DESIGNING TO SUPPORT COMPLEX ORGANIZATIONAL WORK: A PRAGMATIC APPROACH

Arvind Karunakaran, MIT Sloan School of Management, Cambridge, Massachusetts 02142, USA, arvindk@mit.edu
Sandeep Purao, College of Information Sciences and Technology, Pennsylvania State University, University Park, Pennsylvania 16802, USA, spurao@ist.psu.edu
Jingwen He, College of Information Sciences and Technology, Pennsylvania State University, University Park, Pennsylvania 16802, USA, jxh497@ist.psu.edu

Accepted to the International workshop on IT Artefact Design & Workpractice Intervention, 10 June, 2012, Barcelona

Abstract
Navigating complexity in day-to-day organizational work remains an important yet difficult concern. Extant approaches to design for supporting complex organizational work has extolled a logic of “decomposition” and “completeness”. Such approaches, inherited from contexts where the product and process of work is tangible and physical, have been shown to be counterproductive in contexts where these assumptions are challenged. We propose a pragmatic approach to design for supporting the performance of complex organizational work. In contrast to prior approaches, ours embraces a logic of “chunking” and “incompleteness”. We derive and demonstrate our arguments through building two research artifacts (Semantic Procedure Analyzer and ReKon), in two different domains (Petrochemical refineries and Software integration), aimed at supporting actors to navigate the complexities that arise during the day-to-day performance of work. The paper describes the approach, the two research artifact, and an outline of design principles derived from a comparative analysis of the two artifacts and their preliminary evaluation. We conclude by pointing to directions for future research.

Keywords: Complexity, Decomposition, Modularity, Layered Modularity, Templates, Procedures, Artifacts, Chunking, Recombination.
1 Introduction

 Complexity is the hallmark of modern society (Greenwood, Raynard, Kodeih, Micelotta, & Lounsbury, 2011; Scott & Meyer, 1994). The epochal shift from gemeinschaft (community) to gesellschaft (society) has infused complexity into the fabric of our everyday social life (Tönnies 1988). Organizations, as social forms, are no exception to this burgeoning complexity (Anderson, 1999; Axelrod & Cohen, 2001). While organizations make it possible to achieve results that, in scale and scope, are well beyond the reaches of individuals working in isolation, they nevertheless constantly grapple with complexity (Galbraith, 1973). Much of the early work in organizational studies deals with proposing structures and mechanisms to design organizations that can better manage and navigate complexity (Lawrence & Lorsch, 1967; Thompson, 1967). Later works in this arena (e.g. Baldwin & Clark, 2000; Henderson & Clark, 1990; Von Hippel, 1990) draw upon Simon’s (1962, 1996) work on modularity.

 Modularity is a scale-independent and pervasive principle. It offers a way to address complexity via decomposition. Such decomposition may include, for example, loosely coupled product architectures (Ulrich, 1995) or functional sub-units of firms (Roberts, 2007). Many scholars point to modular architectures as the foundation that has lead to profound changes to the nature of technological innovation and the structure of industrial organizations (Baldwin & Clark, 2000; Langlois & Robertson, 1992; Schilling, 2000). The rise in modular architectures has paralleled the rise in the usage of information technology for handling the complexity of organizational work. For instance, business process management offered a way to handle complexity by modularizing organizational work into component activities, mapping these onto different organizational functions, grouping them into processes, and finally, creating codified knowledge out of these processes (Davenport & Short, 2003; Hammer & Champy, 1993).

 However, the results from such efforts have remained tenous at best. Managing complexity in organizational work via modularization has lead to negative outcomes (Schultze & Stabell, 2004; Sterman, Repenning, & Kofman, 1997) at times, even entrapping the organization (Fahey & Prusak, 1998; Repenning & Sterman, 2002).

 We argue that applying principles of modularity toward the management of complex work might be well-suited only in a context where the product and process of work is tangible, physical and distinct (Okhuysen & Bechky, 2009; Yoo, Henfridsson, & Lytinen, 2011). The physicality of manufactured products (e.g. a car or a watch) could make it possible to modularize work via hierarchical decomposition (Simon, 1962) of work activities into modules and tasks that could be assigned to individuals or groups, and later, to integrate these individually completed modules and tasks back into the product unit. However, when the product and process of work is not all that tangible, physical and distinct (e.g. a software or a drug or a chemical), then a straightforward application of modularity principles could lead to counterproductive results. In the case of the former, it is indeed more or less possible to undertake a “scientific approach” (Garud, Jain, & Tuertscher, 2008) to design through a complete specification of activities, modules, processes, interface standards, and design rules. In the case of the latter, such a complete specification is neither possible nor desirable. Consequently, in contexts where the product is not physical and process of work is interwoven, one needs to embrace a “pragmatic approach” (Dewey, 1934; Garud, et al., 2008) to design that is based on the logic of chunking and incompleteness (rather than the logic of decomposition and completeness).

 In this paper, we advance such a pragmatic approach to design rooted in the logic of chunking and incompleteness. By pragmatic approach, we refer to mechanisms that could facilitate actors-on-the-ground to generate actionable knowledge via using, re-using, and re-combining resources-at-hand (cf. Boisvert, 1998; Goldkuhl, 2012). Our work draws on empirical investigation related to supporting complex organizational work in two domains: Petrochemical refineries and Software integration. In these two domains, the product of work is not tangible. In addition, the processes required to build the product
is interlaced (Tuertscher, 2008). We elaborate our approach with the design and instantiation of two artifacts — Semantic Procedure Analyzer (for Petrochemical refineries) and ReKon (for Software integration) — aimed at supporting knowledge needs of employees towards sensemaking, managing, and navigating complexities they face during their day-to-day performance of work.

Our pragmatic approach to design advocates, illustrates and refines three foundational principles: a) Knowledge needed to navigate the complexities of work is often tacit, but is also materially anchored in concrete artifacts that become ‘starting points’ for actors to perform their work; b) Chunking and digitizing these artifacts, and making them available to actors as 'knowledge chunks' via a platform, enables these actors to handle complexity in real time and in an ongoing fashion; c) The resultant platform needs to be open, incomplete and underspecified, so that it could generate digital options in real time that could be recombined and reused over time.

In the remainder of the paper, we first elaborate upon the research approach, then move to describe how the design principles and foundations are illustrated by the two software artifacts; and finally, conclude with a discussion of future research steps.

2 Research Approach and Methods

We follow the canonical design research approach influenced by action design research (Sein, Henfridsson, Purao, Rossi, & Lindgren, 2011) that suggests knowing via building (Kuechler & Vaishnavi, 2008), aided by empirical field work. We do this through 3 iterative and interlocking activities, coupled with ongoing reflection and refinement (See Figure 1).

We began with informal interviews of practitioners from two domains to understand the processes they follow in their day to day performance of work. Following purposive sampling (Patton, 2002), we selected Petrochemical refineries and Software integration as the two domains because a) their product of work is not physical and tangible; b) involves high-level of process intensity; and c) these processes needed to develop the product are interwoven. Particular organizations were selected based on the criteria that a) they were in operation for at least 25 years, and b) involves complex production work. During this
stage, we collected more than 1500 real-world artifacts from these organizations that were used during their day-to-day operations. These included project templates, checklists, guidelines, standard operating procedures, and others. We analyzed those artifacts following qualitative artifact analysis (Bechky, 2008; Carroll & Rosson, 1992). We examined attributes such as format, length, and content of each artifact to make claims about ways in which they may support (or hinder) the performance of work.

Based on the analysis, we developed a set of design principles and heuristics for a) chunking and categorizing those artifacts; b) development of the two platforms. First versions of the platforms – Semantic Procedure Analyzer and ReKon - were then conceived and built. A formative evaluation pointed to prima facie potential usefulness of the respective platforms.

Later, we iterated the platforms based on the insights from the formative evaluation and from ongoing field work. In the case of ReKon, formal semi-structured interviews were then conducted with different employees from the software integration. We conducted 11 from one organization engaged in legacy application support, characterized as high process intensity, with more than 3000 employees worldwide; and 7 from the another organization engaged in information management, characterized as very high process intensity, with more than 220,000 employees worldwide. In the case of Petrochemical refineries, we conducted 12 semi-structured interviews from two petrochemical refineries organizations to understand operator behaviors and how these overlap with the actions specified in operator procedures. Also, multiple demonstrations of the evolving platform were conducted along with informal cycles of feedback aimed at clarifying different features, assumptions about inputs formats and the nature of outcomes generated from these inputs. Based on the insights obtained in these empirical investigations, we returned to iterating the design of the two platforms.

These research steps that took place from January 2009 to March 2012 were accompanied by a concurrent and ongoing stage (similar to Sein et al, 2011) to reflect on the efforts and outcomes. This ongoing reflection has allowed derivation and continual refinement of design principles. We elaborate these in the next section.

3 Problem Specs, Platform Builds, and Design Principles

Both the Petrochemical refineries and Software integration domains are ripe with complexity. In both, the product of work is not tangible or physical and the processes of work are interlaced and filled with hidden interdependencies (Okhuysen & Bechky, 2009).

3.1 Problem Statements

A typical petrochemical refinery is modularized into many plant sub-units, interconnected via numerous upstream and downstream workflows. These workflows are extremely time-sensitive. A delay in production in one of the sub-units can disrupt operations in the entire plant, jeopardizing the functioning of the refinery, and may even cause major accidents. These accidents can affect not just property but also cause loss of lives; can damage not only industrial assets but also communities surrounding the refinery. It is true that these accidents are less likely today than they were three decades ago (Nivolianitou, Konstandinidou, & Michalis, 2006). However, they were not “magically” brought down one fine day. Rather, it took collective effort on part of the petrochemical refinery operators. During their tenure at the refinery, the operators acquire and cultivate expertise and tacit knowledge about running the plant that becomes hard to acquire for newcomers. These include knowledge about the hidden dependencies between different sub-units, mitigation for workflow delays, instrument failure diagnosis and the like. But the looming wave of retirement is expected to reduce the ranks of expert operators in the petrochemical refineries by as much as 20% in the next several years, creating a potential train-wreck scenario (Strahan, 2010). The problem statement for the industry is simple: with this impending wave of retirement, how
would the refineries leverage extant knowledge, and be able to sustain and manage the complex nature of their work?

Software integration too is ridden with complexities. Integration work requires domain knowledge, knowledge of the processes followed at the client site, and system-specific knowledge, including details such as native data formats, file structures, database schemas, APIs, interoperability standards, data porting rules and the like. Such knowledge is often tacit, acquired and internalized over the years by system architects and senior integration engineers (Karunakaran, Purao, & Cameron, 2009; Umapathy, Purao, & Barton, 2008). In addition, integration work requires assembling not systems, but systems-of-systems (Brownswor, Fisher, Morris, Smith, & Kirwan, 2006). These system-of-systems cannot be parsed, modularized and assembled as monoliths; instead, they emerge and evolve with little by way of complete visibility or control for any single individual or team engaged in the integration effort (Lam, 2005). As a result, the track record of such integration projects to tend to be below average (see, e.g. Charette, 2005)). In addition, a high turnover rate in the software industry makes it difficult for software integration organizations to continually leverage upon the experiential knowledge that the individual architects and engineers had acquired over the years. The problem statement for the industry is driven by these concerns: how can knowledge about these complex organizational efforts be captured and made available to successive generations of systems integration professionals?

Extant approaches propose a return to applying principles of modularity so that sociotechnical systems can be designed in a “scientific” way that could manage the complexity of such organizational work. However, due to the latent interdependencies among component parts (Alexander, 1964; Staudenmayer, 1997) and due to the sticky and tacit nature of the knowledge that goes beneath component activities (Brown & Duguid, 2001; von Hippel, 1994), we may predict that such systems can yield negative outcomes (Boehm, 1984; Boehm & Papaccio, 1988; DeMarco, 1995; Heath & Staudenmayer, 2000; Lam, 1994).

3.2 Proposed Approach and the Specification of a Class of Problems

We propose a pragmatic approach to design (Garud, et al., 2008) that leverages upon the real-world artifacts used during the day-to-day performance of work. These are “live” artifacts that get used by actors-on-the-ground on a frequent basis (Orlikowski, 2006). Examples of such artifacts in the software integration domain include functional specification documents, high-level and low-level design templates, code review checklists, effort estimation spreadsheets, boilerplate contract documents and others. Examples of such artifacts in the petrochemical domain include standard operating procedures, job-specific training documents, plant-specific check lists, function-specific guidelines and more. These artifacts act as material anchors (Hutchins, 2005) of knowledge that captures the rules, routines, guidelines and practices acquired from the past. Our fieldwork and interviews with petrochemical operators and system integration engineers suggest that these artifacts, rather than being comprehensive codified accounts of past knowledge, act as useful “starting points” for people performing their tasks.

I mean I usually use them as like starting points for...to get ideas and to kind of make sure I will be covering all the basis...I usually just use it to kind of gather my thoughts and think, try to picture I’m thinking more comprehensively about what we need to do....Especially when things are still broad. (Italics added for emphasis, Domain: Software Integration)

Well, a majority of them you read through them, you are not going to remember every single one the first time you read through. But like they give you a good idea about the plant for the most part. (Italics added for emphasis, Domain: Petrochemical Refineries)

Consequently, these artifacts provide a more feasible alternative to manage the complexity of organizational work as compared to the high abstraction offered by extant approaches such as BPM.
For instance, consider the example standard operating procedures (SOP) to startup the anti-foam carrier (See Figure 2). This SOP has instructions concerning “what to” do to start up the carrier. For example, a initial portion of the SOP says that in order to prime the pump 24.P12.A/B, one need to open suction valve B or C to the anti-foam pump and place a drip pan under the anti-foam pump discharge vent D or E, and finally, remove the cap and open the anti-foam pump discharge vent D or E. Although this SOP does not contain each and every detail related to the anti-foam carrier, it offers a good starting point for the operators toward navigating the complex terrains of the petrochemical plant.

Similarly, consider another example (Figure 3) of a template for gathering requirements for a system integration project in the telecom domain. This template captures the broad set of questions and items needed for gathering requirements, pointers that the project participants can use to structure the requirement gathering task. This template too does not offer a “comprehensive” picture about gathering requirements for a telecom project. Rather, it offers a good starting point for the integration engineers to generate some momentum and “get things rolling”. As one engineer reflects:

I would say I would never start the deliverable without a template. In fact I would never start anything without a template. Without it you are basically shitting. (Italics added for emphasis. Domain: Software integration)
Artifacts such as these do not represent a modularized workflow or abstracted method. Instead, they provide an instance of concrete and realizable action knowledge that is experientially derived, symbolically articulated and materially anchored (Hutchins, 2005; Latour, 1986; Orlikowski, 2006).

There are, however, problems associated with the use and reuse of these artifacts. First, there are too many of such artifacts, so the search costs to locate an appropriate one tend to be high. Second, each of these artifacts can run into several pages, but only a portion is likely to be relevant for the task-at-hand. Yet, no single artifact can fully support a task - the needed knowledge is distributed across multiple artifacts (Karunakaran, et al., 2009).

Based on these, we derived initial design concerns for the class of problems: How to support actors navigate the complexity in their work through facilitating the reuse and recombination of concrete artifacts? How to overcome problems associated with length and coarse-granularity of these artifacts?

In line with our knowing-via-building epistemology, we embarked on the design process through developing two platforms – Semantic Procedure Analyzer (for the Petrochemical refineries domain) and ReKon (for the software integration domain). We began with a set of simple principles concerning: (a) populating the platform with real-world “live” artefacts; (b) chunking those artefacts into smaller knowledge units based on few pre-specified criteria; (c) keeping the platform open by allowing actors to recombine the knowledge chunks and input them back into the platform. In the next two sub-sections, we briefly outline the implementation of these two platforms and reflect about the emergent design principles.

3.3 Platform Builds

3.3.1 Semantic Procedure Analyzer (SPA) for the Petrochemical Industry

We started building the Semantic Procedure Analyzer (SPA) platform by populating it with hundred of standard operating procedures (SOPs). We then “extract” smaller units from these SOPs and chunk them as logical knowledge units in the form of “services”. In this task we acknowledge a) that the operator procedures contain action-knowledge (Argyris, 1993; Bera & Wand, 2009) written in natural language that must be parsed before identifying services; b) the importance of domain knowledge in this parsing
task; c) overlaps across procedures suggest that these commonalities can be explored to identify common
services across procedures; d) procedures such as different operators, locations and time-spans provide
possible leads for identifying boundaries around services. Because each of these strategies is unlikely to
produce an error-free set of outcomes, we therefore argue for a heuristic (Pearl, 1984), instead of
algorithmic, approach for knowledge extraction from operator procedures.

Thus, we conceptualize each SOP as a set of statements written in natural language: some containing
instructions, others containing information such as pre-amble. We used heuristics to extract these
instructions in the form of “services”. The heuristics analyze the instructions by leveraging their
structural, syntactic and semantic attributes. The heuristics first use part-of-speech tagging (Charniak,
1997) to identify elements in each instruction such as actions, actors and objects. These are then subjected
to other heuristics to identify and extract services.

Figure 4 provides a screenshot of the SPA platform. The “red” lines specify the boundaries where a SOP
could be logically chunked into smaller knowledge units. These red lines are suggested by several
heuristics that we briefly outline below. A petrochemical operator could check if these boundaries are
appropriate. If not, s/he can adjust and move the red lines up and down, and decide which chunk is
appropriate for the task-at-hand. In addition, a chunk from a particular SOP could be recombined with a
chunk from another SOP to generate a new artifact that could cater to a new form of task (say, due to the
inclusion of a new device in the plant) or a new contingency (say, due to some delays forecasted in the
upstream production).

Figure 4. Semantic Procedure Analyzer (SPA)

We had a number of heuristics for chunking a SOP into logical knowledge units. One such heuristic scans
the procedure to locate its title. A taxonomy of equipments and units in the refinery aids this heuristic.
Based on the position of the title, heuristics 2 and 3 locates the meta-information and main body of the
procedure is located relative to the position of title. Another heuristic pre-processes the main body of the
procedure. This pre-processing is accomplished by extending part-of-speech-tagging (POST) (Charniak,
1997) with terms from the lightweight ontology that reflect domain knowledge along with a dictionary that differentiates action, such as predicates, subjects, objections, and conjunctions. These heuristics allow re-structuring each instruction into action descriptions made up of a responsible actor, the subject of action and a verb that indicates the action.

A few other heuristics chunk the procedure based on time, location and actors-involved. Time-based heuristic chunks instructions within a procedure that occur without a break in action. These are instructions that are not interrupted by conditions such as waiting for an external action or till a certain parameter value is reached. These instructions may be performed by one or more operators, in sequence or in parallel without an apparent pause. We use indicators of interruptions obtained from the domain experts for this purpose. For example, indicators such as “wait” and “till” provide clues about the boundary around an instruction cluster. Location-based heuristic chunked the instructions within a procedure based on the physical proximity of their target objects. For example, a procedure may contain contiguous instructions but one may require action such as “turning a valve” in one location, followed by “checking a meter” in another location. If these two locations are far, then it is an indication that these two instructions, in spite of their apparent sequence without pause within the procedure, belong to different modules. In the actor-based heuristic, actors responsible for each instruction provide another clue to instruction chunking. This is true because many SOPs are complex and require coordination across multiple operators. For example, a procedure to shutdown a unit may require field operators to carry out some instructions (such as “closing a valve”), and the console operators to follow others (such as “monitoring a parameter”). Implementing this heuristic requires examining the Subject for each instruction in the procedure. A change in the subject suggests the boundary around a cluster of instructions.

The outcome of our approach is a set of chunks, rendered in the form of services - each containing action knowledge that may be independently maintained, manipulated, and tailored by different operators.

3.3.2 ReKon for the Software Integration Industry

For building ReKon, we first collated a set of 1200 real-world project templates obtained from four software organizations. We analyzed those templates and started generating template chunks. To generate the template chunks, we started with qualitative artifact analysis (Bechky, 2008; Carroll, 2002; Carroll & Rosson, 1992). First, we examined the format, length, and other attributes (such as the number of sections, sub-sections, labels etc.) of each template, and recorded these. We then made “claims”(Carroll & Rosson, 1992) about the ways in which these templates could be used, and how they could enable or constrain process design for software development work. Using these, we chunked and categorized each template. Each chunk was placed in a matrix of Phases and Tasks constructed by consulting Project Management Institute’s PMBOK and Lam and Shankararaman’s “enterprise integration” methodology (Lam & Shankararaman, 2004). Some examples of Phases include: Planning, Market Research, Requirements Gathering, Implementation, and Testing. Some examples of Tasks include: IP Waiver, Status Reports, and Client Interaction. The intersection Phases and Tasks provided “cells”. Each cell was populated with template chunks.

To ensure fidelity for this chunking and categorization process, multiple coders worked through multiple rounds. First, a random sample of 122 documents (~10% of the set) was chosen. Coders established common guidelines for chunking and categorizing these templates. They identified headings and sub-headings of individual templates, and parsed these to decide if a template chunk could fit into a particular cell. For example, a client interview protocol represents a template chunk that fits in the cell at the intersection of conducting interviews (task) and gathering requirements (phase). Interview protocols available in multiple templates – say, for different variety of clients (e.g., Small businesses, Large enterprises), or for different types of projects (e.g., Web Development, Legacy System Maintenance) - were separated and placed in this cell. Multiple rounds of chunking and classifications allowed
comparisons and discussions (to resolve differences) across coders. Inter-coder rate of 78% and 86% was obtained during the two rounds, suggesting high levels of agreement (Cohen, 1960).

The complete set (~1220 templates) were then divided and assigned to coders who chunked each into logical fine-granular knowledge units for placement in a cell in the matrix. The templates assigned to each cell were then examined to select the best templates following simple heuristics, such as number of sections, thoroughness of descriptions, and availability of examples. Figure 4 shows a screenshot of the ReKon artifact built to contain these template chunks. It outlines the Phases along the leftmost column, and the Tasks along the top row. Choosing a Task in the top row shows the template chunks available to structure the task for each Phase. This screen below (in Figure 5) captures the Task RFP development and the template chunks available for this task for the Phases: Planning and Market Research.

A system integration engineer could look at these template chunks by navigating into a particular cell. S/he could use a particular chunk from a template, if it is appropriate for the task-at-hand. Else, s/he could combine a template chunk (say, a generic code review template) with another template chunk (say, data porting rules for telecom domain) to generate a new template for a particular task.

3.4 Design Principles and Future Iterations

A formative evaluation of SPA and ReKon pointed to prima facie potential usefulness of the respective platforms. In the case of ReKon, formative evaluation was conducted with users involved in working on complex, real-world, process-intensive projects to implement integration solutions as part of a course. The evaluation consisted of two steps. First step, conducted prior to the introduction of ReKon, assessed the size and number of template chunks compared to having coarse-grain templates. The second step, conducted after the student users explored ReKon for a few weeks, assessed properties such as appropriateness and relevance for project needs. In the case of SPA, formative evaluation was conducted using example procedures from multiple organizations. A varied set of procedures that largely focused on describing operator tasks around the Hydrocracker unit were used to assess the appropriateness of chunking mechanisms employed by SPA. Outcomes suggested evidence that the platform was able to apply heuristics to parse and chunk the SOPs into smaller knowledge units.

Based on the building of these two platforms, we were able to successively refine the design principles that we had previously outlined. The above description of the platforms illustrates such refinements.
First, a pragmatic approach to design leverages upon the existing real-world artifacts by actors used during the day-to-day performance of their work. Although the knowledge needed to navigate the complexities of work is often tacit, they are also materially anchored in such “live” artifacts. These artifacts play a dual role in that they become a) “maps” for sensemaking when an actor embarks on a new complex task; b) “devices” for triggering action when an actor encounters a contingency during the performance of a task. Therefore, they provide a more feasible alternative to manage the complexity of organizational work as compared to the higher abstraction and coarse-granularity offered by extant approaches such as BPM.

Second, since these artifacts are “digitized", they afford a certain agility that goes beyond the limited capabilities of modularization and hierarchical decomposition. That is, these digitized artifacts could be “chunked” (as opposed to “hierarchically decomposed”) in a variety of ways (e.g. time-based, event-based, actor-based) with the help of simple heuristics. These chunks are not just “modularized", but instead, “layered modularized” (Yoo, et al., 2011). Layered modularization is the “hybrid of the modular architecture of a physical product and the layered architecture of digital technology” (Yoo, et al., 2011, p.725). To elaborate, principles of modularization provide a scheme for hierarchically decomposing a physical product into work processes and component sub-assemblies, based on functionality, which could then be integrated back based on interface standards and design rules. Consequently, these standards, relationships and rules need to be fixed and clearly pre-specified (Baldwin & Clark, 2000). However, in the context of complex organizational work, such standards, relationships and rules cannot be fully pre-specified (Garud, et al., 2008). Attempts at such complete pre-specification would devolve to become rigidities that ensue in performance traps (Garud & Kumaraswamy, 2005; Repenning & Sterman, 2002). For instance, actors might not be able to apply their past experiential learning to handle contingencies in real time (Weick & Sutcliffe, 2001). Worse, attempts to fully categorize such contingencies would yield of “normalization of deviance” (Vaughan, 1996) and cause major accidents. Layered modularization, on the other hand, is based on not a complete, but instead, an incomplete specification of standards, functions, rules and relationships. Such incompleteness affords recombination and remixing of homogenized data in the form of digitized “chunks”. These chunks are not hierarchically decomposed based on functionality alone. Instead, they are logically demarcated based on criteria (e.g. size, event, location) that itself is emergent and open to constant modification. Hence, residual control is retained within the hands of actors on the ground (e.g. petrochemical operators, system integration engineers). Indeed, it is these actors - of what Hayek (1945) refers to as the “man on the spot” - who would have the prescience about handling emergent contingencies. The “mix-and-match” capabilities (Yoo, et al., 2011) afforded by these chunks help actors to formulate work-throughs and work arounds (Strauss, 1988; Suchman, 2007), and generate ingenious ways to handle wicked problems that arise during the moments of work (Ciborra, 1996).

Third, since layered modularity is self-referential (Yoo, et al., 2011) and emergent (Garud, Kumaraswamy, & Sambamurthy, 2006) in nature (e.g. one could come up with new templates and procedures through recombining existing templates and procedures), the resultant platform that contains the digital chunks needs to be kept open for user contribution. The newly recombined templates and procedures needs to be feed backed into the platform. Such continual user contributions strengthens organizational memory, and generates digital options (Sambamurthy, Bharadwaj, & Grover, 2003) in real time that could be recombined and reused over time.

Based on these refinements and insights obtained from formative evaluation, we decided to design the next iteration of the two platforms. We also returned to the field to conduct semi-structured interviews to understand how templates and procedures were actually created, deployed and used within organizations. Analysis of data gathered from the interviews revealed additional themes. For instance, we found that most of the newly recombined templates often stays within the laptop/workstation of the individual engineers, and rarely gets back into the central document repository where it could be referred to and reused by other employees. Current document repositories and process and knowledge management
systems lack the needed workflows and design features to facilitate the feedback process. Consequently, we plan to iterate ReKon and incorporate three major features: (a) social tagging of templates; (b) tracking user-metrics; and (c) approval workflows. Similarly, we are iterating Semantic Procedure Analyzer to include more heuristics to parse and chunk a SOP. This work is still ongoing, and we plan to generate a new version of platforms to infer more refined design principles.

4 Conclusion

In this paper, we argued that applying modularity principles toward the management and performance of complex work might be well-suited only in a context where the product and process of work is tangible and distinct (Okhuysen and Bechky 2009, p.468). In cases where the product and process of work is not all that tangible and distinct, a straightforward application of the logic of modularity would lead to counterproductive results. We then proposed a pragmatic approach to design the support of complex organizational work. Such an approach embraces a logic of chunking and incompleteness, rather than the logic of decomposition and completeness, and is premised upon a “layered modular” (Yoo, et al., 2011) as opposed to a “modular” architecture.

Drawing upon the canonical design research approach influenced by action design research (Sein et al 2011) and aided by empirical field work, we built two platforms – Semantic Procedure Analyzer (SPA) and ReKon – that illustrated and refined three foundational principles: a) Knowledge needed to navigate the complexities of work is often tacit, but is also materially anchored in concrete artifacts that become ‘starting points’ for actors to perform their work; b) Chunking and digitizing these artifacts, and making them available to actors as ‘knowledge chunks’ via a platform, enables these actors to handle complexity in real time and in an ongoing fashion; c) The resultant platform needs to be open, incomplete and underspecified, so that it could generate digital options in real time that could be recombined and reused over time.

Formative evaluation pointed to prima facie potential usefulness of the respective platforms. Ongoing work is aimed at improving the platforms through semi-structured interviews with practitioners, design iteration of the platforms, and a summative evaluation in the field.

Acknowledgements

The work reported has been funded by the National Science Foundation under award numbers 722112 and 722141. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF). A part of the work has also been funded by the Center for Operator Performance at Wright State University. We acknowledge contributions of project templates by and discussions with consulting organizations.

References


Carroll, J. (2002). Making use is more than a matter of task analysis. *Interacting with Computers, 14*(5), 619-627.


