIST architecture and its optimization for SoC testing.* Paper presented at the Quality Electronic Design, 2002. Proceedings. International Symposium on.

Lee, A. S., Thomas, M., & Baskerville, R. L. (2015). Going back to basics in design science: from the information technology artifact to the information systems artifact. *Information Systems Journal, 25*(1), 5-21.

Nagle, H. T., Roy, S. C., Hawkins, C. F., McNamer, M. G., & Fritzemeier, R. R. (1989). Design for testability and built-in self test: a review. *Industrial Electronics, IEEE Transactions on, 36*(2), 129-140.

Peffers, K., Rothenberger, M., Tuunanen, T., & Vaezi, R. (2012). Design science research evaluation *Design science research in information systems. Advances in theory and practice* (pp. 398-410): Springer.

Pries-Heje, J., Baskerville, R., & Venable, J. R. (2008). Strategies for design science research evaluation.

Remenyi, D., & Sherwood-Smith, M. (1999). Maximise information systems value by continuous participative evaluation. *Logistics Information Management, 12*(1/2), 14-31.

Robey, D., Anderson, C., & Raymond, B. (2013). Information Technology, Materiality, and Organizational Change: A Professional Odyssey. *Journal of the Association for Information Systems*, *14*(7), 379–398.

Sein, M. K., Henfridsson, O., Purao, S., Rossi, M., & Lindgren, R. (2011). Action design research. *Management Information Systems Quarterly, 35*(1), 37–56.

Shaw, N. T. (2002). 'CHEATS': a generic information communication technology (ICT) evaluation framework. *Computers in Biology and Medicine, 32*(3), 209-220.

Smithson, S., & Hirschheim, R. (1998). Analysing information systems evaluation: another look at an old problem. *European Journal of Information Systems, 7*(3), 158-174.

Sonnenberg, C., & vom Brocke, J. (2012). Evaluation patterns for design science research artefacts *Practical Aspects of Design Science* (pp. 71-83): Springer.

Stufflebeam, D. L. (2003). The CIPP model for evaluation *International handbook of educational evaluation* (pp. 31-62): Springer.

Venable, J., Pries-Heje, J., & Baskerville, R. (2012). A comprehensive framework for evaluation in design science research *Design Science Research in Information Systems. Advances in Theory and Practice* (pp. 423-438): Springer.

Venable, J., Pries-Heje, J., & Baskerville, R. (2014). FEDS: A Framework for Evaluation in Design Science Research. *European Journal of Information Systems*.