A Design Theory for Built-in Evaluation Support

Jonas Sjöström[1], Leona Chandra Kruse[2], Amir Haj-Bolouri[3]

[1]Uppsala University, Uppsala, Sweden
jonas.sjostrom@im.uu.se

[2]University of Liechtenstein, Vaduz, Liechtenstein
leona.chandra@uni.li

[3]University West, Trollhättan, Sweden
amir.haj-bolouri@hv.se

Abstract: Even though the practice of integrating evaluative features into the artifact has long been applied in commercially available software, it is still underrepresented in IS community. This paper presents a design theory for built-in evaluation support. We seek to incorporate the idea of 'the self-evaluating artifact' into the design science research. We are aware of the challenges in designing built-in evaluation features in socio-technical systems and take this issue into consideration in our framework. While the theory draws from the literature on evaluation in DSR, we also provide an expository instantiation of the theory demonstrating a built-in evaluation support mechanism designed and used in DSR research in the Swedish healthcare sector.

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

1 Introduction

The prevalence of online software has had a major impact not only on society, but also on software development practices and research into information systems (IS) design. 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 [1]. 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. It is also regarded as a core activity in design science research (DSR). Ultimately, DSR researchers need to demonstrate the efficacy and utility of their designed artifacts [1, 2], where artifact quality implicitly signals the value of the abstract knowledge embodied in the artifact [1]. 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 [1, 3-5]. 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 frameworks have not sufficiently addressed the prospect of building in the evaluative features into the artifacts themselves. 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.

However, we need to be aware of the challenges in integrating evaluation mechanisms into IS artifacts given their socio-technical nature [6, 7]. Recent discourse on the notion of IS brings about potential to refresh our way to conceptualize and eventually to evaluate IS artifacts, whether through unpacking an IS artifact into its technical, social, and information components [7] or by retiring the mention of the notion of "IS artifact" and therefore becoming specific and precise about the goal, form, and function of the information systems under scrutiny [8]. Either way the challenges remain as to how we can evaluate social and technical aspects of the artifact and if it is feasible to support evaluation of both aspects by incorporating evaluation mechanisms into the artifact.

In this paper, we disclose an underexplored dimension of DSR evaluation: the integration of evaluation mechanisms into artifact instantiations. We synthesize various elements of the DSR evaluation discourse into an espoused design theory [9] for DSR artifact design with built-in evaluation support. We provide an expository instantiation

of the theory, i.e. practical example of the theory-in-use within a DSR project, and discuss implications for both practice and research.

The paper proceeds as follows. In section 2, we present related research emphasizing evaluation of IS artifacts in DSR. In section 3, we account for our theory of built-in evaluation support. In section 4, we provide an expository instantiation of the theory, followed by a concluding discussion. .

2 An Overview of IS Evaluation

There are a number of different purposes for evaluating IS artifacts in DSR, as well as there is a variety of different methods, strategies and activities for conducting the actual evaluation process. Remenyi et al. [10] 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. We will in the upcoming sub-sections elaborate on these categories by illustrating how they are incorporated into evaluation methods in DSR.

Venable et al [5] 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 [10-12]. 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 the notion of formative vs. summative evaluation, and (2) evaluation paradigm, which incorporates the notion of 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 flawed design for the instantiated artifact. 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. On the contrary, 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 [5]. Fig 1 summarizes and illustrates the constitution of FEDS.

Venable et al's [5] 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 [13] 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' [5] FEDS incorporates a similar notion of evaluation guidance. Furthermore, Venable et

al's [5] FEDS extends previous ideas of Pries-Heje et al [4], Venable et al [1] and Peffers et al [13]. Hence, the FEDS approach seems to be a suitable source of inspiration for a design theory for built-in evaluation support.

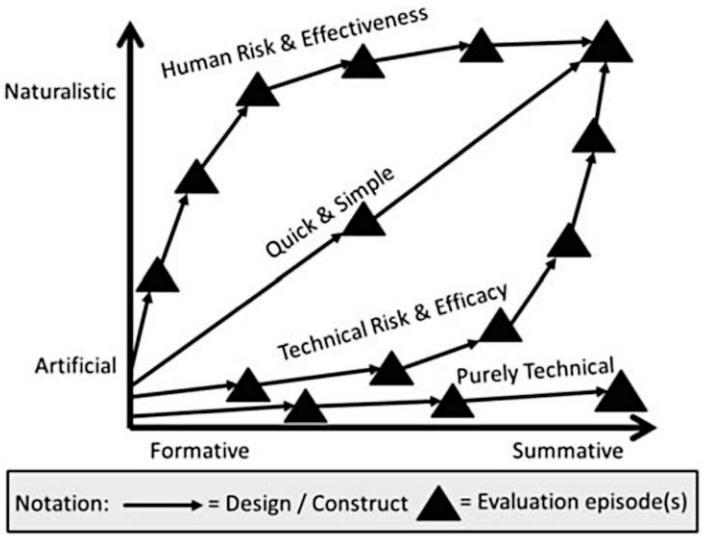

Fig. 1. FEDS (Framework for Evaluation in Design Science) with evaluation strategies [5]

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 [14], and guidelines to tackle possible challenges in evaluation of health care IT [15].

Shaw [16] argued that "*information technology is not a drug and should not be evaluated as such* [17]. *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 [16] 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. Given this tendency to rely on post-implementation evaluation with external tester, 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. At this point, we turn to our design theory for built-in evaluation support (ToBES).

3 A Design Theory for Built-in Evaluation Support

In this section, we present our design theory following the anatomy of a design theory [9]. In doing so, we begin with a description of the purpose and scope of design theory for built-in evaluation support (ToBES), and then continue to relevant principles of form and function, followed by the justificatory knowledge. Next, we will discuss the mutability of ToBES and the principles of implementation for built-in evaluation support. The description of an expository instantiation of ToBES will be depicted in Section 4.

3.1 Purpose and Scope

The purpose of ToBES is to guide design science researchers to build artifacts that support self-evaluation, i.e. that implement features to collect and analyze data about their use.

Let us clarify the scope of our design theory by positioning it in relation to a set of evaluation dimensions derived from the DSR evaluation discourse. First, our design theory deals with naturalistic ex-post evaluation[1, 4], i.e. 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 [1]. Naturalistic ex-post evaluation can be carried using various methods, including (but not limited to) action research, case study, focus group, participant observation, ethnography, phenomenology, and qualitative or quantitative surveys.

We position the design theory within the practice approach to DSR [18], i.e. a focus on the effects an artifact has in in its use context, and the meaning which is ascribed to it by its various stakeholders.

Formative versus summative evaluation.
ToBES proposes 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 [19], or for summative purposes to demonstrate the qualities of an artifact after a period of use.

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*." [1]

Pure technical artifact versus socio-technical artifact

It is also important to define the scope of ToBES in terms of the nature of the IS artifact. ToBES is intended primarily for 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. This so-called automata executes 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 [20], 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 our next sections of ToBES, where we formulate principles of form and function using socio-technical vocabularies and identify the justificatory knowledge that is drawn from both social and technical foundation.

3.2 Principles of form and function

Essentially, our proposition about design for built-in 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 [21], 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 (a) fully automated data collection conducted by the system and (b) self-reported data from users or external testers [22]. Combining (a,b) and (i, ii, iii) produces a data collection strategy matrix as shown in Table 1. Note that in our discussion we put emphasis on the evaluation of socio-technical artifacts, even though ToBES may to some extent also apply to the evaluation of pure technical artifacts.

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

Criteria-based Evaluation

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 [21]. The generic quality criteria may include utility/effectiveness, efficiency, efficacy, and ethicality of the designed system [1]. These generic criteria cover chiefly the evaluation of what Cronholm and Golkühl termed "*IT-systems as such*" [21].

If executed in a non-automated manner, evaluation of IS artifact based on pre-defined criteria requires users to fill out a questionnaire 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 Golkuhl termed "*IT-systems in use*" [21]. 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. 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.

Goal-based Evaluation

Goal-based evaluation is conducted with reference to a set of pre-defined goals of designing the system [21]. 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.

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. 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.

Goal-free Evaluation
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."* [21, 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.

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. 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.

Features for automatic data collection
Finally regarding the form and functions for our design theory, we propose the following 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 the designers/developers
- Facilitate a goal-based feedback channel for end-users to the designers/developers

3.3 Justificatory knowledge

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 [23] [24] and in the so-called Design for Testability (DFT) and Built-In Self-Test (BIST) for industrial and electronic artifacts [25] [26], 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 [27].

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 [28]. 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 ToBES in other fields, but rather to illustrate the broadly used logic of self-testing artifact that we can adopt in doing DSR in IS. As Gregor and Jones [9, p. 328] argued, "*it remains essential to include justificatory knowledge in ISDTs, although this knowledge could be incomplete. The justificatory knowledge provides an explanation of why an artifact is constructed as it is and why it works […]*". We can thus argue that the rationale behind the integration of evaluation support into the system is in itself a justificatory knowledge. In addition to that, we present an overview of further justificatory knowledge for each type of built-in evaluation in Table 2 and continue with a brief discussion afterwards.

Table 2. Justificatory knowledge for each type of built-in evaluation

Built-In Evaluation	Justificatory Knowledge
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 theory or 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 most likely 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 researchers to reflect about upcoming evaluations during research planning. The choice of criteria and/or goals to guide evaluation may impact 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 software 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.

3.4 Artifact mutability

ToBES is a generic theory to support the design of artifacts with built-in evaluation features. Those features may be implemented in many ways; therefore ToBES does not propose a detailed implementation; or a generic tool to support data collection. It merely points out that in principle, such features may be valuable, and that the theory guides 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, and in order to interpret that data we need to know what the artifact looked at that time.

3.5 Testable propositions

According to Gregor & Jones [9], we should provide a set of testable propositions (i.e. hypothesis that 'if we apply this theory given certain circumstances, we should expect certain results'. Taking into account the purpose and scope of ToBES together with its justificatory knowledge as discussed in the previous sections, we can generate the following propositions:

P1. Built-in or automated evaluation methods require fewer resources than non-automated evaluation methods, i.e. criteria-based, goal-based or open-ended evaluation

P2. Built-in or automated evaluation methods are equally effective as or more effective than non-automated evaluation methods, i.e. criteria-based, goal-based or open-ended evaluation

P3. Built-in or automated evaluation methods require less time than non-automated evaluation methods, i.e. criteria-based, goal-based or open-ended evaluation

P4. Built-in or automated evaluation methods results in less subjectivity and therefore higher reliability than non-automated evaluation methods, i.e. criteria-based, goal-based or open-ended evaluation

4 The Design Theory Applied: The Case 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.

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 DSR 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.

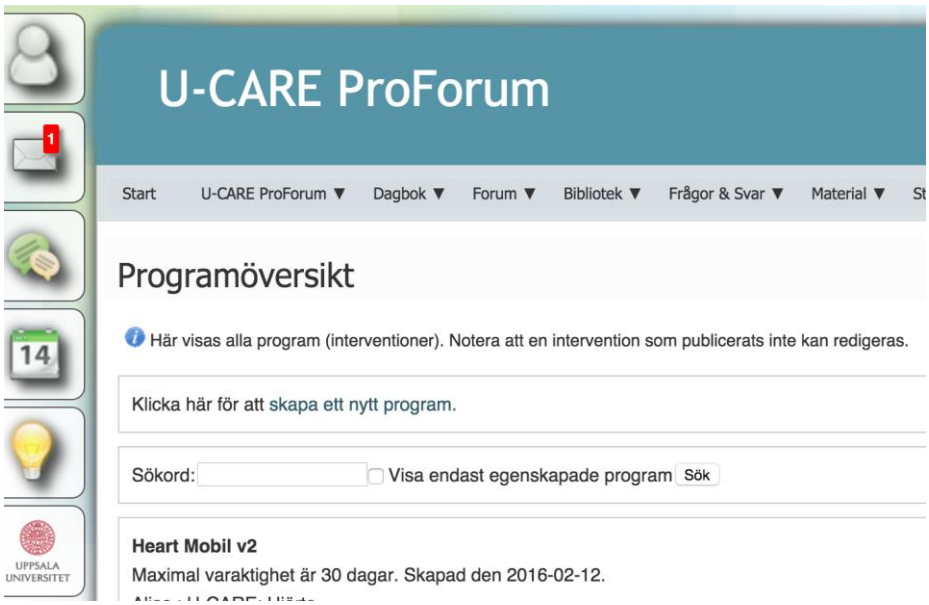

Fig. 2. The 'light-bulb' feature to promote stakeholder feedback

Fig 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. The categories are comprised by a rephrasing of usability criteria, a simple **'**I found a bug**'** category, and a pre-selected category named **'**I've got this great idea**'**. The basic design idea is that it should be as easy as possible to provide feedback on the design of the software, and that that 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.

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 ToBES, 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.

5 Concluding Discussion

In this paper, we have developed ToBES – a design theory for built-in evaluation support. ToBES emphasizes the naturalistic ex-post evaluation [1, 4], which focuses on evaluation together with real users, real problems, and real systems in real environments. DSR-researchers may use our theory to focus on the actual effects an artifact has in its use context, without loosing direct interaction with the end-users of the artifact. Hence, we position our design theory within the practice of DSR [18], 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, ToBES 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 ToBES through three classes of data collection, which emphasize strategies for ex-post naturalistic evaluation (shown in Table 1) [21,22]. Thus, ToBES aims to offer a rich feature for DSR-evaluation, which can serve and support the end-users of an artifact through artifact use, but also generate input for DSR-researchers throughout their cycles of DSR. Finally we believe that current frameworks and methods for DSR-evaluation [1,4,5,13] could benefit from adopting the general idea of ToBES, which emphasize a notion of continuously improving artifact quality and utility. Doing so, ToBES may enhance the prescriptive nature of designing sufficient and efficient artifacts in DSR [2], by offering in situ features that captures and distributes substantial data for built-in evaluation (shown in Fig 3).

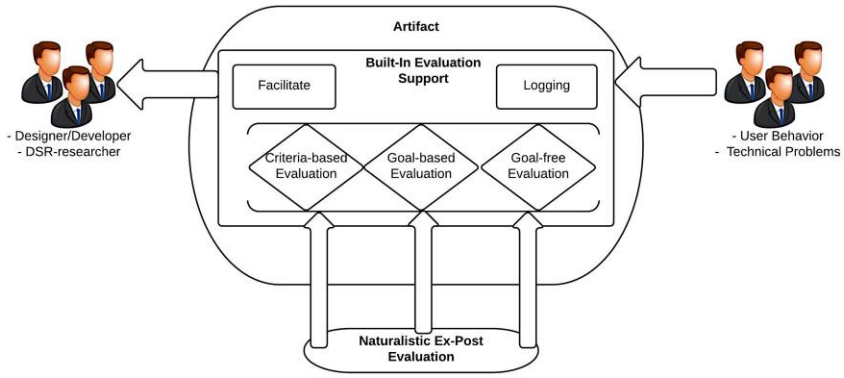

Fig. 3. A Conceptual View of ToBES

Fig 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 ToBES relies on the idea of naturalistic ex-post evaluation. Thus, the testable hypotheses of ToBES 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 [25,26], 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 DSR-research in terms of implementing formative evaluation features [19].

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

2 Hevner, A.R., March, S.T., and Park, J.: 'Design Science in Information Systems Research', MIS Quarterly, 2004, 28, (1), pp. 75–105

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

4 Pries-Heje, J., Baskerville, R., and Venable, J.R.: 'Strategies for design science research evaluation', 2008

5 Venable, J., Pries-Heje, J., and Baskerville, R.: 'FEDS: A Framework for Evaluation in Design Science Research', European Journal of Information Systems, 2014

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

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

8 Alter, S.: ‘The concept of ‘IT artifact’has outlived its usefulness and should be retired now’, Information Systems Journal, 2015, 25, (1), pp. 47-60
9 Gregor, S., and Jones, D.: ‘The anatomy of a design theory’, Journal of the Association for Information Systems, 2007, 8, (5), pp. 312–335
10 Remenyi, D., and Sherwood-Smith, M.: ‘Maximise information systems value by continuous participative evaluation’, Logistics Information Management, 1999, 12, (1/2), pp. 14-31
11 Smithson, S., and Hirschheim, R.: ‘Analysing information systems evaluation: another look at an old problem’, European Journal of Information Systems, 1998, 7, (3), pp. 158-174
12 Stufflebeam, D.L.: ‘The CIPP model for evaluation’: ‘International handbook of educational evaluation’ (Springer, 2003), pp. 31-62
13 Peffers, K., Rothenberger, M., Tuunanen, T., and Vaezi, R.: ‘Design science research evaluation’: ‘Design science research in information systems. Advances in theory and practice’ (Springer, 2012), pp. 398-410
14 Grant, A., Plante, I., and Leblanc, F.: ‘The TEAM methodology for the evaluation of information systems in biomedicine’, Computers in Biology and Medicine, 2002, 32, (3), pp. 195-207
15 Ammenwerth, E., Gräber, S., Herrmann, G., Bürkle, T., and König, J.: ‘Evaluation of health information systems—problems and challenges’, International journal of medical informatics, 2003, 71, (2), pp. 125-135
16 Shaw, N.T.: ‘‘CHEATS’: a generic information communication technology (ICT) evaluation framework’, Computers in biology and medicine, 2002, 32, (3), pp. 209-220
17 Heathfield, H., Pitty, D., and Hanka, R.: ‘Evaluating information technology in health care: barriers and challenges’, bmj, 1998, 316, (7149), pp. 1959
18 Iivari, J.: ‘Distinguishing and contrasting two strategies for design science research’, European Journal of Information Systems, 2015, 24, (1), pp. 107-115
19 Sein, M.K., Henfridsson, O., Purao, S., Rossi, M., and Lindgren, R.: ‘Action design research’, Management Information Systems Quarterly, 2011, 35, (1), pp. 37–56
20 Bunge, M.: ‘Philosophy of Science: From Explanation to Justification’ (Transaction Publishers, 2009. 2009)
21 Cronholm, S., and Goldkuhl, G.: ‘Strategies for information systems evaluation-six generic types’, Electronic Journal of Information Systems Evaluation, 2003, 6, (2), pp. 65-74
22 Ågerfalk, P.J., and Sjöström, J.: ‘Sowing the seeds of self: a socio-pragmatic penetration of the web artefact’, in Editor (Ed.)^(Eds.): ‘Book Sowing the seeds of self: a socio-pragmatic penetration of the web artefact’ (ACM, 2007, edn.), pp. 1-8
23 Bhola, H.: ‘Building a Built-in Evaluation System: A Case in Point’, 1984
24 Dave, R.H.: ‘A built-in system of evaluation for reform projects and programmes in education’, International review of Education, 1980, 26, (4), pp. 475-482

25 Nagle, H.T., Roy, S.C., Hawkins, C.F., McNamer, M.G., and Fritzemeier, R.R.: ‘Design for testability and built-in self test: a review’, Industrial Electronics, IEEE Transactions on, 1989, 36, (2), pp. 129-140
26 Jervan, G., Peng, Z., Ubar, R., and Kruus, H.: ‘A hybrid BIST architecture and its optimization for SoC testing’, in Editor (Ed.)^(Eds.): ‘Book A hybrid BIST architecture and its optimization for SoC testing’ (IEEE, 2002, edn.), pp. 273-279
27 Agrawal, V.D., Kime, C.R., and Saluja, K.K.: ‘A tutorial on built-in self-test. I. Principles’, IEEE Design & Test of Computers, 1993, (1), pp. 73-82
28 Simm, W., Ferrario, M.A., Gradinar, A., and Whittle, J.: ‘Prototyping'clasp': implications for designing digital technology for and with adults with autism’, in Editor (Ed.)^(Eds.): ‘Book Prototyping'clasp': implications for designing digital technology for and with adults with autism’ (ACM, 2014, edn.), pp. 345-354