International Journal of Cooperative Information Systems
Vol. 22, No. 4 (2013) 1341002 (21 pages)
© World Scientific Publishing Company
DOI: 10.1142/S0218843013410025

COLLECTIVE PROBLEM SOLVING USING SOCIAL COMPUTE UNITS

BIKRAM SENGUPTA*, ANSHU JAIN† and KAMAL BHATTACHARYA‡
IBM Research, Bangalore, India
**bsengupt@in.ibm.com*
†anshu.jain@in.ibm.com
‡kambhatt@in.ibm.com

HONG-LINH TRUONG§ and SCHAHRAM DUSTDAR¶
Distributed Systems Group
Vienna University of Technology, Austria
§truong@dsg.tuwien.ac.at
¶dustdar@dsg.tuwien.ac.at

Received 10 March 2013
Accepted 28 October 2013
Published 6 January 2014

Service process orchestration using workflow technologies has led to significant improvements in generating predicable outcomes by automating tedious manual tasks but suffer from challenges related to the flexibility required in work especially when humans are involved. Recently emerging trends in enterprises to explore social computing concepts have realized value in more agile work process orchestrations but tend to be less predictable with respect to outcomes. In this paper, we use IT services management, specifically, incident management for large scale systems, to investigate the interplay of workflow systems and social computing. We apply a recently introduced concept of social compute units (SCU), and flexible teams sourced based on various parameters such as skills, availability, incident urgency, etc. in the context of resolution of incidents in an IT service provider organization. Results from simulation-based experiments indicate that the combination of SCUs and workflow based processes can lead to significant improvement in key service delivery outcomes, with average resolution time per incident and number of SLO violations being at times as low as 53.7% and 38.1%, respectively of the corresponding values for pure workflow based incident management. Moreover, significant benefits may also be obtained through cross-skilling of practitioners via exposure to new skills in the context of collaborative work.

Keywords: Social compute units; business process management; workflows; incident management.

1. Introduction

Business process management (BPM) and workflow systems have had tremendous success in the past two decades with respect to both mindshare and deployment. We can safely consider service-oriented architecture (SOA) to be a business-as-usual design practice. On the other hand, we are observing enterprises embracing social computing as an alternative for executing more unstructured yet team-based collaborative, outcome-based strategies. Gartner[1] predicts that by 2015, we will observe a deeper penetration of social computing for the business as enterprises struggle to deal with the rigidity of business process techniques. Current workflows are suitable for automation of menial tasks but inflexible when it comes to supporting business users who must deal with complex decision making. However, a significant conceptual gap clearly exists between workflows on the one hand, and social computing as it is known today.

The goal of this paper is to introduce a framework that establishes the interactions of business processes with a concept called social compute units (SCU),[2] recently introduced by some of the authors. An SCU is an abstraction of a team consisting of human resources that bring together the appropriate expertise to solve a given problem. The SCU abstraction treats the SCU as a programmable entity. The resources that make up an SCU are socially connected. The term *socially* implies connectedness of an individual beyond his or her organizational scope. The reason for connectedness could be prior ad-hoc collaborations but also collaboration within a given scope of responsibility where the scope is distributed across organizational verticals.

The context in which we propose the use of SCUs in this paper is the IT service management (ITSM) domain. More specifically, we are interested in the incident management process within ITSM. IT service vendors maintain large, complex IT infrastructure on behalf of their clients, and set up skill-based teams with the responsibility of maintaining different infrastructure components, such as applications, servers, databases, and so on. When an interruption or degradation in service in some part of the IT infrastructure is detected, service requests are raised in the form of *tickets* that describe the incident. However, due to inherent dependencies between different system components, identifying the root cause of the problem is a complex, and often time-consuming, activity. The traditional approach to incident management is to have a human dispatcher intercept the ticket and review the incident description. Using his/her knowledge of the system and dependencies, the dispatcher then determines the most likely component that may be faulty and forwards the ticket to the relevant team, where it is assigned to a practitioner for investigation. The practitioner may determine the presence of a fault in the component and the incident may be resolved by taking corrective action to remove the fault. However, often the practitioner may discover that the fault does not lie in the component she/he is managing, and sends the ticket back to the dispatcher, who then needs to decide on the next team the ticket should be sent to, and this process continues till the right team receives the ticket and resolves the incident.

Such a process-driven approach may be reasonable when the problem description is detailed and clear. In reality, we find the end user reporting the incident to state the symptom at best. It is the human dispatcher's responsibility to guess the root cause and identify the right person for the resolution job. This may be appropriate for simple and low severity incidents, but is risky in more complex situations. In those cases the overall resolution time may exceed the contractually agreed upon response time as manifested in service level objectives (SLO). The consequences can be degradation of client satisfaction and/or monetary penalties. Our proposal is to demonstrate the benefits of bringing together appropriate resolution units, conceptualized as SCUs, that possess the right skills composition to deal with the eventualities of the given context, as defined by the system where the incident occurs. The members of an SCU may be drawn from components where there is a higher likelihood of a fault, and component dependency information may be used to on/off-board members as investigation proceeds. Such an agile way of managing incident resolution should help to facilitate parallel investigations and thereby quicker resolution. However, SCUs may also incur a higher cost (since multiple practitioners are investigating a problem at once), hence its use has to be judiciously interleaved with the more standard workflow-driven sequential investigation of incidents, so that the right trade-off between quicker resolution and higher cost may be achieved.

Besides improving operational parameters such as resolution time and SLO adherence, an SCU also provides a collaboration platform for practitioners having complementary skills to come together for root cause analysis and problem resolution. This provides a great opportunity for skills transfer — for a practitioner to learn the features of a new component over a period of time by participating in joint problem resolution with peers who specialize in those components. Such on-the-job learning can hasten the acquisition of new skills and be a very valuable addition to any formal learning programs that an organization arranges for employee training. The main contributions of the paper areas follows:

(1) The development of a technical framework for SCU sourcing, invocation and evolution in the context of IT incident management spanning multiple teams and organizational verticals.
(2) A simulation based approach to (i) study the efficiencies that may be gained through SCUs over standard process management approaches, and the trade-offs involved (ii) understand the opportunities SCUs create for cross-skilling of practitioners based on collaborative resolution of problems.

The rest of the paper is organized as follows. In Sec. 2, we present a motivating example and introduce the concept of SCUs. In Sec. 3, we introduce the formal system model for the use of SCUs in incident management, and describe the life-cycle of an SCU from its invocation, evolution to dissolution. Section 4 presents a simulation based method to demonstrate the benefits that may be achieved through a combination of SCUs and workflow based approaches. After discussing related work in Sec. 5, we conclude with a discussion on future work and extensions of our framework in Sec. 6.

2. Motivating Example

Consider an IT service provider that manages applications on behalf of a client. Each application is a complex ecosystem of its own, consisting of actual application code, the hosting middleware and operating system, servers, storage components, network and firewall configurations, etc. An incident in any application may have its root cause in any of its sub-systems. Resolving the incident for the application may henceforth require multiple skills, from programming skills to networking skills. IT service providers that manage hundreds or thousands of applications cannot scale by assigning individual teams to manage individual applications. Instead, it is more cost-efficient to form organizational units that are formed around skills and manage specific, similar system components.

Figure 1 shows an example of a component dependency graph for an application and its management context. The connectors between nodes indicate the nature of relationship or dependency between the components. For example, the straight line between *application* and *application server* denotes a tight coupling as in a component being hosted on another (thus the dependency type is *isHostedOn*). Each dotted line, e.g. between *web server* and *application server*, denotes a loose coupling between components as in one component being connected to the other through a web-service call or an HTTP request (*isConnectedTo* being the dependency type).

Each layer of an application is managed by an organizational entity as denoted on the left hand side. The *application* component is managed by the application management team, which has the right set of coding skills and application knowledge to debug issues or extend functionality. The middleware layer (web server, application server and database management system (DBMS)) is managed by the application hosting team that knows how to manage application servers and databases. In principle each management entity can be modeled as another node in

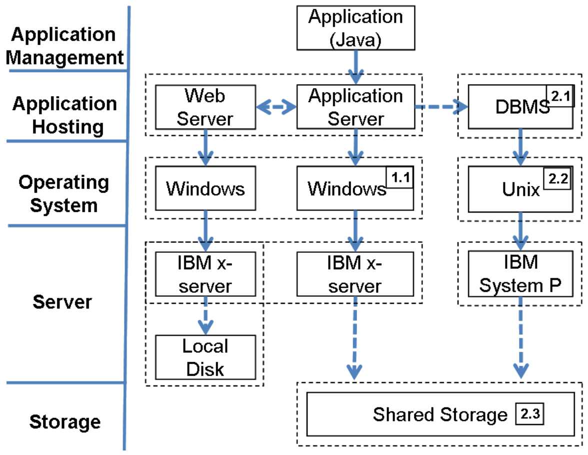


Fig. 1. Dependencies between application components and their management context.

the dependency graph but for simplicity we have depicted management entities as dotted line boxes around the components they are responsible for. For example, the dotted line box around the DBMS component could also be represented by a DBMS node connected to a database management node through a connector stereotyped as *isManagedBy*.

Let us now consider incident management — an important area in ITSM[3] — for the above example. An incident is an event which is not a part of the standard operation of a service and causes, or may cause, an interruption to or a reduction in, the quality of that service. The goal of incident management is to restore the service functionality as quickly as required. The service restoration time is tied to the SLO the client and provider have agreed upon and usually depends on the severity of the incident. An incident may be caused by a known problem for which there exists an action plan, or, may also require problem resolution as the root cause is not known. Let us now examine incident resolution through two scenarios, with a focus on who is resolving an issue.

(1) An incident states an issue with capacity exceeded on the server hosting the application server. The action plan (1.1 in Fig. 1) is to delete all unnecessary temporary files using a pre-defined sequence of steps. The resource required is a member of the operating system team and has Windows skills.
(2) An incident indicates an issue with slow performance of the DBMS. The root cause needs to be identified. The IT help desk responsible for dispatching the ticket will route the ticket to the Application hosting team and a resource will be assigned to look for standard causes, such as full log files (2.1 in Fig. 1). The assigned resource determines that the cause is capacity exceeded on the server and hence passes the incident on to the server support to free up capacity (2.2 in Fig. 1). The server support team member executes some pre-defined steps to free up space, however notices that this is not solving the issue as the problem lies in the growth of the tables. This requires the ticket to be passed on to the storage team (2.3 in Fig. 1). The storage team allocates space and takes necessary steps to resolve the issue and close the incident.

In the first scenario, the incident description has sufficient clarity for a dispatcher to quickly identify the relevant component and required skills. The ticket can be dispatched to the Windows operating system team, where an available practitioner will execute a standard set of actions to resolve the issue. Thus, such a ticket is amenable to a well-defined, repeatable incident management process. On the other hand, the second case involves a ticket where the symptom reported can be due to one of several possible causes. Using the standard sequential approach to incident management, we can try to proceed one component at a time, ruling out the presence of a fault in one component, before passing on the ticket to the next likely component. However, by the time the actual problem is discovered and fixed, significant time may have elapsed and an SCU may also have been breached. It may be noted that the time spent is not only due to the investigations that have to be carried out by each team, but also due to the delays that occur when a ticket has to be transferred across organizational boundaries, and the time that is wasted when a ticket is pending in a queue, waiting for a practitioner to become available and accept the ticket.

We postulate that tickets such as in the second case above, need a different approach to incident management. Instead of being investigated by one practitioner from one component team at a time, they may need simultaneous attention from multiple practitioners and teams. In this paper we will apply the concept of an SCU[2] to address the second scenario above. An SCU is a "loosely coupled, virtual and nimble" team of socially-connected experts with skills in domains relevant to the problem at hand. Socially connected in common terms is widely understood outside the work context. We define connectedness with respect to a *social fabric*. We believe that in general a human resource is a connected entity represented as a point (or node) in the social fabric. The social fabric consists of a continuum of network expressions such as an organizational network or a collaboration network. The former is typically well defined with specific roles and responsibilities assigned to nodes in the network. The latter is an expression of an individual's ability to transcend organizational boundaries to act as a source or sink of information.

An SCU team member may not be a dedicated resource, but someone who can invest a certain amount of time toward solving a problem, when the requirement comes up. An SCU is created on request (e.g. from the problem domain's business owner), has a certain amount of computing power derived from the skills and availability of its members and from their collaboration, uses its computing power toward addressing the problem which triggered its creation, and is dissolved upon problem resolution. These characteristics make the SCU an attractive approach for incident resolution management in an IT service organization. For more details on the SCU concept, interested readers are referred to Ref. 2. In the rest of the paper, we will describe in detail how SCUs may be used within an IT service management environment for facilitating collaborative incident resolution spanning multiple teams/organizations.

3. Incident Management Using SCUs: A Technical Framework

We first describe our system model, and then outline the basic principles that guide an SCU through its life-cycle. There may be different ways of realizing the abstract framework presented in this section and a concrete instantiation will be described and evaluated in the following section.

3.1. *System model*

A visual UML representation of the key concepts and relationships in our system model is depicted in Fig. 2 and these are explained in detail in what follows. We assume a component dependency model represented as a graph $G = (V, E)$, where

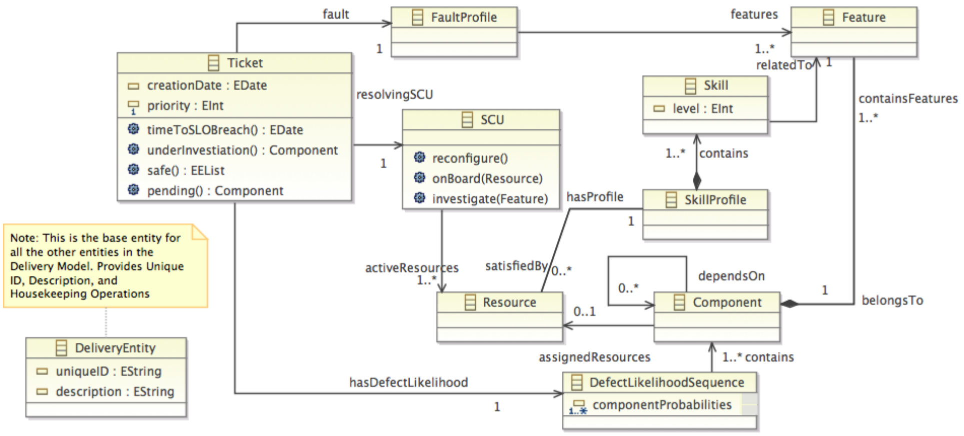


Fig. 2. System model.

V is a set of nodes, each representing a component of the overall IT infrastructure being managed, and E is a set of directed edges representing dependency relationships between the nodes. Each edge may be represented by a triple $\langle C_i, C_j, DT_k\rangle$, where $C_i, C_j \in V$ and represent system components, and $DT_k \in DT$ is the dependency relationship that exists between C_i (source) and C_j (target), drawn from a set DT of possible relationships relevant to the domain. A component $C_i \in V$ has a set of possible *features* C_i^F through which it provides services to the end-user. A *ticket* T_i is raised when an interruption or degradation in IT performance is detected, and may be initially represented as $\langle TS_i, D_i, P_i\rangle$, where TS_i is the time-stamp when the ticket was raised, D_i is a textual description of the problem, P_i is a priority level indicating the criticality of the problem that may be drawn from an ordered sequence of numbers $\{1, 2, \ldots, p\}$, with 1 representing the highest priority and p the lowest. Based on its priority P_i, a ticket will also have a service level objective SLO(P_i) which is the window of time W_i within which the incident needs to be resolved, thereby setting a deadline $TS_i + W_i$ that the ITSM team has to strive to meet. Each ticket is also implicitly associated with a fault profile $\text{FP} = \{(C_i, f_j) \,|\, C_i \in V, f \in C_i^F\}$, which is a set of features in a set of system components that have developed a fault, and need to be investigated and fixed. In the majority of cases, we may expect a ticket to be associated with a single malfunctioning feature in one component. Of course, the fault profile is not known when a ticket arrives, but needs to be discovered in course of the incident management process.

When a ticket is raised, it is the dispatcher's responsibility to review the problem description, and try to identify the likely component(s) that are not functioning correctly, so that the ticket may be dispatched to the relevant team(s). The dispatcher is aided in this task by the component dependency graph, and in general, she/he may be expected to devise a dispatch strategy that contains an ordered sequence of components $\{C_{i_1}, C_{i_2}, \ldots, C_{i_{|V|}}\}$, which we call the *likelihood sequence* ($\mathrm{LS}(t)$) for a ticket t, going from the most to the least likely component that may be the cause for this incident. $\mathrm{LS}(t)$ represents the order in which the ticket should be dispatched to the different components for investigation, till the root cause is identified and fixed. Note that we assume a fully ordered sequence only for simplicity; the likelihood of some of the elements may be deemed to be equal, in which case any of them may be picked as the next component to be investigated. $\mathrm{LS}(t)$ would depend on the ticket description, which may contain some indicators of the potential source of the problem, while the component dependency graph would make certain related components candidates for investigation as well, based on the nature of the relationship they have with the sources. The likelihood sequence is a key construct in our incident management framework. It drives the dispatch strategy for the business-as-usual (BAU) way of routing tickets one team at a time. The SCU-based approach also uses the likelihood sequence to decide on the composition and dynamic reconfiguration of an SCU during an incident resolution activity. The likelihood sequence may be expected to evolve as investigation proceeds and better understanding is achieved regarding the nature of the problem.

ITSM services will be provided by a set of practitioners (human resources) R. Each $R_i \in R$ has a skill profile $\mathrm{SP}(R_i) = \{(C_p, f_q, l_r) \,|\, C_p \in V, f_q \in C_p^F, l_r \in L\}$, where L is an ordered sequence of skill levels $\{1, 2, \ldots, l\}$, with 1 being the lowest (most basic) and l being the highest skill level for any skill. A skill is the ability to investigate and correct a particular feature in a given component. Most practitioners will possess a set of skills, with varying skill levels, and the skills of a practitioner will usually be centered around different features in a single IT component, although there may be a few experienced practitioners with skills that span multiple components. Given this, each component is immediately associated with a *team* — a group of practitioners who have skills in one or more of the component features and may be called upon to investigate an incident that may have potentially been caused by the component. For a component C_i, this is given by $\mathrm{Team}(C_i) = \{R_i \,|\, R_i \in R, \exists\, l \in L, \exists\, f \in C_i^F \cdot (C_i, f, l) \in SP(R_i)\}$. For each combination (f, l) of a feature f and skill level l, we have an effort tuple $\langle E_{\mathrm{inv}}(f, l), E_{\mathrm{res}}(f, l)\rangle$, which indicates representative (e.g. average) effort needed by a practitioner with skill level l in feature f to investigate if the feature is working correctly ($E_{\mathrm{inv}}(f, l)$), and to restore the feature to working condition ($E_{\mathrm{res}}(f, l)$) in case a fault is detected.

A social compute unit $\mathrm{SCU}(T_p, t)$ for a ticket T_p at a point in time t, can be represented by $\langle \mathcal{C}(t), \mathcal{R}(t), \mathcal{S}(t)\rangle$, where $\mathcal{C}(t) \subseteq V$ is the set of components currently being investigated (i.e. at time t), $\mathcal{R}(t) \subseteq R$ is the set of practitioners that are part of the SCU at time t, and $\mathcal{S}(t)$ is an abstraction of the current *state* of the investigation, encompassing all components/features that have been verified to be functioning correctly till this point, all the faulty features that have been restored, and all the features that are currently under investigation or restoration. Thus, an SCU is a dynamic entity whose composition (in terms of components and practitioners) as well as state (in terms of features investigated or restored), will continually evolve with time.

The management of an incident — from its creation to closure — will incur a cost, primarily due to effort spent by practitioners on various investigation tasks. We assume that when a practitioner joins an investigation effort, she/he will need to spend a certain amount of effort E_{init} to familiarize with the problem and the current investigation state. Subsequently, she/he will expend effort on feature investigation and restoration, commensurate with her skill levels. If she/he is part of a SCU, she will be expected to spend E_{collab} effort to collaborate with the larger team through the sharing of findings. This effort may be proportional to the duration of her stay in the SCU, as well as the size of the SCU during that period (in terms of the number of practitioners and components/features involved). If we wish to monetize the effort spent, we may assume a function $\text{UnitSalary}(R_i) : \text{SP}(R_i) \rightarrow \mathcal{R}$ (where $\mathcal{R}$ is the set of real numbers), that returns the cost of unit effort spent by the practitioner, based on his/her skill profile. We also assume that there is a certain amount of effort needed to set up and dissolve an SCU, given by $\text{SCU}_{\text{setup}}$ and $\text{SCU}_{\text{disolve}}$, respectively. Also, in the process-driven sequential way of incident management, there will be a certain delay D_{transfer} imposed each time a ticket is transferred from one team to the next. Finally, delays may be introduced due to unavailability of practitioners.

3.2. *SCU based incident management*

Figure 3 depicts the overall incident management process that we propose. It is important to note that our SCU approach complements, rather than replaces, sequential process-driven incident management, represented in Fig. 3 as the BAU flow. We do not expect an SCU to be required for every ticket, rather we base this decision upon the specific context of a ticket at a given point in time. When an incident arrives, the problem description should be reviewed, and the relative likelihood of different components being the source of the problem, has to be evaluated. This may be done by a human agent (e.g. a dispatcher) who uses a combination of ticket description and knowledge of component dependencies to identify potential faulty components. Supervised learning techniques such as support vector machine (SVM)[4] may also be used to suggest for new tickets, the likelihood (represented by a probability distribution) of each component being the source of the problem.[5] A combination involving a human dispatcher being assisted by an automated agent is also possible. It may be noted that such a likelihood evaluation is anyway done (even if implicitly) as part of a standard incident management process, since the dispatcher has to decide each time she/he receives a ticket, which component team needs to be contacted next to investigate the problem. Also, it is not necessary for the entire likelihood sequence to be generated as soon as an incident arrives. Instead, this may also be done incrementally, by considering at any point in time which are the most likely components that may be the cause for the incident (taking into

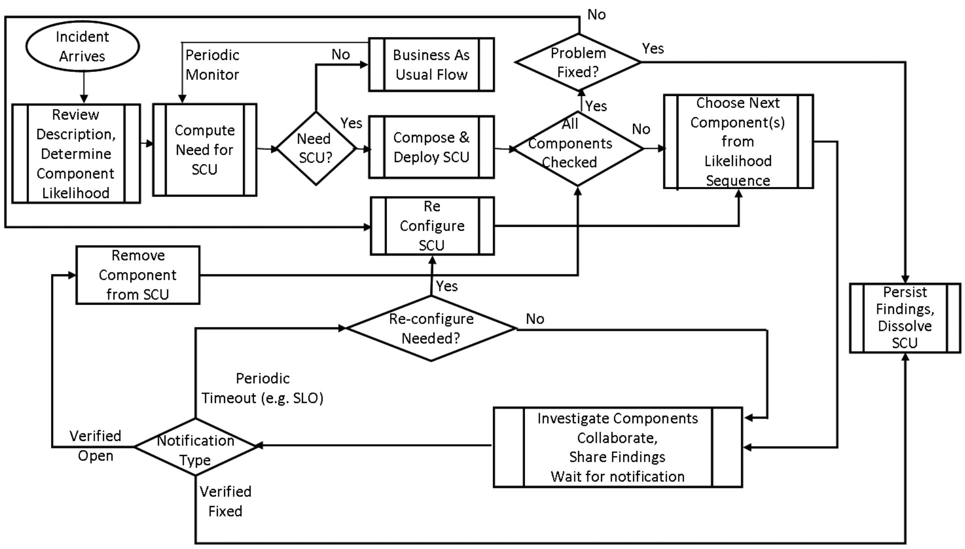

Incident Arrives
Review Description, Determine Component Likelihood
Periodic Monitor
Compute Need for SCU
Need SCU?
No
Yes
Business As Usual Flow
Compose & Deploy SCU
No
Problem Fixed?
Yes
Yes
All Components Checked
No
Choose Next Component(s) from Likelihood Sequence
Re Configure SCU
Remove Component from SCU
Yes
Re-configure Needed?
No
Persist Findings, Dissolve SCU
Periodic Timeout (e.g. SLO)
Verified Open
Notification Type
Verified Fixed
Investigate Components Collaborate, Share Findings Wait for notification

Fig. 3. Incident management using SCUs.

account components that have already been investigated), and involve only those teams in the next phase of investigation.

Once this initial analysis has been done, we need to decide whether or not to invoke an SCU. In case the ticket deadline is sufficiently far away and/or there is a very strong likelihood of one particular component being the source of the problem, then the system may decide to follow the BAU mode, in which the ticket is dispatched to the most likely faulty component, where a practitioner will have to investigate it (this is explained in more detail later in the context of Fig. 4). However, there will still be a need to monitor the situation so that in case the

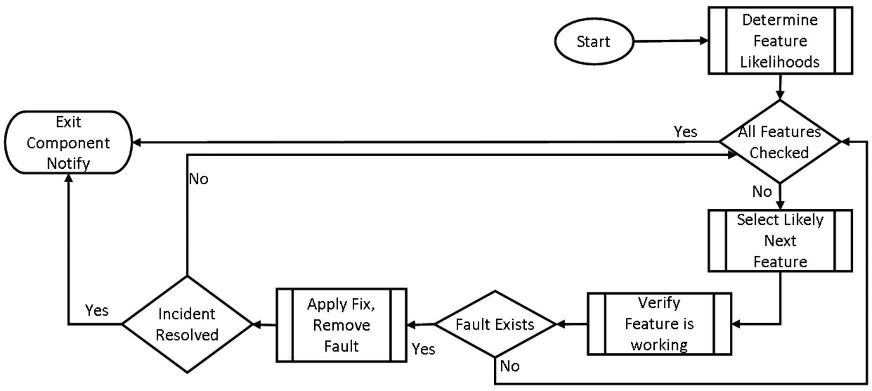


Fig. 4. Investigation within a component.

deadline is approaching without the root cause been detected, then a decision may be taken to set up an SCU to accelerate the investigation.

In case the BAU mode is not deemed appropriate in view of an impending deadline or lack of clarity in the problem description, an SCU may be invoked. Here, a few of the more likely faulty components are identified, and a set of practitioners who have the necessary skills in these components are on-boarded to the SCU. The decision on how many such components to consider, how many practitioners to on-board, what their skill levels should be, etc. may be taken based on the availability and the urgency of the situation. For example, if there is a crisis situation with the deadline of a high priority ticket being near at hand, then highly skilled practitioners from most/all of the component teams may have to be involved. In less urgent cases, one practitioner per component, for a small number of components at a time (say, two to three) may be sufficient. Once on-boarded, the practitioners representing a component would try to determine if the incident has been caused by a fault in one of its features, using the process depicted in Fig. 4. Here, a practitioner would proceed through the feature list of a component in the order of their relative likelihood of having a fault (as inferred by him/her based on the ticket description and understanding of features), and for each feature, investigate if it is correctly functioning, and if not, apply a fix to restore the feature. If the fix resolves the incident as well, then any ongoing investigation will be stopped across all components. Suitable findings from the investigation process will be harvested for later reference, and the SCU will be dissolved (Fig. 3). Otherwise, the practitioner may move on to the next feature. If the current practitioner(s) representing a component do not have all the skills needed to cover the complete feature set, then on completion of their investigation, they may be replaced by other suitable practitioners. Note that this basic approach toward investigating a component remains the same whether a BAU or SCU mode is used. In case multiple practitioners are investigating the same component together in an SCU, they may partition the feature list amongst themselves to ensure that there is no redundancy of effort. Also, within an SCU, a practitioner will be expected to collaborate with others, e.g. through the sharing of findings, as shown in Fig. 3.

As investigation proceeds, it is necessary to periodically monitor the situation and take appropriate action (Fig. 3). For example, if the incident deadline is approaching, then there would be a need to re-configure the SCU by on-boarding more practitioners to cover other components. In case a high priority ticket arrives that needs the attention of a practitioner who is currently part of another (relatively less urgent) ticket's SCU, then the practitioner may have to leave the SCU and should be replaced by another suitably skilled colleague who is available. Again, if all the features of a component have been verified to be functioning correctly, then the component may be removed from the SCU and the corresponding practitioners off-boarded. New SCU members may then be added from the next likely set of component(s). Finally, in the unusual case when all components have been investigated without the defect being identified, the SCU may be re-constituted and re-run, with higher skilled practitioners if needed.

4. Experiments

To experimentally evaluate our proposed SCU-based approach for incident management, we have designed an event driven simulator that mimics the flow of tickets through an IT service delivery environment. Broadly, there are two classes of results we are interested in. The first relates to the impact of the SCU-based approach on the operational performance of the service delivery system, for example, the time taken to resolve an incident, the number of SLO violations, etc. We study such performance characteristics in detail, comparing the results obtained through the process-driven sequential BAU mode with those obtained through a combination of BAU and SCUs. We also experiment with different SCU constructs during this study, based on varying the number of components that have at least one practitioner on-boarded to the SCU at a time. These results are discussed in Sec. 4.2. Then we further investigate the impact of varying the dispatching accuracy on these results, as discussed in Sec. 4.3.

The second category of results that are of interest relates to the implicit cross-skilling that can occur in an SCU. This can occur by virtue of the fact that practitioners with different expertise profiles (e.g. involving different system components) come together in an SCU to work on a common problem, jointly reviewing possible root causes, approach taken, and results. In the process, a practitioner can gain valuable knowledge about related system components that may not be within his/her direct expertise. While the theoretical development of a detailed skill transfer model is beyond the scope of this paper, here we are interested in studying the extent to which the SCU approach provides avenues for such cross-skilling by enabling collaboration opportunities between practitioners having complementary skills. These results are discussed in Sec. 4.4.

We first describe the experimental setup in Sec. 4.1, before presenting the results.

4.1. *Experimental setup*

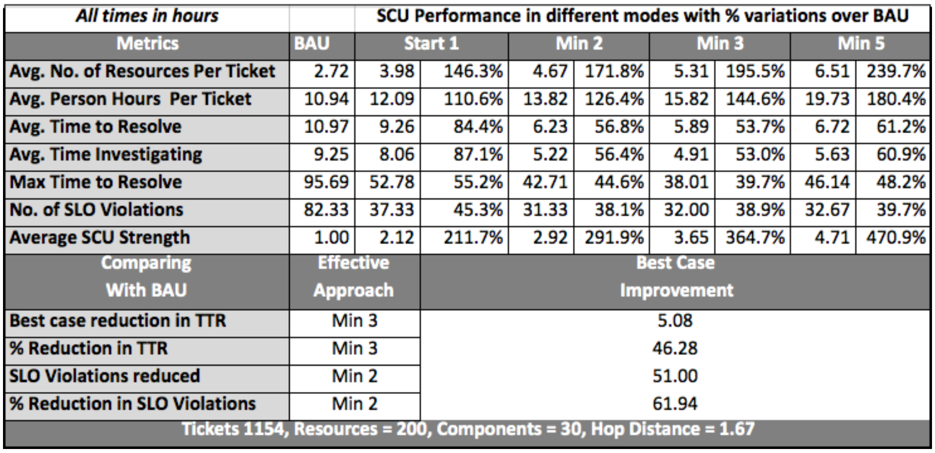
The simulation framework is built on Java and has four major components: events generator that generates standard events related to incidence creation and management; ticket lifecycle simulator, which manages various timers and notifications related to a ticket; delivery model, which includes the basic models of all the system entities (tickets, components, features, resources, etc.) and relationships, whose generated runtime is directly used within the simulation framework; and SCU runtime manager, designed as a library for implementing an SCU model in a service delivery environment.

For our experiments, we have considered an IT system with 30 components, with each component having between 0 to 5 dependencies generated as a random graph. For generating ticket data, we used a power-law probability distribution of tickets across components, which is suitable for generating Pareto-like long tailed distributions. Based on our experience from working with large ITSM organizations, we have set this closer to a 70:30 distribution, which means that only 30% of the components cause 70% of the tickets. Overall, we generated 1154 tickets to cover a one week period of study, and maintained a resource pool of 200 practitioners to ensure a reasonable service rate. The ticket arrival rate is modeled as a stochastic process using a Poisson distribution initialized by average hourly arrival rates of tickets as we have seen in several actual service delivery environments. We assumed four different priority levels for tickets, with SLOs of 4 h (highest priority), 8 h, 24 h and 48 h (lowest priority), respectively. The relative distribution of the priority levels, were 2% (highest priority), 8%, 20% and 70% (lowest probability). All these values were selected based on our review of multiple ticket sets and SLOs. We assumed each practitioner to have skills in all the features of one component (which is often the case since practitioners in such environments are usually specialists in a particular technical domain). The staffing levels of each component were determined based on their relative workload (in terms of number of tickets received, the number of features, and the effort needed to investigate and fix each feature). Each ticket was assigned a fault profile of a single feature in one component.

We studied two modes of incident management — a fully process driven BAU mode, and a heterogeneous mode of BAU and SCUs. In the former, a ticket is investigated by one practitioner from one component at a time, and whenever a ticket has to cross organization boundaries, we assumed a delay of 30 min to account for the process-related overheads. This is actually a conservative estimate, since in real-life service environments we have found tickets to be stuck for hours or days together in transfer between the components, and this was a key motivation for the SCU. In the heterogeneous mode, SCUs were automatically assembled for every highest priority ticket. For the rest of the tickets (including those initially dispatched in BAU mode), the decision to compose an SCU was based on the urgency of the situation at a given point in time. We used four levels of urgency, based on distance from the SLO deadline, and gradually increased the span of an SCU to cover more components (while having a single practitioner per SCU component) as the ticket moved from one urgency level to the next higher one.

4.2. *Resolution effort, time and SLO performance*

The table in Fig. 5 presents the results obtained from our first set of simulation-based experiments. The column BAU stands for the mode where only process-driven sequential investigation was carried out for each ticket, while the rest of the columns involve situations where the BAU mode was complemented by SCUs. We experimented with different variations of this latter mode. We started with a conservative policy of initializing each SCU with a single component (Start 1), but still investing the SCU setup cost (e.g. for getting the collaboration platform ready)



All times in hours		SCU Performance in different modes with % variations over BAU							
Metrics	**BAU**	**Start 1**		**Min 2**		**Min 3**		**Min 5**	
Avg. No. of Resources Per Ticket	2.72	3.98	146.3%	4.67	171.8%	5.31	195.5%	6.51	239.7%
Avg. Person Hours Per Ticket	10.94	12.09	110.6%	13.82	126.4%	15.82	144.6%	19.73	180.4%
Avg. Time to Resolve	10.97	9.26	84.4%	6.23	56.8%	5.89	53.7%	6.72	61.2%
Avg. Time Investigating	9.25	8.06	87.1%	5.22	56.4%	4.91	53.0%	5.63	60.9%
Max Time to Resolve	95.69	52.78	55.2%	42.71	44.6%	38.01	39.7%	46.14	48.2%
No. of SLO Violations	82.33	37.33	45.3%	31.33	38.1%	32.00	38.9%	32.67	39.7%
Average SCU Strength	1.00	2.12	211.7%	2.92	291.9%	3.65	364.7%	4.71	470.9%
Comparing With BAU	**Effective Approach**	**Best Case Improvement**							
Best case reduction in TTR	Min 3	5.08							
% Reduction in TTR	Min 3	46.28							
SLO Violations reduced	Min 2	51.00							
% Reduction in SLO Violations	Min 2	61.94							
Tickets 1154, Resources = 200, Components = 30, Hop Distance = 1.67									

Fig. 5. Experimental results: resolution time, effort and SLO violations.

in anticipation of the need to on-board more practitioners. In the other variations, we initialized each SCU with two, three and five components. Once set up, an SCU was, of course, allowed to expand in size by on-boarding practitioners from more components, as the ticket progressed toward its deadline. In each of these experiments, we assumed a highly skilled dispatcher. To model this, the likelihood sequence of tickets was generated carefully by ensuring that the faulty component is amongst the most likely ones in a high percentage of cases — using a Pareto distribution of 80:20, corresponding to a Pareto index of 1.16 (for 80% of tickets, the faulty component will occur within the top 20% of elements in the sequence). We also generated tickets with unclear root causes, where the faulty component occurred later in the sequence (with a probability that decreased progressively as the likelihood decreased). Moreover, we adjusted each sequence to ensure that the position of a likely component correlated well with that of some of its neighbors in the dependency graph, so that these neighbors were likely candidates as well. Overall, the average distance of the faulty component from the start of the sequence over all tickets was only 1.67, reflecting a skilled dispatch performance.

We compare the performance of BAU and SCU modes along two main dimensions: effort and time to resolve (TTR). In terms of effort spent, the BAU mode is, in general, more efficient than the SCU mode. This is because in the former, only a single practitioner is being assigned at a time to conduct an investigation (on the most likely component at that point), while in the latter, multiple practitioners are assigned, and the aggregate effort invested is likely to be higher. Thus, both the metrics average number of resources per ticket and average person hours effort per ticket (aggregated across all resources who worked on a ticket) show an increase as we go from BAU to SCU mode, and across the different variants of SCU modes.

While the overall effort spent in SCU mode is, in general, higher, the collaborative investigation power of an SCU also significantly reduces the time to resolution, as seen from the values of the metrics average TTR, average time investigating and max TTR. In all of these, the performance of the BAU mode lags far behind that of the SCU modes. From the business impact perspective, the most compelling case for the SCU comes from the dramatically improved SLO performance that results from its faster resolution of tickets, with number of SLO violations ranging between only 38.1% and 45.3% of the corresponding number for BAU. With the stringent penalties that IT vendors have to pay for poor SLA performance, the financial implications of this are far reaching. It may also be noted that while the SCU approach may consume more aggregate effort from practitioners, this does not necessarily translate to higher costs for the vendor. This is because, vendors typically maintain a dedicated team to provide production support to customer systems, and the effort available from these practitioners, if not utilized, may go waste and result in under-utilization, even though the vendor would still have to bear the same costs in terms of employee salary.

While an SCU has the flexibility to scale up as needed, we find that the average SCU size at any point in time (or its "strength") ranges from 2.12 to 4.71. While this may also partly be due to resource unavailability that prevents it from growing very large (since there will be many other tickets that keep practitioners engaged), the size is small enough for easy governance.

4.3. *Impact of dispatching accuracy*

As mentioned in Sec. 4.2, we have so far assumed that the dispatcher (who creates the likelihood sequence) is skilled at identifying the component(s) that are potentially at fault. While this may be true for experienced dispatchers in long-running environments, this need not be the case in all service delivery settings. For example, when an IT service vendor assumes responsibility for managing a customer's IT environment, knowledge about the system components and dependencies is likely to be incomplete and inaccurate to begin with. Moreover, IT environments often undergo significant transformations as new components are introduced, and older ones are withdrawn, and the component dependency graph thereby undergoes many changes. In these situations, a dispatcher is likely to be much less skilled in identifying the components at fault, based on his/her review of symptoms reported in the tickets. The faulty component for a ticket may then appear much later in the likelihood sequence drawn up by the dispatcher. As a result, the ticket would be incorrectly routed, hopping across several teams before arriving at the correct one and being resolved. Obviously, this significantly increases the overall time taken to resolve such incidents, and thereby many more SLOs are likely to be violated.

To study this phenomenon, we experiment with different levels of dispatcher accuracy. We model accuracy as the factor $N:M$, where N represents the percentage of tickets for which the dispatcher is able to position the faulty component within the first $M\%$ of elements in the likelihood sequence. The impact of dispatching accuracy on the average defect distance (from the start of the likelihood

Accuracy Percentage	Avg. Defect Distance	BAU TTR in Hours	BAU SLO Violations
100	0.93	8.23	54
90	2.22	13.67	108
80	3.45	20.57	174
70	4.75	40.97	349
60	6.03	76.27	532
50	6.80	100.53	594

Fig. 6. Impact of dispatch inaccuracy.

sequence at position 0), as well as on the average TTR and number of SLO violations for the BAU mode are shown in Fig. 6. As can be seen, decreasing dispatch accuracy leads to a significant deterioration in BAU performance, both in terms of TTR incidents, as well as the number of SLOs that are violated.

Figure 7 depicts the reduction in TTR (along y-axis) that may be achieved over BAU, using different SCU strategies (along x-axis) for different dispatcher accuracy levels (the line diagrams). As can be seen, as dispatcher accuracy decreases, the use of SCU-based strategies leads to increased savings in (absolute) resolution time of incidents, with substantial benefits obtained at low levels of accuracy (60% and 50%). At the same time, the figure also shows that given an estimated dispatcher accuracy, the savings vary by SCU strategy, and thereby the optimal strategy may be adopted once an estimate of the dispatcher's accuracy is available. Thus, Min 3 appears as an efficient strategy for most accuracy levels yielding significant TTR savings, while Min 5 can at times lead to decreased savings, possibly due to the

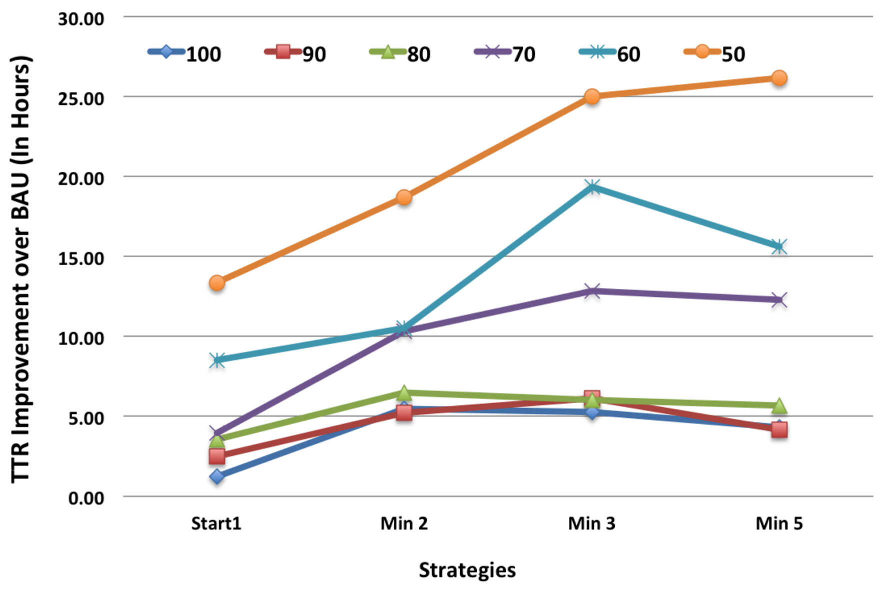


Fig. 7. TTR reduction by SCU strategies for different dispatch accuracies.

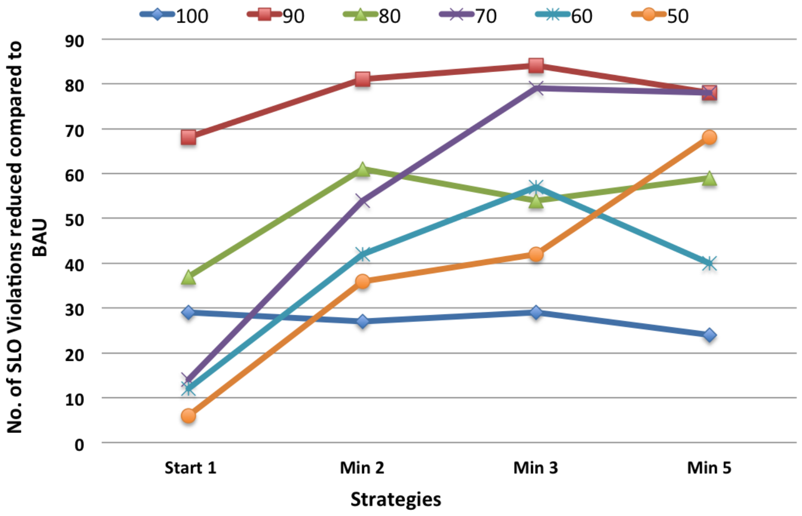


Fig. 8. SLO violation reduction by SCU strategies for different dispatch accuracies.

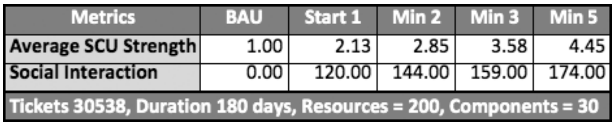
increased waiting time caused by unavailability of practitioners already engaged in large existing SCUs.

Figure 8 shows the reduction in number of SLO violations over BAU using different SCU strategies and for different accuracy levels. Like in the case of TTR, SCU-based strategies consistently outperform BAU in terms of the number of SLO violations, for any accuracy level. Unlike TTR, however, lower dispatch accuracy levels does not necessarily mean larger SLO savings over corresponding BAU values, since increased TTR associated with lower accuracy eventually causes many tickets to breach SLO even when an SCU mode is used.

The results demonstrate that SCUs continue to be a useful strategy for situations where the dispatcher skill levels are low. While TTR savings will consistently accrue with decreasing accuracy, continued reductions in SLO violations cannot be sustained beyond a point, although all SCU strategies will still outperform the BAU mode.

4.4. *Collaboration and cross-skilling*

An SCU brings together practitioners having complementary skills to collaborate on the common task of resolving a problem. During this collaboration, practitioners discuss possible causes of the incident, try fixes and share results, and in general work as a team, bound by a common purpose. This has at least two advantages. First, the social network of each practitioner increases or becomes stronger, as greater familiarity develops with peers, which can, over time, improve team cohesion and productivity. The second is that through the process of resolving a problem together, technical knowledge gets shared between the practitioners, resulting in an improved understanding of related components, which a practitioner may not be a



Metrics	BAU	Start 1	Min 2	Min 3	Min 5
Average SCU Strength	1.00	2.13	2.85	3.58	4.45
Social Interaction	0.00	120.00	144.00	159.00	174.00
Tickets 30538, Duration 180 days, Resources = 200, Components = 30					

Fig. 9. Social interactions.

specialist for. This, when repeated over a period of time, provides a natural avenue for cross-skilling practitioners through on-the-job learning via peer interactions, and can be a valuable complement to any formal, structured training programs the organization may have for its practitioners. Since these benefits accrue over a period of time, we conducted simulation experiments over 30538 tickets generated over a period of six months (instead of the seven day period used for the earlier experiments). It may be noted that these benefits are unique to the SCU mode of working, since the normal BAU mode does not provide for such interactions — thus no social connections are made through work, neither is any knowledge transferred.

In terms of social interactions, we measure the average number of (distinct) colleagues that any practitioner may have collaborated with, by virtue of being in the same SCU any time during the six months. Figure 9 depicts the results for the different SCU modes introduced earlier. On an average, a practitioner makes a surprisingly high number of connections (unique social interactions) within these six months: 120 for Start 1, and going up as high as 174 for Min 5 (there are a total of 200 practitioners). As may be expected, larger the number of practitioners an SCU is initiated with, higher is its average size (SCU strength), and higher the average number of connections that can be made. The results indicate that the SCU approach may be very helpful in making practitioners familiar with each other quickly, via the collaborative work they carry out.

An actual measure of cross-skilling through SCUs would require formal learning models that are beyond the scope of this paper, which we leave for future work. However, the basis of such learning would be time spent by two practitioners working together in an SCU, where each practitioner possesses a skill for a specific component being investigated, and the act of working together results in a (marginal) transfer of skills between the two, proportional to the time spent. Hence, here we focus on the exposure a practitioner receives to another component in terms of the time spent with an expert on that component in an SCU, and then aggregate this over all the SCUs the practitioner participates in over the period of study. Figure 10 depicts the results for the Min 3 mode of SCUs. As can be seen, for each skill (component) appearing on the x-axis, a very high number of new practitioners get some exposure ($> 0\,\text{h}$), while for several skills (particularly on the left half of the axis), a fairly significant number of practitioners get > 50, > 100 or even $> 200\,\text{h}$ of potential learning time. This is because these skills were already in higher number in the original resource pool, since these components receive more tickets, and thus have more opportunity to spread to the rest of the pool via SCUs. However, what is interesting to note is that there are even some niche skills that

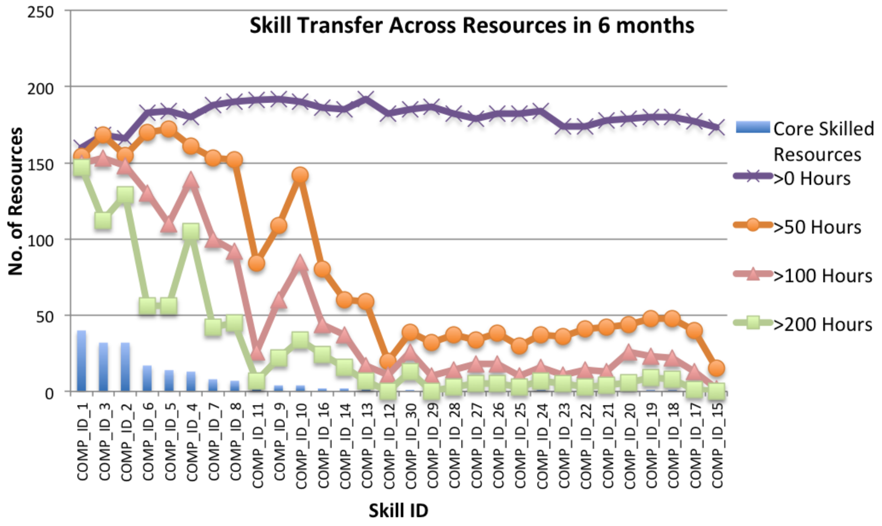


Fig. 10. Exposure to new skills.

appear on the right half of the skills axis (where the original number of core skilled resources was low), which new practitioners may be receiving some training on, with many of them receiving > 50 h of exposure, and some even > 100 or > 200 h.

These results clearly demonstrate the value SCUs can provide toward organizational learning in an incremental, peer-driven way, and in the context of actual work carried out.

5. Related Work

We believe that the novelty of our work is in the usage of SCU teams in problem resolution of otherwise sequential services processes. For example, the authors of Ref. 6 developed SYMIAN, a simulation framework for optimizing incident management via a queueing model based approach, which identifies bottlenecks in the sequential execution of ticket resolution. SYMIAN is based on the current implementation of incident management in the IT service provider organizations. Our approach is different from Ref. 6 as it takes into account optimization of resolution time through parallelization of work effort in the context of otherwise sequential execution of work.

A number of researchers have looked at the problem of mapping tickets to teams based on the problem description. For example, Ref. 7 develops a Markov model by mining ticket resolution sequences of historical tickets that were transferred from one team to another before being resolved. In Ref. 8 supervised learning techniques are used to determine the likelihood that a service request corresponds to a specific problem symptom; prior performances of teams are then analyzed to identify groups that have the necessary skills to resolve these problems. In Ref. 9, an auction-based pull model for ticket allocation is proposed, along with incentive mechanisms for practitioners to be truthful about expected resolution time when bidding for tickets. Unlike our approach, however, none of these works consider dynamic team formation to facilitate faster resolution of tickets through collaborative problem solving. The use of component dependency graphs in the incident management process has also been explored.[10,11] However, these have mainly been used to correlate problems and to search possible solutions rather than to automatically establish a suitable team for solving problems.

Human-based tasks, e.g. in BPEL4People,[12] can be used to specify human capabilities or certain management tasks, e.g. by utilizing human-specific patterns.[13] However, this model relies on specific, pre-defined management processes which are not suitable for complex problem resolution, as we have discussed in this paper. Crowdsourcing[14,15] has been employed for solving complex problems, but while it also offers parallel computation power, our approach is distinct in its use of *social collaboration* to harness complementary skills within an organization and drive towards a common goal.

This paper significantly extends the work earlier reported by the authors in Ref. 16. Among other things, the impact of dispatcher skill on operational metrics such as TTR and SLO violations, and the investigation of social interactions and cross-skills exposure through SCU collaboration, are new results not covered in Ref. 16. Also, the base simulation results related to resolution effort, time and SLO violations (Sec. 4.2) are derived from a ticket set with a different likelihood sequence generator than the one used in Ref. 14, with the more standard 80:20 distribution being used in this work.

6. Conclusions and Future Work

In this paper, we demonstrated how the construct of an SCU may be employed in the IT service management domain to foster collaborative problem resolution, and how it may co-exist with and complement structured workflow driven business processes for incident management. The interplay of SCUs with business processes significantly improve key operational metrics such as TTR and SLO adherence. By providing a common platform for practitioners with different skills to come together and resolve problems, SCUs open up exciting avenues for knowledge transfer and cross-skilling over a longer term.

Our work is part of a broader mission to investigate the interplay of service-oriented and social computing concepts. Whereas we believe the initial results from our simulations based on real-world experiences from the service delivery business of a large IT Service provider are very promising, future work will address the following:

(1) Our current model assumes the SCU to be an organizationally implemented work model, i.e. skill and availability of resources will drive SCU formation. Social computing has a richer set of mechanisms, such as incentive and rewards, that are not yet part of our framework.

(2) We will investigate formal models of cross-skilling via on-the-job learning, based on the initial work laid out in this paper. The interplay of such learning with more structured organizational training is an exciting area of research.
(3) An important next step is to realize this approach in a real service delivery environment.

References

1. Gartner identifies four converging trends that will change the face of IT and business, 15 November 2010, www.gartner.com/it/page.jsp?id=1470115.
2. S. Dustdar and K. Bhattacharya, The social compute unit, *IEEE Internet Comput.* **15**(3) (2011) 64–69.
3. IT infrastructure library. ITIL service support, version 2.3. Office of Government Commerce, June 2000.
4. T. van Gestel, J. A. K. Suykens, B. Baesens *et al.*, Benchmarking least squares support vector machine classifiers, *Mach. Learn.* **54**(1) (2004) 5–32.
5. IBM SPSS. http://spss.co.in/.
6. C. Bartolini, C. Stefanelli and M. Tortonesi, Symian: A simulation tool for the optimization of the IT incident management process, in *Proc. DSOM*, Samos Island, Greece, pp. 83–94, 2008.
7. Q. Shao, Y. Chen, S. Tao *et al.*, Efficient ticket routing by resolution sequence mining, *14th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Las Vegas, Nevada, USA, 2008.
8. A. Khan, H. Jamjoom and J. Sun, AIM-HI: A framework for request routing in large-scale IT global service delivery, *IBM J. Res. Develop.* **53**(6) (2009) 4-1–4-10.
9. P. M. Deshpande, D. Garg and N. R. Suri, Auction based model for ticket allocation in IT service delivery industry, *Int. IEEE Conf. SCC*, Honolulu, Hawaii, USA, pp. 111–118, 2008.
10. P. Marcu, G. Grabarnik, L. Luan, D. Rosu, L. Shwartz and C. Ward, Towards an optimized model of incident ticket correlation, in *Integrated Network Management (IM)*, pp. 569–576, IEEE Press, Piscataway, NJ, USA, 2009.
11. R. Gupta, K. H. Prasad and M. Mohania, Automating ITSM incident management process, in *Int. Conf. Autonomic Computing*, Chicago, Illinois, USA, pp. 141–150, 2008.
12. WS-BPEL Extension for People (BPEL4People) Specification Version 1.1, November 2009. http://docs.oasis-open.org/bpel4people/bpel4people-1.1-spec-cd-06.pdf.
13. N. Russell and W. M. Aalst, Work distribution and resource management in BPEL4People: Capabilities and opportunities, in *Proc. 20th Int. Conf. Advanced Information Systems Engineering*, CAiSE '08, pp. 94–108, Springer-Verlag, Berlin, Heidelberg, 2008.
14. A. Brew, D. Greene and P. Cunningham, Using crowdsourcing and active learning to track sentiment in online media, in *Proc. 2010 Conf. ECAI 2010: 19th European Conf. Artificial Intelligence*, pp. 145–150, IOS Press, Amsterdam, The Netherlands, 2010.
15. A. Doan, R. Ramakrishnan and A. Y. Halevy, Crowdsourcing systems on the world-wide web, *Commun. ACM* **54**(4) (2011) 86–96.
16. B. Sengupta, A. Jain, K. Bhattacharya, H. L. Truong and S. Dustdar, Who do you call? problem resolution through social compute units, *ICSOC*, Shanghai, China, pp. 48–62, 2012.