IBM Research Report

Service Interaction Networks: Simulation and Applications

Kashyap Dixit
IBM Research Division
IBM India Research Lab
New Delhi, India

S Kameshwaran
Indian School of Business
Hyderabad, India

Sameep Mehta
IBM Research Division
IBM India Research Lab
New Delhi, India

Vinayaka Pandit
IBM Research Division
IBM India Research Lab
Bengaluru, India

N Viswanadham
Indian School of Business
Hyderabad, India

IBM Research Division
Almaden - Austin - Beijing - Delhi - Haifa - T.J. Watson - Tokyo - Zurich

Abstract

Service Interaction Networks [7] are abstractions used in modeling and analyzing people-centric aspects of service delivery. They pose interesting challenges from modeling, algorithmic and knowledge mining viewpoints. Typically, the data needed for the construction of service interaction networks is sensitive in nature and organizations may not readily share the data. Therefore, it is imperative that we build realistic simulations of service interaction networks so as to enable not only the development of novel analyses, but also to validate their applicability. Lack of benchmark datasets to compare against poses challnges in validating simulations. We present a simulation tool designed to take into account the people-centric effects of service delivery and an interesting approach to validate it. We hightlight some novel analyses of service interaction networks that require new modeling and algorithmic insights.

1 Introduction

One of the distinguishing features of the services industry is the high emphasis on people interacting with each other and serving customers rather than transforming physical goods in the process. This aspect of service delivery results in several unique characteristics of services as opposed to other industrial settings. Naturally, modeling and analyzing services and their delivery, taking into account people-centric aspects is an important research direction.

In a typical service delivery scenario, an organization consisting of a set of trained actors receives a time stamped stream of service requests. Each request is serviced by a subset of the actors as per its requirements. It is natural to view this process in terms of an abstract network. The notion of *Service Interaction Networks* was introduced by Kameshwaran et al. [7] to help model and analyze wide range of problems that arise in such a setting. However, several reasons (elaborated later) make it difficult to obtain the necessary real-life data, especially for the purpose of publicly disseminated research. In this paper, we present a simulation platform that captures various people-centric issues of service delivery and can be used to generate instances that are likely to reflect real-life observations. We present some example analyses of service interaction networks that have interesting applications in service delivery.

The central abstract construct in service delivery scenarios is the so-called *process model*. It consists of many atomic work units (specialized and well-specified). The process model connects the various different atomic units associated with the process so as to reflect the logical and precedence constraints, and communication requirements. Typically, skilled individuals or a team possessing an appropriate combination of skills is assigned to an atomic unit and we call them as *agents* or *actors*. As we conceive of the service requests as tasks to be performed, throughout the paper, we will refer to the process model of the request as *task-graph*. Abstractly, it consists of nodes that represent specific, atomic work units that can be assigned to actors and edges that capture interaction requirements between the actors assigned to the corresponding work units. In general, the edges could connect multiple work units, i.e, they could be hyperedges. A request, represented by its task-graph is executed by assigning actors to each of the nodes. This assignment takes into account required skill sets, capacity and other specifications of the tasks. If we consider the set of all the actors in the system and the interactions between them during the course of executing a large number of service requests, we get networks that capture the overall interaction patterns among them. Each request also has an overall outcome associated with it. Combinging the outcomes with the interaction patterns gives rise to an enriched *Service Interaction Network* [7] that could potentially reveal critical insights on the service delivery.

Ideally, instead of monitoring request level outcomes, one would like to further micro-monitor the execution of task-graphs. But, there are several issues that make micro-monitoring difficult. Chief among them is that a subset of the actors (functionalities in other words) assigned to the requests could be independent service vecndors (ISVs) whose actions may not be monitored. Even within an organization, such micro-monitoring could result in issues related to morale, trust between colleagues, and unfair competitions etc. Therefore, in most scenarios, only high-level monitoring, such as measuring the overall effectiveness of a request resolution, is carried out. We now discuss some scenarios in which notion of service interaction networks is helpful.

1.1 Examples

Let us consider the case of an orchestrator of supply chains. *Orchestrator* is a management literature metaphor to describe a player who organizes and manages a set of activities in a network by ensuring value co-creation opportunities in the system and value appropriation mechanisms for each player.

Examples of orchestrators are Li & Fung (Hong Kong) in retail [3] and Medion in personal computing [9]. Let us consider the case of Li&Fung, which orchestrates an innovative service supply chain in apparel industry [6]. They create a customized value chain for each customer order and orchestrate a production process that starts from raw materials all the way to the finished products. In other words, they seamlessly bring together the large apparel retailers and a global pool of apparel-related suppliers and create value without owning any of the resources used in the production. The key differentiator they bring in is the extensive domain knowledge. Each order typically gets executed by multiple task-graphs which go through the complete production cycle and is orchestrated by their domain experts. Just the revenue for the services offered (not including the actual value of the eventual retail goods) is USD 11 Billion. Identifying the providers for different orders is based on network constraints and local economic, political, public policy, and cost considerations. However, if there is more information regarding the providers' success and failure measures and their interacting capabilities with other providers, then the orchestrator can choose the providers in an optimal fashion with reduced risk.

Let us now consider the case of distributed software development. A typical software service firm delivers its services from geographically distributed teams. Teams possess expertise in different functionalities from a broad domain of services being delivered. Expertise could be product design and architecture, development capability in different technical disciplines, integration, testing, and debugging. Typical projects flow through different functionality modules at various stages of the project in a precedence oriented fashion. Parameters associated with a project such as, overheads incurred, efficiency, success etc. depend heavily on the team behavioral aspects of the different teams involved and effectiveness of coordination between related teams. Typically, the project manager ensures that the teams have the required skill set to *successfully* complete the project. However, choosing team with the sole intention of matching skill set demand may not produce optimal teams. In this scenario, we can view each project as a task-graph and each team as an actor as it is responsible for basic units of the task-graphs. Organizational constraints allow us to measure the effectiveness of each project only at a macro level rather than at the level of how individual teams performed. A firm can, not only gain better understanding of the team dynamics, but also arrive at better team compositions in future by analyzing the macro level data corresponding to the task-graphs and their execution in innovative ways. Recently, papers in the area of software engineering have considered these and other related issues in distributed software development [12, 11, 13].

1.2 Our Contribution

Sensitivity of the required data is a major motivation for simulating service interaction networks. The data required, even when anonymized to protect privacy, reveals significant statistical information about the efficacy of the service organization. For example, one can compute statistical distribution of the outcomes. While an organization may be willing to use the insights obtained from novel analyses such as those presented in Section 4, they would like to obtain the insight without furnishing the sensitive data. The main goal of our work is to simulate realistic service interaction networks so as to enable interesting modeling, analytical, and algorithmic research in this domain.

Apart from the above business consideration, there are even more compelling technical motivations for research on novel simulations. This is due to the following unique characteristics of service delivery: (i) the degree to which human interactions dominate the total time and (ii) the degree to which human interactions influence the overall quality. Most industrial jobs execute analogues of the task graph in an automated setting in which the machines play dominant role. The impact of human factors on overall quality is not as high as in services. Therefore, simulation models developed in the machine-dominated settings cannot be trivially extended. In Section 2, we elaborate further on these issues.

In this paper, we present a tool which is designed to capture almost all important people-centric aspects of service delivery and build realistic simulations of service interaction networks. The tool also provides simple interfaces by which domain knowledge of experts can be used to improve the results. The main insight we exploit is that typical performance models used for machines cannot capture complexities of human interactions. We employ some heuristics and techniques of randomness to overcome the limiations of models and try to capture real-life behaviours. We present some novel analyses of service interactions networks, both structural and statistical, and show how they can enable important applications in service delivery.

1.3 Related Work

In computer science, the notions of Turing Machine and Operating Systems have been fundamental. Turing Machine gave theoretical tools to understand *computation* with *time and space* as the abstract resources, and Operating Systems helped model various resources used by a computer system and orchestrate their use by different computing processes. Ramaswamy and Banavar [10] proposed a formal model for service delivery motivated by the above concepts. They motivate the analogies between the notion of "algorithms" and that of "processes" in service delivery. Further, they draw analogies between operating systems and service delivery systems. They also highlight the unique features of services that require the development of a new formal model for service delivery. Their paper, part of an ongoing research program, presents a formal model and shows how it can be applied in practical settings from domains such healthcare, IT infrastructure, and hospitality services.

Modeling of service delivery in terms of ideas from supply chain management also faces several challenges. Unique characteristics of the services, such as, *co-production-consumption*, *heterogeneity*, *intangibility*, and *perishability* (CHIP) [8], result in phenomenons not observed in supply chains. An excellent survey article by Dietrich and Harrison [4] covers various modeling, mathemtical, and algorithmic challenges that arise in building efficient service chains as opposed to traditional supply chains.

A common theme in [10, 4] is the emphasis on formal modeling of service delivery. We view our work as a tool that feeds the right input data and the parameters to the higher level models as above. The raw data needs to be analyzed, especially taking into account the people-centric aspects (see Section 2), and an appropriately transformed data makes for a better input to such models. Simulations and demonstration of analyses on top of them provide such a boot-strapping mechanism. Further, information mined from such analyses can be excellent input to applications like workforce design, retention, and optimization.

2 Need for Simulation

Dietrich and Harrison briefly capture a gamut of implications of people-centric aspects of service delivery: "In business services, particularly those with a significant labor component, there exists significant variability in the amount of resource required to complete a step in service production. We also expect to see non-linear effects in team formation (both super-additive and sub-additive), and changes in the capability and efficiency of individuals over time. Additionally, human workers have the capacity to learn, to improve their efficiency and to acquire new skills through training or work experience. Hence the choice of which workers to use for a specific task may be influenced not only by current needs, but by a view of expected future requirements. " [4]. We further elaborate on these issues as they shape almost all aspects of the tool.

Simulations of industrial processes for quality control and systems for analyzing operations have proved very useful. Most simulations model individual steps using simple parametric models and they model the systems using simple queuing networks. However, such an approach is not sufficient in service delivery where the human labor and human interactions play a dominant role. Simple mathemtical models cannot capture the essense of human elements, even in professional settings. Consider the following comparision between mechanical and human elements of a system:

- Unlike machines, people learn new skills on the job and hence planning for trnasitions in roles is essential (*learning and transitioning*).
- Under normal working conditions, performance of machines is predictable. Performance of people can witness both super-additive or sub-additive effects depending on team settings and skills imparted with appropriate training. (*team composition and training*)
- When machines have to interact, their adherence to designed protocols is absolute. When people interact, even in professional settings, adherence to designed protocols is treated flexibly and its effect can be both positive and negative. (*flexible protocol design*)
- Unlike machines, people have accentuated *ramp-up* and *slack-down* states. (*ramp-up, slack-down*)
- Failure of machines is rare and typically of *crash* type. People are prone to frequent faults; but, mainly *transitive* in nature. (*nature of faults*)
- Unlike for machines, exogeneous factors influence the process of people entering and exiting the systems (*effect of exogeneous factors*).

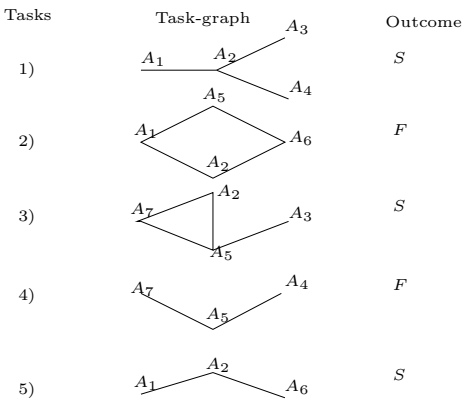

Figure 1: Example: $N = 7$ and $T = 5$

These points call for research on analyzing service interaction networks. But, data pertaining to service delivery are sensitive and hard to obtain. Therefore, simulation is essential in developing modeling and algorithmic techniques for service interaction networks as shown in [7]. Below, the notion of service interaction networks is formalized.

2.1 Service Interaction Networks

In our setting there is an orchestrator $\mathcal{O}$ who can avail the services of a set of actors $\mathcal{V} = \{1, 2, \ldots, N\}$. The orchestrator $\mathcal{O}$ specializes in accomplishing high-level tasks from a broad domain and the actors provide the different functionalities to accomplish the high-level tasks. Given a task t, the orchestrator composes a task-graph consisting of a subset of $\mathcal{V}$ to accomplish the tasks. We assume that the set $\mathcal{T} = \{t_1, t_2, \ldots, T\}$ denotes the set of tasks performed by the orchestrator over a significant period of time. In the most general form a task-graph W_t can be represented as a graph $G_t(V_t, A_t)$ with vertices $V_t \subseteq \mathcal{V}$ and edges A_t between vertices in V_t that model the direct interactions between any two actors in V_t. However, certain tasks require repeated interactions which are cumbersome to represent in terms of direct edges. Therefore, we represent each task-graph as a hyper-edge connecting all its actors. Abstractly, this highlights the fact that an interaction involving all its actors took place before the task was accomplished. Associated with each task-graph W_t is an "outcome" R_t which is a measure of the effectiveness of its execution. In general, the outcome could be a normalized number between 0 and 1. We call the collection of actors along with the hyper-edges representing the various tasks and their outcomes as *service interaction networks*. An instance of a service interaction network is given by $(\mathcal{V}, \mathcal{T}, \cup_{\forall t \in \mathcal{T}}(W_t, R_t))$ where the tuples (W_t, R_t) associate task-graphs with their outcomes. Figure 1 is an example with $N = 7$ and $T = 5$.

3 Simulator

In this section, we describe the structure and functionalities of the main components of the simulator. Each component is implemented by one or more class(es), wherein the members capture the key properties and functions implement the key operations. *Please note the goal here is not to provide a commercial toolkit and therefore, certain instantiations of the operations are simplistic and are for only expository purposes.* We believe that the simulator is rich enough to enable a domain expert to capture and model complex real life scenarios. In our discussion, we provide component level description rather than at the class level.

3.1 Actor Component

This component instantiates, stores and maintains the information for each actor. This information is subsequently used in assigning actors to tasks and further analysis (c.f. Section 4.2). To aid these tasks, constructors and standard functions like *get()*, *set()*, *copy()*, etc are provided. In the rest of the paper, we let N denote the number of distinct job roles in the system. The member of this class are:

- **Actor Name** and • **Actor ID**. Both are identifiers.

- **Job Roles:** An array of size N which speicifies the different roles that the actor is capable of performing.
- **Capacity:** This N element array captures the capacity of an actor corresponding to each role in *Job Roles*. The physical meaning of the capacity is domain dependant. The capacity can imply work hours per day, number of orders processed per day etc.
- **Cost:** This is an array of size N which specifies the cost of employing an actor for different job roles. This gives the users the flexibility of associating differents costs to an actor depending on the role performed.
- **Performance Models:** This is an array of N performance models, one for each role that the actor can perform. For each role, the performance model specifies (μ, σ), the mean and variance of the actor's performance in the role. But, this is not applied as it is. This is adjusted to people-centric effects such as compatibility with other actors in the task, slackening factor of the actor (both aspects are elaborated below). Once an adjusted (μ_a, σ_a) is obtained dynamically, Box-Muller method of sampling from a normal distribution is used to obtain his performance in the current task.
- **Compatibility List:** The performance of an actor depends on how he teams with other members of the team. This list specifies the list of other actors with whom he has high compatability. It may either be populated by authorized users or be learnt dynamically based on past interactions.
- **Incompatibility List:** Similarly, list of incompatible actors.
- **Slackening Factor:** This attribute aims to model a peculiar characteristic of people centric activities. An agent who is well qualified to perform a job role can over time become relaxed, complacent, or bored due to repetitive nature of job. This can have a undesirable effect on the agent's performance. The slackening factor ($s \in [0,1]$) can be either set up domain experts or it can be policy driven. For example, if A performs a job role J with a frequency F in the last T tasks, increasing the slackening factor by an ϵ. Although we use a single slackening factor, in general it might be better to have separate factors for each role. consecutively for K times, the chances of slackening increase by ϵ.
- **Available:** Flag to indicate the availability of the actor.
- **History:** This is a data structure of actor's past interactions. Stores the list of tasks assigned, outcome of those tasks, and the role performed in those tasks. This can be used in both simulation and analyses. For example, we use the frequency of a particular role in the last T tasks perfomed by the actor to determine his slackening factor in our simulations.

Quantities such as F, T, ϵ are user specified constants based on domain knowledge.

3.2 Role Component

Similarly, this class is used to store the information about various role. The member of this class are:

- **Role Name** and • **Role ID**. Identifier fields.
- **Role Description:** Stores the description of the role and meant for report generation.
- **Role Expertise Levels:** For each role, it stores the levels of expertize required to perform the role. This is used to capture that certain specialized jobs can only be done higher level actors.
- **Role Transition:** As detailed earlier, that unlike machines, people tend to gain new skills and can either perform advanced jobs or can perform same job with higher expertise. This field stores the transition information on the transitions that are possible between different roles and different levels of experience. This is actually common data structure across all instances of this class. This data structure is useful in certain analyses described in Section 5.

For each data element in **actor** and **role** components, associated generator functions are provided. Different functions can be plugged to simulate different scenarios. For example, capacities for actors can be sampled from normal distribution to model the most common scenarios. Exponential distribution can be used create very few actors with very high capacity. Similarly, different performance models for each actor can be used to simulate normal as well as special cases. For example, few select agents can be modeled as *super successful* (tasks always successful if they are present) or *super failure* (tasks always fails if they are present). Due to space consideration, we omit detials of different generators within the simulator which are currently based on our understanding and discussions with some subject matter experts (SMEs) in some domains. Appropriate generators as per special needs can always be plugged in.

3.3 Tasks Component

This component is used to specify task requirement, priority, actor assignment and outcomes.

- **Task Name** and • **Task ID**. Identifier fields.
- **Task Graph** As mentioned earlier, each task is broken into subtasks. The relationship between the subtasks is modeled as a (task) graph. Currently, the simulator only supports precedence relationship which essentially reduces the graph to linear chains.
- **Required Roles :** For each subtask, the required job role to perform the subtask is specified in this attribute.
- **Required Expertise :** The expertise level required for each subtask is stored in this field.
- **Required Capacity :** This field captures the required capacity for each subtask.
- **Potential Actors:** Based on the above three components, a list of potential actor IDs which could be assigned to a subtask is stored. Essentially, actors who can fulfill the three requirements and are available are added to this list. The list is maintained for each subtask and subsequently for each task.
- **Job Criticality:** This attribute specifies the importance of the job. The importance is domain dependant. It can be dependant on cost, priority, profit, penalty etc.
- **Outcome:** The outcome for each task is stored in this field. There can be various methods to simulate the outcome as described later.

The key member functions in this component apart from *Read/Write From/To Stream*, *Get/Set* are :

- **Task Generator:** The task generator generates attributes like number of subtasks, requirements, criticality etc. Currently, each attribute is generated using a standard distribution. For example, the number of subtasks is generated using power law which results in lot of tasks with low or moderate number of subtask and few tasks with high number of subtasks. The distribution for each attribute is input to the generator and can be changed to simulate difference scenarios.
- **Assign Potential Actors:** As mentioned earlier, the actor list is scanned for each subtask and each actor which fulfills the requirement criteria (role, expertise and capacity) and is available is assigned as a potential actor. No change is made to the available flag (See Section 3.1) of agents in this step. In essence, an agent can be assigned to multiple subtask within a task or across tasks at this level.
- **Outcome Assignment:** This method is used to compute the outcome of the task after assignment of actors. The simplest of method is to randomly assign one of the possible outcomes. Similarly, the outcomes can follow normal distribution or multinomial distribution. Our current method is as follows: for each actor assigned to the task, the actor's performance model corresponding to the role of the assigned sub-task is applied as explained in Section 3.1. This method takes into account effect of compatible and incomptible actors in the assignment. We then take an average of the performance levels of all the assigned actors. If a binary $\{Success, Failure\}$ decision is desired, then, it is made based on thresholding.
- **Book Keeping:** This function performs updations like freeing up the actors, capacities, role transition etc.

3.4 Workflow Component

A workflow is defined as a set of tasks. This is a logical aggregation notion used to allow the practitioners to classify related tasks together. That way, specialized generators capturing domain knowledge can be used as needed. As such, it is not part of formalization of service interaction networks.

- **Workflow Name** and **Workflow ID**. Identifier fields.
- **Tasks:** List of IDs of tasks belonging to this workflow.

The key member functions are:

- **Workflow Generator:** Generates a workflow (and its associated tasks) capturing its domain characteristics.
- **Final Actor Assignment:** This functions uses the potential actor assignment information to actually assign actors to different sub-tasks of indidivual tasks. In fact, different methods can be used to do the assignment as in the case of outcome generation. They could be random feasible assignment, greedy assignment, or assignment schemes that achieve a higher level business objective such as optimal workforce management [1].

4 Analyses and Experiments

As already argued, effective simulation of service interaction networks is critical to get the data required to validate the efficacy of novel analyses of service interaction networks. We present an interesting validation of our simulation technique. We then present some examples of novel analyses on service interaction networks.

4.1 Validation

Validating simulations of service interactions networks is a very challenging and interesting exercise as there are no benchmark datasets to compare against. We present an approach that we believe has wider applicability in validating simulations where no or limited benchmark data exists.

It is imperative to choose a publicly documented domain in which people-centric aspects dominate the outcomes. For this puprose, we choose the dataset of internet movie database (IMDB)[1] which maintains the following: a vast list of movies, list of important actors for each movie, and average user rating for each movie. Over thousands of movies, the statistical distribution can hide people-centric effects. So, we choose a small set of A actors and the set of movies M in which atleast K actors from A have appeared. We present results for a set of 28 actors and 118 movies which share atleast 3 actors from the list. In terms of service interaction networks, the movies represent the tasks and the user ratings represent the outcomes (between 0 and 10). Corresponding to each actor, we create a profile consisting of his average performance, list of compatible actors, and list of non-compatible actors. We now simulate an instance which has 118 tasks; the ith task has as many actors as the ith movie from the list A. In an interesting twist, we assign the actors to the tasks randomly rather than the matching profiles. This is because this is the simplest and most commonly used assignment scheme in simulations.

Let H_o denote the histogram of the user ratings for the movies. Let H_s be the histogram for the outcomes of the tasks in the simulation. We compare H_o and H_s. If the simulation methodology is reasonable, then, the two histograms should bear resemblance. This would indicate that, even for small networks, we can depend on the simulator to paint realistic scenarios. Here are the comparisions for different parameter settings of the simulator.

H	< 2	2-3	3-4	4-5	5-6	6-7	7-8	> 8
H_o	7	8	16	14	24	25	18	6
H_{nc}	0	0	0	10	94	13	1	0
H_{s1}	1	7	13	27	45	16	5	4
H_{s2}	1	2	8	14	49	23	17	4

In the above table, H_o is the original histogram of movie ratings. H_{nc} is the histogram obtained when the simulation does not take into account the compatibility and incompatibility effects. Note

[1] http://www.imdb.com/interfaces

that H_{nc} is useless and is just concentrated around the mean. The histogram H_{s1} is the result of simulation with compatibility and incompatibility lists, and compatible settings are assumed to result in improved performance. This histogram is close to the original. Only in the ranges $\{[0,2],[7,7]\}$, the simulation result is different from the original (and difference is transferred to mean). Histogram H_{s2} shows the result of the simulation with compatability and non-compatibility lists, and an assumption that an incompatible setting results in higher variance in performance. In this case, notice the similarity with the original except in the range $[0,2]$. Thus, our techniques capture people-centric effects realistically even when the simulation is on small number of actors and tasks.

4.2 Analyses

We briefly discuss some analyses of service interaction networks requiring novel modeling, mathematical, and algorithmic insights and results of experiments based on the simulator. Details of the formulations, modeling and mathematical results obtained in these case-studies are also part of our larger research agenda and can be seen in [7, 5].

4.2.1 Ranking actors

Traditional ranking in social and technological networks, like PageRank focus mainly on the structural aspects of the network. But, service interaction networks are unique: they have both structure and outcomes (for each interaction). Morever, ranking in a service interaction network has to rank the actors based on their effectiveness. Effectiveness of an actor should depend on both his structural connections and outcomes of the interactions in which the actor is present. In [5], this problem was formalized and a novel algorithm based on the notion of *alpha-centrality* [2] was proposed and evaluated on real-life data from IMDB. The details of the experimental set-up and interpretation of results can be seen in the version of the paper available online [5]. Briefly, they compared the sensitivity of traditional ranking algorithms and the alpha-centrality based algorithm to changes in structure and outcomes of the network. Here, we briefly present the results of the analysis on simulated networks obtained as described in Section 4.1. The Kendall-Tau (τ) measure shown in the table below indicates the sensitivity. The closer it is to 1, less sensitive is the algorithm towards the outcomes. Closer it is to 0, more sensitive is the algorithm to the outcomes. We refer the reader to [5] for better understanding of the experiments. Analysis for ranking actors can be used to appropriately transition the expertise level of actors.

Instance	τ for traditional SVD ranking	τ for new algorithm in [5]
1	1.0	0.47
2	0.95	0.29
3	0.97	0.25
4	0.99	0.18
5	0.88	0.17

4.2.2 Rating of actors and teams

The service interaction networks capture the structural and outcomes aspects of the interactions. Rating of actors and teams in a way consistent with the higher level observations has many useful applications. Intuitively, a rating technique should: (i) not only rate individual actors, but also different teams based on their effectiveness as a team and (ii) the ratings should be consistent with the observations at the higher level (task level). In [7], the problem of rating actors and teams was formalized, and especially contrasted with a simple-minded heuristic based on aggregated scores. An initial version of the current simulator was used to show the effectiveness of a novel iterative refinenement algorithm for rating actors and teams. Compared to the aggregate-based rating heuristic, they showed that the iterative technique could improve the percentage of tasks that are consistent with the ratings by upto 20%. However, the simulated tasks had upto 20 actors in a task which favours the iterative technique. In the following table, we present the results when the number of actors in a task is much smaller, ie, < 5, thus further validating its efficacy. The details of the experimental set-up and measurement of improvement can be seen in [7]. This analysis can be used in applications like targeted team building, workload assignment to maximize quality and throughput etc [1].

#Actors $= N$	% Improvement over aggregate heuristic $T = O(N)$	% Improvement over aggregate heuristic $T = O(N \log N)$
50	9	10
100	11	8
150	11	8
200	7	7
250	8	7
300	8	7
400	8	6

5 Discussion and Future Work

In this section, we outline some of our planned and ongoing work on novel analyses for service interaction networks. We also discuss planned enhancements for the simulator.

- As mentioned in the introduction, one of the distinguishing features of services is that people can make role transitions (unlike machines). Two types of transitions are possible: firstly, the actor can move to higher level of exprtise; secondly, to different job roles. The former transition is relatively easy and can be done by observing the agent's historical performance. However, for handling the second transition, better algorithms need to be designed which can determine if the actor has acquired necessary skills for the target role.

- The simulation platform, currently, only supports those task graphs which can be represented in a linear fashion. In future, we would like to extend the capability to support generic graph structure.

- One of the issues highlighted in the introduction was that people's adherence to protocols is to a lesser degree than machines. Therefore, when we design a protocol for service delivery, we should estimate its impact on the delivery system assuming that the protocol is likely to be adopted in a flexible manner. One of the interesting ways of extending our tool is to allow mechanism to specify protocols, generate tasks, and execute them with flexi-adherence to the protocol and evaluate its impact. We are currently formulating a way of doing this in a systematic manner.

To summarize, we contend that people-centric aspects of service delivery and the sensitivity of the data required to analyze them necessitate novel and effective simulations. Our tool is especially designed to capture the people-centric effects and we presented an interesting approach to validate simulation in the absence of datasets. In this emerging discipline new simulation techniques will go hand-in-glove with new modeling and analysis techniques.

References

[1] G. Banavar, M. Goyal, N. Kambhatla, R. Lotlikar, D. Majumdar, G. Parija, S. Roy, and S. Soni. Workforce optimization for IT application service providers. INFORMS 2008 Annual Meeting, 2008.

[2] Philip Bonacich and Paulette Lloyd. Eigenvector-like measures of centrality for asymmetric relations. *Social Networks*, 23:191–201, 2001.

[3] John Seely Brown, Scott Durchslag, and John Hagel. Loosening up: How process networks unlock the power of specialization. In *The McKinsey Quarterly: Special edition on risk and resilience*. 2002.

[4] Brenda Dietrich and Terry Harrison. Serving the services industry. *Operations Research and Management Science (OR/MS)*, 33(3), 2006.

[5] Kashyap Dixit, S. Kameshwaran, Sameep Mehta, Vinayaka Pandit, and N. Viswanadham. Simultaneously exploiting structure and outcomes in interaction networks for node ranking. IBM Research Report R109002 available at http://domino.watson.ibm.com/library/ CyberDig.nsf/home, 2009.

[6] John Hagel, Scott Durchslag, and John Seely Brown. Orchestrating loosely coupled business processes: The secret to successful collaboration. Available at http://www.johnhagel.com/orchestrating_collaboration.pdf, 2002.

[7] S Kameshwaran, Sameep Mehta, Vinayaka Pandit, Gyana Parija, Sudhanshu Singh, and N. Viswanadham. Analyses for service interaction networks with applications to service delivery. In *Proceedings of SIAM International Conference on Data Mining*, 2009.

[8] C. Lovelock, J. Writz, and J. Chatterjee. *Services Marketing: People, Technology*. Pearson Education, 2006.

[9] Andrea Ordanini and Kenneth L. Kraemer. Medion: the retail orchestrator in the computer industry. 2006.

[10] Lakshmish Ramaswamy and Guruduth Banavar. A formal model of service delivery. In *IEEE SCC (2)*, pages 517–520, 2008.

[11] Bikram Sengupta, Satish Chandra, and Vibha Sinha. A research agenda for distributed software development. In *28th International Conference on Software Engineering (ICSE)*, pages 731–740, 2006.

[12] Vibha Sinha, Bikram Sengupta, and Satish Chandra. Enabling collaboration in distributed requirements management. *IEEE Software*, 23(5):52–61, 2006.

[13] Giuseppe Valetto, Mary Helander, Kate Ehrlich, Sunita Chulani, Mark N. Wegman, and Clay Williams. Using software repositories to investigate socio-technical congruence in development projects. In *Fourth International Workshop on Mining Software Repositories (MSR)*, page 25, 2007.