2011

Guiding Situational Applications from a Structuration Perspective

Ronald Maier
University of Innsbruck, ronald.maier@uibk.ac.at

Ulrich Remus
University of Canterbury, ulrich.remus@canterbury.ac.nz

Guiding Situational Applications from a Structuration Perspective

Ronald Maier
School of Management
University of Innsbruck
Innsbruck, Austria
Email: ronald.maier@uibk.ac.at

Ulrich Remus
ACIS Department
University of Canterbury
Christchurch, New Zealand
Email: ulrich.remus@canterbury.ac.nz

Abstract

Situational applications are a new breed of software assumed to fit to the types of tasks and contextual requirements encountered in dynamic work environments described as weakly structured, highly diverse and fast-changing. The aim of this paper is to discuss the characteristics of situational applications and how organizations can benefit from them with the help of Structuration Theory. Concepts from Structuration Theory allow us to differentiate situational applications from traditionally developed business applications according to the specifics of design and development, the resulting product as well as its deployment and usage. Theoretical implications of this discussion are a much more concise description of situational applications and a host of potential research avenues to further explore research questions on situational applications while managerial implications not only call for creating an organizational and technological infrastructure in support of situational applications, but also for guidance of these grassroots approaches to software development and use.

Keywords

End-user Computing, Structuration Theory, Situational Application, IS Research Agenda

INTRODUCTION

With the growing number of contents and services available on the Internet and on organizations' Intranets, users can now build lightweight applications by arranging them into new information technology (IT)-based services which can be further shared and recombined fitting to current purposes, contexts and environments of prospective users. This emerging breed of software is called situated, situative or situational application (SA), as it focuses on solving distinct problems for a particular social situation, user group or other context at hand (Shirky 2004). SAs are particularly interesting for knowledge workers who need to deal with weakly defined problems, unexpected events and exceptions outside well-structured and -documented business processes. Typically, knowledge workers need instant and ad-hoc solutions for their most pressing business – or knowledge – problems (Schultze 2000), which often corporate IT with their rigid development processes cannot deliver in a timely and efficient manner. Thus, it is not surprising that employees already conduct ad-hoc application development on a large scale, with 19%- 30% of staff having implemented a business function, process, or activity outside of a formal IT development project (Cherbakov et al. 2007a). This share still tends to grow with the advent of more end-user friendly development environments as well as the enormous increase of the number of building blocks, such as web services, feeds, plug- and add-ins and widgets, now available on the Internet and on organizations' Intranets. Another factor boosting adoption of SAs is the renewed interest in supporting knowledge work with collaborative technology backed by the 2.0 hype, thus moving SAs beyond personal use to support collaborative action in collectives of people performing knowledge work such as project teams, work groups or communities-of-practice. The potential positive impact extends from individual benefits to benefits on a collective or even organisational level (DeLone and McLean 1992). Examples are increased productivity due to adapting the work environment to personal skills, and preferences on an individual level as well as improved organisational effectiveness (Lewin and Minton 1986) by continuous improvement of IT-supported collaborative practices and processes that can also be more easily shared and transferred to other social situations as good or even local best practices (O'Dell and Grayson 1998).

SAs are assumed to accommodate change and remain adaptive, however, using SAs also creates some new challenges with respect to quality attributes such as performance, security and availability (Balasubramaniam et al. 2008), similarly to the challenges discussed with respect to cloud computing (Kaufman 2009). There is also no formal budget, applications are in perpetual beta and best practices of adoption are nonexistent, so it is not surprising that corporate IT is often reluctant to fully take on board these new applications and might even view them as “controlled chaos”. Another area of concern is related to control, security, privacy and integrity, such as how to deal with data ownership, opening up enterprise data resources, connection of external and internal services and sharing of these services on a global scale (Cherbakov et al. 2007a).

Whereas current research in this area has focused on design science and technical approaches supporting employees with SA environments, tools, and systems (Hoyer and Fischer 2008), there is still little known about the (social) processes, potential conflicts, and governance of developing and using SAs (Pahlke et al. 2010). The aim of this paper thus is to address the following central research questions: What are the assumed new qualities that designing, building and (re-)using SAs have compared to software applications that are built to order? What are the social processes taking place when SAs are designed, developed and used? How should IT management deal with the managerial implications of SA adoption, i.e. an organization decides to use SAs (Rogers 1995), acceptance, i.e. that users decide voluntarily to use SAs (Dillon and Morris 1996) and most importantly assimilation, i.e. SA-based solutions diffuse into organisational work practices, processes and corresponding daily activities (Chatterjee et al. 2002). The resulting new conceptualisation of SA is intended to take up the theoretical implications and answer the call for extending existing definitions that have been primarily developed from a technical perspective, neglecting social and organizational aspects of SAs (Pahlke et al. 2010).

Consequently, theories are required for studying the processes involved in providing an organisational and technical infrastructure, building blocks and managerial guidance as well as in developing - or more precisely, composing, appropriating and (re-) using SAs. Structuration Theory provides a potentially useful way of understanding work practices in emerging contexts such as those affected by SAs. In particular, Adaptive Structuration Theory (AST) has been proposed as a framework to explore the mutual influences of technology and social processes (DeSanctis and Poole 1994). So far, AST research has aimed at a holistic account of the structuration process, including identification of relevant IT, task, contextual, and organizational structures, description of structuring moves and of the effects of organizational and other contextual forces (Jones and Karsten 2008). AST research has been focused on applications that are traditionally designed, built and operated. However, we think that Structuration Theory, and in particular, AST can also be used as a framework for analysing the assumed changes brought to organizations with SAs. Some ideas to accommodate these changes in organizations towards emergence, innovation and improvisation, in particular Orlikowski’s (2000) ideas to adopt a practice lens, have already been brought into Structuration Theory applied to study effects of IS use. In the following, we will first differentiate SAs from traditionally developed business applications in order to elaborate on the specific characteristics before we employ Structuration Theory in order to analyse the social processes involved in handling SAs in organisations.

CHARACTERIZING SITUATIONAL APPLICATIONS

Situated software, end-user computing, grassroots computing, (lightweight) composite applications, opportunistic software, situational software are all terms for the same type of software emphasizing the idea that software is designed in and for a particular social situation or context (Shirky 2004), called SA here, in order to solve a pressing business need. This focus on time, i.e. immediate solutions for current business problems as well as end-users, i.e. users acting as designers and developers, differentiates SAs from business applications which are professionally developed with a clear division of labour between multiple roles, such as requirements engineer, designer, developer, tester and administrator as well as sophisticated development and change processes. End-users with little or no programming background can be able to assemble SAs from existing ones or from building blocks, significantly reducing the traditional development process of edit, compile, test and run (Cherbakov et al. 2007a) and thus blurring the boundaries between design time, build time and run time. Thus, SAs can be seen as specialization of end-user computing (McBride and Wood-Harper 2002) for dynamic application areas in which tasks are highly situated, change is fast-paced and development relies not only on a specific technical infrastructure, but on building blocks readily available to be combined into situation-specific solutions. In contrast to traditional software, situated software usually is not constructed by a team of software developers. Instead, it is created by users desiring specific functionality that traditionally developed software does not provide. Typically, it is personalized, localized software that has evolved organically and has been created by the community that uses it (Balasubramaniam et al. 2008). One current example of SAs are enterprise mashups (Crupi and Warner 2008; Hoyer and Fischer 2008; Pahlke et al. 2010).

In comparison to developing traditional software, such as individual or packaged software, SAs need a different development approach. In this approach, those who best understand the business problem at hand develop rapid solutions without the overhead and formality of traditional IT methods and procedures, thus shortening the edit-compile-test-run development life cycle. The following table compares the traditional way of implementing software, such as individual or packaged software, with the way of building SAs (Cherbakov et al. 2007b).

Table 1: Differences between traditional and situational applications (extended from Cherbakov et al., 2007a)

	Traditionally developed business applications	**Situational applications**

Design and Development		
Problem addressed	Clearly defined business need that requires redesign of business processes and practices for a large number of similar cases	Weakly defined, current, pressing business need requiring an immediate solution, often for a particular case
Time-to-value	Months or years	Days or weeks
Development phases	Variety of established procedures and methods for software design including sequential, e.g., waterfall model, and iterative, e.g., agile model, development frameworks, Well defined, following an agreed schedule (although with frequent schedule overruns)	No defined phases or milestones, focus on a good-enough solution
Functional requirements	Limited number of (representatives of) users defines intended use; IT freezes requirements to move to development; scope creep often caused by changing business needs.	Defined by users themselves; as requirements change, SAs usually change to accommodate business changes; SAs encourage unintended uses
Non-functional requirements	Resources allocated to address concerns for performance, availability, security & other non-functional requirements, often results in robust solutions	Little or no focus on scalability, maintainability, availability, security etc.
Testing	By IT department or provider with some user involvement	By users through actual use
Funding	Often coincides with periodic IT planning; requires approved budgets	No formal budget; often developed and run without notice by corporate IT
Primary stakeholders	Executives and managers in the lines of business, corporate IT	Individual developer or self-organizing small team; user community, backed by infrastructure provided by corporate IT or external IT service providers
Targeted users	Large, generic group of people acting in clearly defined roles	Known individuals (often the application builder) / small team; attracts unintended users
Governance	Centralized, formal	Grassroots, community based
Evolution	Top-down, centrally driven, depends on funding	Organic, based on user feedback and participation
Product		
Maturity	Mature, proven technologies	Frequent use of less mature and emerging technologies
User interface	No special focus	Emphasis on Rich Internet Applications (RIA) use of technologies, such as Ajax, Adobe Flex, Adobe Integrated Runtime, and XAML
Tools	Stand-alone integrated development environment	Browser-based
Integration technology	Often proprietary application programming interfaces (API)	Light-weight integration technologies, such as REST, pipes, add-ins, plug-ins

As such, SAs can be seen from two perspectives, the process perspective, focusing on activities within the life cycle of a SA, and the product perspective, seeing SA as an artifact. Both perspectives are strongly related as implementation stages can be defined as a set of activities (process) together with its outcomes (product) (Cooper and Zmud 1990). In particular the process perspective with its obvious links to social processes calls for further research as there is significant shortage of in-depth investigations into the impact of end-user computing in general and SAs in particular on an organizational as well as an individual level (Pahlke et al. 2010). Summing up, SAs combine or adapt pre-existing content, data or application functionality to fit an immediate business need. SAs are typically developed and deployed by autonomous end-users in a grassroots approach with no clear demarcation between design, development and usage. A SA thus can be defined as an immediate software-based solution for a current business problem composed from building blocks by an end-user using a flexible integration infrastructure. Please note that this contrasting of SAs and traditionally developed business applications emphasizes two extreme positions. In practice, the boundaries are sometimes blurry.

Example: Let us assume a business process for allocating field staff, undertaking maintenance work in a time-critical mission. One of the tasks in this process is to find field workers with the right expertise and allocate them to ad-hoc teams. Following the old process, the allocation manager had to manually search the internal yellow page system by area of expertise, look up personnel numbers and then check their availability via the collaboration server. One of the problems was to manually coordinate field workers according to their actual location, and optimize the time to get to the mission's site. Being Internet savvy, she thought of creating a SA using mashup tools to combine the internal search and availability check with a map which shows the current location of each field worker. She would then be able to use this information to flexibly assemble teams according to their geographic proximity. The first step was to talk to the IT department in order to open up the

Web service for allowing internal searches through the yellow page system. This Web service returns key personnel data, such as employee no, name or address for experts matching the input parameter of an area of expertise. The service retrieves data from the employee search service which itself interfaces to the enterprise resource planning system's human resource management module. With the help of a colleague who already developed some consumer mashups, she combined both, the internal yellow pages search service and the check availability service together with the external Google Map Web API into a new enterprise mashup. As the mashup was incorporated into the process, the allocation manager just needs to enter areas of expertise and she can easily browse through a Google map showing actual locations of corresponding field workers. After having successfully used this mashup for a couple of weeks, a lot of interest emerged to use the SA as a service and recombine it into other, more complex services. Next to some suggestions for improvement, e.g., to show the mission's site, one idea was to open it up for wider internal use, but also expose it to external business partners. The IT department also realized that opening the yellow pages' search service triggered a large number of similar requests to have access to existing or create new useful services. However, numerous redundant or slightly different services were created in parallel, so that a choice has to be made with regard to quality criteria, such as correctness and availability.

A STRUCTURATION-THEORETIC PERSPECTIVE ON SA

Structuration Theory is without doubt one of the most popular theories used in information systems (IS) research. Its focus on social aspects of IS, in particular its focus on structure and on the processes by which structures are used and modified over time appeals to IS research (Poole and DeSanctis 2004). It is out of the scope of this paper to provide a full review of Structuration Theory as there are simply too many contexts that ST has been applied to and as a number of articles have already critically reviewed its application in the field of IS (Jones and Karsten 2008; Poole and DeSanctis 2004). Instead, we focus on the key concepts Structuration Theory could provide for investigating social aspects of SAs. We limit our selection to concepts derived from the interpretation of Structuration Theory, based on Jones and Karsten (2009) and DeSanctis and Poole's (1994) Adaptive Structuration Theory (AST). It is thus important to note that there are several differences, such as Poole's rejection of Giddens' view of structures as memory traces and their stance towards the commensurability of positivist and interpretive research. (see the debate in Jones and Karsten 2008; Jones and Karsten 2009; Poole 2009). AST's main purpose has been to actively guide empirical research which was still lacking in Giddens' theory and other social theories (Kontopolous 1993; Sewell 1992). Thus, AST draws not only from Giddens' work, such as Strauss's (1975) negotiated order perspective, Barthes' (1974) structuralism, and work in conversational and discourse analysis. AST is organized around a set of concepts that are meant to be applied generically in the study of IS implementation and use, as well as in other contexts. The concepts should not be seen as variables, but rather describe the process by which structures associated with an IS are produced and reproduced in social interaction. In doing so, Poole and DeSanctis argue for a social constructionist and reflexive view of these concepts (DeSanctis and Poole 1994; Poole and DeSanctis 1990; Poole and DeSanctis 2004). In the following, key concepts from both Structuration Theory and AST are briefly reviewed which are deemed important for studying IS relying on situational rather than traditional applications. These concepts are discussed with the help of the case example on SAs for allocating field staff.

Structure is key to Structuration Theory. Structures include resources (command over people or material goods) and rules (recipes for action) which operate to provide a social system within its three dimensions power (structures of domination), norms/routines (structures of legitimation), and meaning (structures of signification) (Poole and DeSanctis 2004). Corresponding dimensions of interaction, described as communication, power, and sanctions, are identified, with which the structural dimensions are linked through modalities of interpretive schemes, facilities, and norms (Jones and Karsten 2008). There is much debate about whether structure can be inscribed or embedded in technology (Poole 2009). This is an important claim as it posits that structures do not exist except in actors' memory traces or in "practice" (Orlikowski 2000). Drawing on the case example above, resources are the dispatcher's command to allocate field workers, systems holding information about the current location of field workers as well as human resource-related information. Rules are the policies which need to be followed when allocating field workers.

Technical objects extend 'structural features' (Markus and Silver 2008). Two of the original concepts from AST (features and spirit) have recently been redefined as technical objects, functional affordances, and symbolic expressions (Markus and Silver 2008) in order to address several concerns about the original concepts. IT artefacts are conceptualized in ways that help to hypothesize about them and investigate their potential effects while retaining the core insights of DeSanctis and Poole's initial conceptualizations (Markus and Silver 2008). Structural features are the specific types of rules and resources, or capabilities, offered by the system. Features within a SA for allocating field workers, for example, might include searching for employees, mapping their expertise to the domain that is searched for, or finding the employee closest to the location in which a field worker is needed. They govern exactly how information can be gathered, manipulated, and otherwise managed by users. In this way, features bring meaning (what Giddens calls "signification") and control ("domination") to group interaction (DeSanctis and Poole 1994). The concept 'technical object also considers packaging, arrangement and appearances of properties. As illustrated in the case example, this would be an important enhancement for IT effects and design studies with regard to SAs, as in particular the flexible (re-) combination of components into new SAs is seen as one of SAs' key characteristics.

Functional affordances describe what a user (or user group) can do with technical objects, given the user's capabilities and goals" (Markus and Silver 2008). Due to the fact that users design and develop SAs themselves, this relationship assumedly is closer compared to traditional applications. If SAs are designed with reusability by diverse user groups in mind, there should be comparably more functional affordances available compared to traditional applications.

Symbolic expressions clarify how users or user groups may interpret technical objects, comprising functional and values-oriented symbolic expressions. In contrast to sprit, symbolic expressions may relate to a technical object as a whole or to any of its components, making this concept particularly applicable for SAs. Thus, a SA may have various, even conflicting symbolic expressions for a specified user group (Markus and Silver 2008). Investigating these conflicts and how they are resolved within user communities is an interesting question. As SAs are typically developed by only one user or a very small group, there are assumedly fewer conflicts and their resolution might be easier compared to traditional applications. Communities dealing openly with conflicts should outperform communities with no clear governing rules.

Structuration occurs as actors move to invoke existing structures or to create new ones, producing and reproducing structures and associated social system (Poole and DeSanctis 2004). (Re-)Production of structure by action may not occur exactly as anticipated, as there may be unacknowledged conditions and unintended consequences of intentional action. Reproducing accepted behavior may therefore result in unintended consequences of also promoting other, potentially undesirable behavior (Jones and Karsten 2008). In the case example, formal and informal procedures of how to allocate field workers (social structure, i.e. resources and rules) directly influence developing the SA by guiding which requirements it will actually target. Using the SA creates a new IT-enabled allocation procedure, imposing new restrictions on structures. These new structures may also lead to new issues with regard to security and compliance with existing policies which in turn increase the need for adjusted governance. They can also reinforce old structures, e.g., the basic structure of the allocation process with respect to procedure and power distribution between central dispatcher and field workers.

Appropriation is defined as immediate, visible actions that evidence deeper structuration processes. Appropriation occurs when rules and resources from a technology or other structural source are brought into action. By looking at appropriation moves, researchers can uncover how a given rule or resource is brought into action (DeSanctis and Poole 1994). It is important to note that social structures (including also technology structures) are (re-)produced in social life. Structural features can therefore be appropriated in very different ways, visible by different appropriation moves. There are various factors influencing the selection of appropriation moves. With regard to group support systems, they have been found to depend on the group's internal system (styles of interacting, knowledge and experience, perceptions of others' knowledge, and agreement on appropriation) (DeSanctis and Poole 1994). Appropriation happens on a personal level, i.e. the individual local context of each employee reusing SAs, but also on a collective level, i.e. the community of users who jointly interpret and thus make sense of how SAs can successfully be brought into action.

Instrumental uses describe not only what and how structures are used, but also why they are used, i.e. purposes for which a (group of) user(s) decide(s) to deploy technology or other structures (DeSanctis and Poole 1994). Similar to the concept of faithfulness, features may be appropriated for different instrumental uses, or purposes. As SAs target individual employees, reasons and purposes are limited to those given by an individual. However, on the one hand employees collaborate and on the other hand if a community commits to the continuous development of a SA, then this will be the constituting unit for purposes of using this SA. Still, local contexts of individual employees might differ from each other substantially, so that instrumental use might carry different meanings for different employees.

Other sources of structure emerge as technology, task, and environmental structures are applied during the course of social interaction (DeSanctis and Poole 1994). Apart from advanced IT as one source of structure, further example sources of structure are content and constraints of a given work task, organizational environment, corporate information, histories of task accomplishment, meeting context (Niederman et al. 2008), cultural beliefs and modes of conduct (DeSanctis and Poole 1994). For example, the increasing presence of knowledge work and grassroots approaches in software development suggest to rethink models explaining utilisation of software and its impact on performance, e.g., the task-technology fit model (Goodhue 1995; Goodhue and Thompson 1995; Zigurs and Buckland 1998). Users now can vary all characteristics of the model, task, technology and individual when organizing their activities with supportive SAs. In other words, empowered employees play commanding roles with respect to other sources of structure which might be substantially more important than in case of software applications used for routine work.

Faithfulness describes whether appropriations of technology are consistent with the spirit and structural feature design (faithful appropriation) (DeSanctis and Poole 1994). It is an individual or group decision to appropriate technology features faithfully or unfaithfully. Again, SAs differ from traditional software in that much less effort is spent on elaborating on a technology's spirit in the design phase and capturing sufficient proportions of intended uses, so that unfaithful appropriations assumedly are much more likely to occur. Also, due to the fact that SAs can be recombined, it might be much less clear what actually constitutes an unfaithful application.

Unintended consequences often occur while using SAs and are often desirable. This effect will be even stronger as other users may use the SA as a component in their own SA supporting a different activity. In the case example, using the SA created the desire for other staff to compose other SAs, to further develop this SA, or to use it differently, for example as a simple visual tool to track peoples' locations. This might indeed lead to new behavior or unintended use in other contexts, such as to survey staff which will react to this in ways that might or might not be favourable to overall productivity. There might also be an effect on how field workers use the original systems because they will interpret this new procedure and act upon their intended goals, e.g., they might strive to update their yellow pages more regularly with entries signaling domains that they like to be assigned to and delete entries about domains that they like to avoid.

Autonomy is criticized as there may not always be feasible options to choose from for actors. In Giddens' view, actors act autonomously, however, structure is always enabling as well as constraining. Whatever effects technology has on social practice depends on how social agents engage with it in their actions. Structural constraint places limits upon the feasible range of options for an actor in given circumstances (Jones and Karsten 2008). With rising numbers of services available on the web and appropriate tools to compose SAs, end-users are now able to overcome some of the restrictions imposed by traditional IS. Users develop their own applications, thus enhancing their autonomy and agency. In our example, the allocation manager puts together a mashup to support her activity without calling in IT, avoiding overhead and formality of traditional IT procedures.

Agent's knowledgeability is always bounded on the one hand by the unconscious and on the other hand by the unacknowledged conditions and unintended consequences of action. Giddens views agents as being highly knowledgeable about what they do even if they are not always able to express it verbally and as actively involved in the enactment of social practices rather than being controlled by structural forces of which they are unaware. As a result, agents are knowledgeable about their actions, continuously reflect on their conduct and thus are not passive, subject to exogenous forces, or ignorant of the influences on their actions (Jones and Karsten 2008). The shift of development work towards end-users allows them to be more active and to more profoundly deploy their creativity, as well as to be more conscious about their actions. In addition, feedback given by the user community, in which the SA is published, used and shared, may lead to stronger reflection and responsibility about the SA. In the case example, the allocation manager receives direct feedback from the Intranet community on their reuse of the SA with regard to future improvements or other possible uses of the SA. Finally, feedback from other potential users can be immediately taken on board and built into a continuously evolving SA. Agents also learn about how to deploy their resources and gradually improve their rules.

Routines are "integral to the continuity of the personality of the agent…and to the institutions of society" (Giddens 1984, p. 60) and play an important role in sustaining societies. In often changing work environments, routines are difficult to establish and maintain. SAs might support routinization by filling in the gap in dynamic work environments that traditional ISs cannot fulfil. If a SA is reused widely in a community, it will serve its purpose and increase structure of an activity. The evolutionary character of SAs might also inhibit routinization, as always new and better adapted SAs can be created. For example, many variants of SAs might be used in the community thus rendering it more difficult to understand the structure of an activity or the underlying business process. SAs also facilitate social integration without co-presence, as seen in the case example. One of the key features of SAs is the ability to integrate different types of information. Here, SAs integrate temporal (when are field workers available) and spatial information (where are they located). This might result in fewer physical meetings necessary for allocating field workers.

Disembedding mechanisms is a recent, yet often neglected concept which nevertheless provides interesting perspectives for interpreting structuration processes in IS research (Jones and Karsten 2008). It discusses how social relations are disembedded, or "'lifted out' from local contexts of interaction and…restructur[ed] across indefinite spans of time-space" (Giddens 1990, p. 21). This aspect is important for SAs, which are explicitly exposed to a wider community of users and thus lifted out of the local context of the SAs' creators. In the case example, the SA could be published on the Intranet, so that other users could integrate it into their own SAs, supporting their local contexts.

Trajectory of the self describes how individuals reflexively construct a narrative of personal identity despite the more personal focus of modernity and self-identity (Giddens 1991) which may be relevant to IS researchers in understanding how individuals make sense of IS phenomena and how IS are involved in shaping personal identity. Giddens refers, for example, to the collage effect created by electronic media, whereby distant events increasingly intrude on everyday life [Jones and Karsten, 2008]. SAs might provide opportunities to actively engage in building software to support immediate needs. The corresponding individuation effect might strengthen personal identification not only with SAs, but also with the context in which they are used. Moreover, SAs are shared among individuals in the organization's Intranet or on the Internet who might form a community committed to improving IT support for a certain type of activities which might strengthen personal identification with a social institution crossing the boundaries of the employing legal institution.

Attitude towards technology impacts how technology structures are appropriated. Users might be comfortable in using technology, they might also value using the technology in their work situations; and they might be challenged by using the technology (Zigurs et al. 1991). These attitudes may direct the application of the technology and provide sufficient rigor and confidence to manage the task supported by the technology (DeSanctis and Poole 1994). The fact that SA users are designers should positively influence the attitude towards using the technology, particularly respect and challenge. Comfort, i.e. feeling secure to use a technology might be negatively influenced compared to traditional software applications as employees commit to the software and thus there is no "legitimation" from formal organizational units such as senior management or a steering board that typically sanction the use of traditional software.

Table 2: Differences between traditional and situational applications

	Traditionally developed enterprise applications	**Situational applications**
Structure	Introducing new applications breaks the evolutionary process of creating new structures by human agency in everyday activities and replaces them by those aiming at changed desired behaviour	The duality of structure and agency influencing each other, directly maps to SAs that are continuously refined in select-design-develop-use loops
Technical objects	Clear set of components which form part of an IT artefact. They might also include unintended properties	SAs are sets of loosely bundled capabilities that can be implemented in many different ways
Functional affordances	Technical objects are designed with clearly identifiable functional affordances by designers and possibly key users representing users	Close relationship between technical objects and users, conceptualized as functional affordance and symbolic expressions
Symbolic expressions	Technical objects are designed with clearly identifiable symbolic expressions by designers and possibly key users representing users	No clearly identifiable symbolic expressions; SAs are used as they fit in situations. Self-commitment of SA designers as users replaces domination and legitimation
Structur-ation	Application design attempts to anticipate as much as possible of desired production and reproduction of structures, conditions and consequences of corresponding actions	Design of SAs limits anticipation of structuration to the current situation and allows for recombining or reusing SAs in diverse conditions
Appropri-ation	Customization, deployment and change management make sure that appropriation happens in a coordinated way in line with organizational goals	Happens in a self-organized way on a personal level by each user of an SA as well as on a community level to jointly interpret successful appropriations
Instrumen-tal uses	Clearly identified and similar across the entire target groups of users	Diverse, as users of SAs and situations in which they use SAs differ from each other
Other sources of structure	Contingent factors need to be considered when designing applications, at the time of implementation and during maintenance, especially with new releases	Contingent factors are at the core of the very definition of SAs describing the types of situation(s) for which SAs are designed
Faithfulness	Goal is faithful appropriation as applications encourage desired behaviour. Low share of unfaithful appropriations which mostly conflict with organizational goals	High share of unfaithful appropriations; high proportion of unfaithful appropriations are in line with organizational goals
Unintended consequences	Low share of unintended consequences due to software established in order to encourage or even enforce desired behaviour	High share of unintended consequences due to low number of users envisioned and participating in design and design for reusability for ex-ante unknown purposes
Autonomy	Varying level of autonomy and agency depending on the level of structure of the processes or tasks to be supported.	High level of user autonomy and agency due to grassroots initiative and self-organizing end-user computing approach
Agent's knowledge-ability	Key users participating in design processes of traditional applications require a high level of knowledgeability and often represent larger groups of users with varying levels of knowledgeability	All users designing SAs require a high level of knowledgeability about their tasks as well as technology to support these tasks
Routine	Typically increase level of routine due to comparably long duration of traditional software development projects thus aiming to support more stable processes and tasks	SAs primarily target day-to-day activities and their impact on level of routine depends on stability of SAs and variations in reuse
Disembed-ding me-chanisms	Strong disembedding effects if applications are adaptable to varying situations and designed for flexibility and customization	Strong disembedding effects if SAs are designed for reusability outside the context of creation

Trajectory of the self	Applications are typically designed to abstract from the subjectivity of individual users into roles, tasks and resources and thus obscure their effects on shaping personal identities	Individuation means users commit to SAs and the organizational design of the context in which they are used.
Attitude towards technology	Users have positive and negative attitudes towards technology	All users designing SAs have a comparably positive attitude towards technology

DISCUSSION

SAs describe a relevant and innovative form of developing, offering and using software in organizations at the intersection between end-user computing, Web computing and component or service-oriented architectures. Structuration Theory has been used to shed light on the distinctive characteristics of SAs, most importantly the evolutionary processes of their selection, development and use that allow for new forms of appropriating software. As a result, SAs can be differentiated from traditional business applications with respect to (1) the processes of *designing and developing*, the SA process, (2) the *product* or outcome of these processes, i.e. the SA software, (3) the processes of *deploying and using* SAs and (4) the cross-sectional processes of *governing* and *guiding* the processes behind (1) – (3). This new perspective widely extends existing definitions that have been primarily developed from a technical perspective, neglecting social and organizational aspects of SAs. In the following we provide a brief outline which bundles of concepts in Structuration Theory could be used to further investigate social processes involved in developing, deploying and using SAs:

(1) Designing and developing: Timing is a distinct characteristic of the process of developing SAs focusing immediate solutions for current business problems in a particular social situation. Another differentiating factor is the focus on end-users as actors with no clear demarcation between design and development as well as between the roles of designers, developers and users. Users designing SAs are typically highly knowledgeable in the domain and have a positive attitude towards technology. Furthermore, autonomy is required for users designing software for their support. SAs are thus conceived in a grassroots approach instead of institutional arrangements and evolve organically in short, iterative cycles based on user feedback and participation thus blurring the boundaries between design time, build time and run time. In particular, concepts around autonomy and agency seem to be important for investigating characteristics of users and tasks (routine) suitable for SAs. Here, an interpretive approach would be useful to understand the context of the SA together with the process of mutual influences between SA and its context (Walsham 1995).

(2) Product: SA software is a set of loosely bundled capabilities that has been assembled from building blocks, reusing pre-existing software, both, software maintained inside an organization as well as publicly accessible third-party software. It combines or adapts, e.g., refines, extends or narrows existing content, data or application functionality in order to fit an immediate business need, but again representing a building block prepared for being found by other users and put to unforeseen further reuse. SA software typically employs less mature, emerging technologies and light-weight integration technologies. Here, particularly, characteristics of technical objects, functional affordances, symbolic expressions that are distinctive for SAs seem to be valuable concept to further investigate this area. A design science approach (Hevner et al. 2004) might be suitable, by developing design artefacts, i.e. IT infrastructures suitable for developing SAs, building blocks for SAs as well as SAs themselves and evaluating the effects they have on specific user groups or communities.

(3) Deploying and using: SAs are typically continuously shared and refined in select-develop-use loops among a small number of related individuals that maintain SAs they have conceived. As SAs are designed to solve an immediate business need, they are appropriated contingent to the concrete situation typically of day-to-day work practices in a self-organized way by committed individuals and a community jointly interpreting their success. Thus, they have a clearly identified intended use while explicitly encouraging diverse, unintended reuse. However, an explicit characteristic of SAs is the blurring of design and development on the one hand and deployment and use on the other hand as one concrete SA evolves organically and at the same time already represents the building block of other SAs. In particular, concepts focusing on ‘use’, such as intended use, faithfulness of appropriation and routine can be beneficial to shed more light to the complex processes behind individual and community appropriation. Here, traditional positivist approaches by testing hypotheses derived from initial propositions about effects of SAs on user behaviour might be suitable.

(4) Governance and guidance: From a managerial perspective, SAs pose a substantial challenge to the governance of corporate IT. With the advent of more light-weight technologies such as tagging, RSS and mashups, in many organizations employees are busy trying out new alternatives by arranging services flexibly to help them fulfill their needs. As a result, corporate IT is now challenged to provide an IT infrastructure including building blocks and tools to develop SAs as well as needs to take care of integrating SAs into a robust, scalable, enterprise-wide IT architecture that incorporates configurations implementing policies that are compliant with the organisation’s security requirements and thus ensures accountability of the IT infrastructure. Corresponding power distribution and rules for guidance will have to balance the need for an organizational and technical infrastructure compliant with the organization's regulatory environment without losing the momentum of the grassroots level approaches for user-designed SAs that supposedly can boost productivity of knowledge work. Understanding the structuration processes behind seems to be promising for exploring the difficult governance issues for guiding design, development and use of SAs. Governance is about specifying decision rights and providing accountability (Weill and Ross 2004). Many decisions have to be framed here, e.g., who decides on what types of SAs can be developed by users in what ways observing which rules and adhering to what standards. This is increasingly important as organizations establish information security management initiatives, require traceability and justification of all activities potentially affecting information security and get external certification of their compliance to standards such as ISO/IEC 2700x (www.27000.org) which might, for example, require auditing their organisation as well as suppliers of services used in SAs. Viewing governance structures (i.e., rules and resources in terms of Structuration Theory) as being continuously re-created in practice (Orlikowski 2000), at the same time analyzing them according to Giddens's original ideas of approaching the duality of structures with the help of the three dimensions of structure (domination, legitimation, signification) and interaction (communication, power, sanctions), linked by modalities (interpretive schemes, facilities, norms) makes the use of Structuration Theory even more appealing. This is particularly true with respect to the communities that emerge when several employees (re-)use a SA and jointly take care of its continuous development. Here, a set of regulations should concern instrumental uses, (unwanted) unintended consequences and the effects on routines while assessing and guiding users' autonomy, knowledgeability and attitudes towards technology. Hence, it would be useful to pursue a critical (McGrath 2005) or interpretive research agenda to investigate the multi-faceted issues and potential conflicts arising around governance and guidance.

Implications of this discussion include a much more concise description and definition of SAs contrasting traditionally developed business applications with SAs using key concepts of Structuration Theory. The extended conceptualisation of SAs now also includes a distinct social perspective for the four dimensions 'designing and developing', 'product', 'deploying and using', and 'governing and guiding' enabling researchers to explicitly focus on the recursive relationship between users' interaction and technology (Pozzebon and Pinsonneault 2005). At the same time, our discussion lays out a host of potentially beneficial research avenues to further explore research questions on SAs. As such, SAs would serve as an interesting vehicle to be studied under the lens of Structuration Theory, as this type of software also exhibit specific characteristics that so far have been paid insufficient attention in IS research. This includes some of Giddens' often neglected concepts such as disembedding mechanism and trajectory of the self, as well as the focus on wider institutional processes. The proposed research agenda for SAs fits well with recent recommendations for future research using Structuration Theory (Jones and Karsten 2008). This suggests focusing on single concepts of Giddens' work, such as some of those we used for contrasting SAs with traditional software, concepts that are still underexplored, such as broadening the perspective from micro-level towards macro-level institutional processes, or even working towards a consistent theoretical account of the IT artifact (Orlikowski and Iacono 2001). The latter would also include the further development of Orlikowski's practice lens to focus on the embodied and affective character of use practices (Jones and Karsten 2008).

CONCLUSION

Managerial implications of this research not only call for creating an organizational and technological infrastructure in support of SAs, but also for guidance of these grassroots approaches to software development. The concepts provided by these theories might help (1) improve understanding about the effects SAs have on organizational practices and business processes, (2) raise awareness in communities using SAs about social processes of designing, sharing and appropriating SAs "with structuration in mind", (3) explore how these social processes can be aligned with business goals and technical requirements, e.g., concerning performance and security as well as (4) elicit success factors and barriers to re-using SAs. Summing up, SAs are both, promising and risky and thus require careful consideration as to what aspects of knowledge-intensive business processes benefit from such an approach and the corresponding more liberal IT regime while others might be better organized with the help of traditional, professional IT and the corresponding more strict IT regime. This also calls for a revision of IT control objectives, e.g., as laid out in the COBIT framework (www.isaca.org), IT processes, e.g., as specified in the ITIL collection of best practices in design, development and operation of IT (www.itil.org) as well as security frameworks, e.g., the ones laid out in the ISO/IEC 2700x standards

REFERENCES

Balasubramaniam, S., Lewis, G.A., Simanta, S., and Smith, D.B. 2008. "Situated Software: Concepts, Motivation, Technology, and the Future," *IEEE Software* (25:6), pp 50-55.

Barthes, R. 1974. *S/Z: An Essay*. San Francisco: Hill and Wang.

Chatterjee, D., Grewal, R., and Sambamurthy, V. 2002. "Shaping up for E-Commerce: Institutional Enablers of the Organizational Assimilation of Web Technologies," *MIS Quarterly* (26:2), pp 65-89.

Cherbakov, L., Bravery, A., Goodman, B.D., Pandya, A., and Baggett, J. 2007a. "Changing the Corporate It Development Model: Tapping the Power of Grassroots Computing," *IBM Syst. J.* (46:4), pp 743-762.

Cherbakov, L., Bravery, A., and Pandya, A. 2007b. "SOA Meets Situational Applications, Part 1: Changing Computing in the Enterprise," in: *IBM developerWorks*.

Cooper, R.B., and Zmud, R.W. 1990. "Information Technology Implementation Research: A Technological Diffusion Approach," *Management Science* (36:2), pp 123-139.

Crupi, J., and Warner, C. 2008. "Enterprise Mashups Part I: Bringing Soa to the People," in: *The SOA Magazine*.

DeLone, W.H., and McLean, E.R. 1992. "Information Systems Success: The Quest for the Dependent Variable," *Information Systems Research* (3:1), pp 60-95.

DeSanctis, G., and Poole, M.S. 1994. "Capturing the Complexity in Advanced Technology Use: Adaptive Structuration Theory," *Organization Science* (5:2), pp 121-147.

Dillon, A., and Morris, M. 1996. "User Acceptance of Information Technology: Theories and Models," *Annual Review of Information Science and Technology* (31).

Giddens, A. 1984. *The Constitution of Society*. Berkeley: University of California Press.

Giddens, A. 1990. *The Consequences of Modernity*. Stanford: Stanford University Press.

Giddens, A. 1991. *Modernity and Self-Identity*. Stanford, CA: Stanford University Press.

Goodhue, D.L. 1995. "Understanding User Evaluations of Information Systems," *Management Science* (41:12), pp 1827-1844.

Goodhue, D.L., and Thompson, R.L. 1995. "Task-Technology Fit and Individual Performance," *MIS Quarterly* (19:2), pp 213-236.

Hevner, A.R., March, S.T., Park, J., and Ram, S. 2004. "Design Science in Information Systems Research," *MIS Quarterly* (28:1), pp 75-105.

Hoyer, V., and Fischer, M. 2008. "Market Overview of Enterprise Mashup Tools," *6th International Conference on Service-oriented Computing*, Sydney: LNCS, pp. 708-721.

Jones, M.R., and Karsten, H. 2008. "Gidden's Structuration Theory and Information Systems Research," *MIS Quarterly* (32:1), pp 127-157.

Jones, M.R., and Karsten, H. 2009. "Divided by a Common Language? A Response to Marshall Scott Poole," *MIS Quarterly* (33:3), pp 589-595.

Kaufman, L.M. 2009. "Data Security in the World of Cloud Computing," *IEEE Security and Privacy* (7:4), pp 61-64.

Kontopolous, K.M. 1993. *The Logics of Social Structure*. Cambridge, UK: Cambridge University Press.

Lewin, A.Y., and Minton, J.W. 1986. "Determining Organizational Effectiveness: Another Look, and an Agenda for Research," *Management Science* (32:5), pp 514-553.

Markus, M.L., and Silver, M.S. 2008. "A Foundation for the Study of IT Effects: A New Look at Desanctis and Poole's Concepts of Structural Features and Spirit," *Journal of the Association for Information Systems* (9:10), pp 609-632.

McBride, N., and Wood-Harper, A.T. 2002. "Towards User-Oriented Control of End-User Computing in Large Organisations." *Journal of End User Computing* (14:1), pp 33-41.

McGrath, K. 2005. "Doing Critical Research in Information Systems: A Case of Theory and Practice Not Informing Each Other" *Information Systems Journal* (15:2), pp 85-101.

Niederman, F., Briggs, R.O., de Vreede, G.J., and Kolfschoten, G.L. 2008. "Extending the Contextual and Organizational Elements of Adaptive Structuration Theory in GSS Research," *Journal of the Association for Information Systems* (9:10), pp 633-652.

O'Dell, C., and Grayson, C.J. 1998. "If We Only Knew What We Know: Identification and Transfer of Internal Best Practices," *California Management Review* (40:3), pp 154-174.

Orlikowski, W.J. 2000. "Using Technology and Constituting Structures: A Practice Lens for Studying Technology in Organizations," *Organization Science* (11:4), pp 404-428.

Orlikowski, W.J., and Iacono, C.S. 2001. "Research Commentary: Desperately Seeking the 'IT' in IT Research - a Call to Theorizing the IT Artifact," *Information Systems Research* (12:2), pp 121-134.

Pahlke, I., Wolf, M., and Beck, R. 2010. "Enterprise Mashup Systems as Platform for Situational Applications," *Business & Information Systems Engineering* (2:5), pp 305-315.

Poole, M.S. 2009. "Response to Jones and Karsten, "Giddens's Structuration Theory and Information Systems Research," *MIS Quarterly* (33:3), pp 583-587.

Poole, M.S., and DeSanctis, G. 1990. "Understanding the Use of Group Decision Support Systems: The Theory of Adaptive Structuration," in: *Organizations and Communication Technology,* J. Fulk and C. Steinfield (eds.). Newbury Park, CA: Sage, pp. 175-195.

Poole, M.S., and DeSanctis, G. 2004. "Structuration Theory in Information Systems Research: Methods and Controversies," in: *The Handbook of Information Systems Research,* M.E. Whitman and A.B. Woszczynski (eds.). Hershey: Idea Group, pp. 206-249.

Pozzebon, M., and Pinsonneault, A. 2005. "Challenges in Conducting Empirical Work Using Structuration Theory: Learning from It Research," *Organization Studies* (26:9), pp 1353-1376.

Rogers, E. 1995. *Diffusion of Innovations*. New York: Free Press.

Schultze, U. 2000. "A Confessional Account of an Ethnography About Knowledge Work," *MIS Quarterly* (24:1), pp 3-41.

Sewell, W.H. 1992. "A Theory of Structure: Duality, Agency, and Transformation," *American Journal of Sociology* (98:1), pp 1-29.

Shirky, C. 2004. "Situated Software in Clay Shirky's Writings" in: *Networks, Economics, and Culture*.

Strauss, A. 1975. *Negotiations: Varieties, Contexts, Processes and Social Order*. San Francisco: Jossey-Bass.

Walsham, G. 1995. "Interpretive Case Studies in IS Research: Nature and Method," *European Journal of Information Systems* (Vol. 4), pp 74-81.

Weill, P., and Ross, J. 2004. *IT Governance: How Top Performers Manage It Decision Rights for Superior Results*. Boston (MA, USA): Harvard Business Press.

Zigurs, I., and Buckland, B.K. 1998. "A Theory of Task/Technology Fit and Group Support Systems Effectiveness," *MIS Quarterly* (22:3), pp 313-334.

Zigurs, I., DeSanctis, G., and Billingsley, J. 1991. "Adoption Patterns and Attitudinal Development in Computer-Supported Meetings: An Exploratory Study with Samm," *Journal of Management Information Systems* (7:4), pp 51-70.

COPYRIGHT