Analyzing Reliability in Hybrid Compute Units

Muhammad Z.C. Candra, Hong-Linh Truong, Schahram Dustdar
Distributed Systems Group, Vienna University of Technology
{m.candra,truong,dustdar}@dsg.tuwien.ac.at

***Abstract*—Modern development of computing systems caters the collaboration of human-based resources together with machine-based resources as active compute units. Those units can be dynamically provisioned on-demand for solving complex tasks, such as observed in collaborative applications, crowdsourced applications, and human task workflows. Such collaborations involve very diverse compute units, which have different capabilities and reliability. While the reliability analysis for machine-based compute units has been widely developed, the reliability analysis for the hybrid human-machine collaborations has not been extensively studied. In this paper we present models and a framework for analyzing the reliability of hybrid compute units (HCU), which represent on-demand collectives of humans collaboration supported by machines (hardware and software units) for performing tasks. We present the implementation of our models and study the reliability of HCUs in a simulated system for infrastructure maintenance scenarios. Our evaluation shows that the proposed framework is effective for measuring the reliability of the collaboration collectives, and beneficial to obtain insights for improvements.**

Keywords*—reliability analysis, hybrid human-machine collaboration, human computation.

I. Introduction

In the past, we saw mostly machine-based compute units, i.e., hardware and software, providing computing services consumed by human. However, recently many approaches have been developed for intertwining collaborative human-based and machine-based compute units, e.g., [1], [2], to solve complex problems that require creativity and intelligence, where human-created solutions are a must. We use the notion of hybrid compute units (HCUs) [3] as an abstraction representing the composition of collaborative human-based and machine-based compute units that applications need to execute a complex computing task. In today's Internet-based computing landscape, such HCUs can be dynamically provisioned on-demand from, e.g., online collaboration platforms and crowdscourcing marketplaces for human-based units, and cloud-based services provisioning for machine-based units.

Reliability is one of the important quality measures of a system. In a traditional machine-only computation, *reliability* is typically defined as the ability of a system to function correctly over a specified period of time, mostly under predefined conditions [4]. However, in the context where human-based units are involved, the *reliability* property is used with different quantifications and interpretations, e.g., the reliability property can be interpreted as (i) the probability of human errors so that such errors can be mitigated to obtain a high level of safety environment [5], [6], e.g., in healthcare, and transportation sector, (ii) the ratio of successful task executions in a workflow or a business process, e.g., [7], [8], or (iii) the quality of results or contents, e.g., [9], [10], [11].

A set of tools for modeling and analyzing the reliability of HCUs is useful, e.g., (i) for application designers to design, evaluate, and improve collaboration components for executing tasks, (ii) for resource platform providers to deliver more reliable machine-based and human-based compute units such as by providing a reliability-aware discovery and composition service, and (iii) for task owners to tune the task specification to achieve the required reliability.

However, analyzing the reliability of HCUs introduces many challenges. The diversity of the compute units and their individual reliability models brings forth different failure characteristics that must be taken into account when measuring the reliability. The complexity of the collaboration's structures and the large scale of the involved units also contribute to the complexity of the reliability analysis.

Our work presented in this paper tackles the above-mentioned challenges. Our goal is to provide a toolset that can be utilized for analyzing the reliability of HCUs. We adopt models to measure the reliability of individual machine-based and human-based units and introduce a model that can be used for describing the complex structure of collaborations, i.e., a *collective dependency* model. Furthermore, to deal with the large scale of the collaboration landscape, we introduce the notion of *virtual standby units* that abstracts the group of units available from the pool of computing resources. These models are then utilized to perform the reliability analysis.

The salient contributions of this paper are threefold:

(a) We introduce models to describe the dependencies in the running collaborations and to abstract large pools of computing resources.
(b) We propose a framework for HCUs reliability analysis.
(c) We present an implementation of a tool for simulating collaborative human-based and machine-based computing and analyzing the reliability of the composed HCUs.

Furthermore, we verify our approach by presenting some reliability analyses using infrastructure maintenance use cases and we simulate and study the variability of the reliability that may be gained by employing different scenarios.

The rest of this paper is organized as follows. Section II discusses the notion of HCU and our motivation. In Section III, we present models as a basis for our reliability analysis. Section IV discusses the HCU reliability analysis framework. Section V presents the prototype implementation and experiments to exemplify the HCU reliability analysis. Section VI discusses some related works. Finally, Section VII concludes the paper.

The work mentioned in this paper is partially supported by the EU FP7 FET SmartSociety (http://www.smart-society-project.eu/). The first author of this paper is partially supported by the Vienna PhD School of Informatics.

978-1-5090-0089-0/15 $31.00 © 2015 IEEE
DOI 10.1109/CIC.2015.30

IEEE computer society

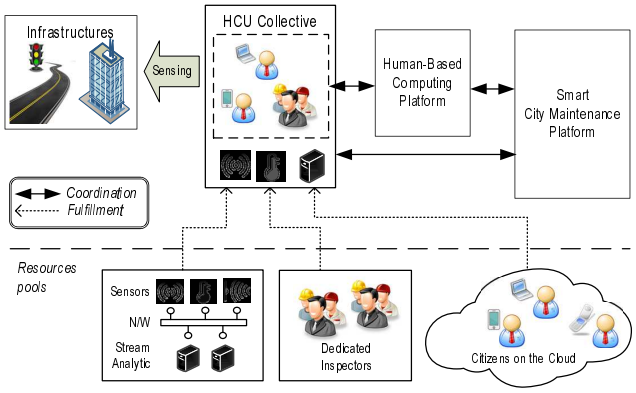


Fig. 1: Infrastructure Maintenance Scenario

II. Motivation

A. Background - Hybrid Compute Units

In a hybrid collaboration, a collective consisting of diverse compute units is composed to execute a complex computing task, i.e., a task that requires active collaboration of humans assisted by machines or software services, given by a consumer process or application. This notion of HCU collective (or HCU for short) constitutes a construct for loosely coupled, and nimble group of human-based and machine-based compute units, which can be composed, deployed, and dissolved on-demand.

The manifestation of HCUs can be seen in various computing systems. For example, human-based services, either standalone or collaborative, can be utilized in a mixed orchestration with software-based services as people activities in business processes, e.g., [1], [12]. In the cloud, we have also seen various sources of human-based compute units, such as crowdsourcing marketplaces, e.g., [13], [2], and social networks, e.g., [14], [15] being utilized as pools of active compute units. The machine-based counter part, such as software-based services, can be provisioned in-house or on-demand from the cloud, for example using cloud based service composition techniques, [16]. Furthermore, diverse collaborative compute units are also utilized in new methodologies for solving complex problems that requires both human knowledge and machine capabilities, such as in smart-city management, traffic control, and urban planning [17], [18].

Many factors affect the reliability of an HCU. The reliability of the underlying resources, as well as the dependencies among them are some of the main factors. Moreover, unlike the traditional reliability analysis, which deals with a priori knowledge of a defined set of units with a certain structure [4], in HCU we deal with on-demand provisioning of dynamic collaboration collectives that may have different compositions for each instances. Hence, different provisioning strategies may yield different reliability of the system.

B. Motivating Scenario

To motivate the importance of the reliability analysis for HCUs, we discuss a scenario where a collaborative system for infrastructure maintenance is utilized in smart-buildings or smart-cities. The system can be employed, for example, by a corporation for maintaining large building complexes. The maintenance is conducted pro-actively by analyzing a possible facility breakdown as shown in Fig. 1. For this system to work, sensors are installed on the monitored facilities to capture occurring events, which are streamed through sensor networks to a data processing center running stream analytics.

In many cases, installing sensors on every facilities is not always feasible and adequate. One traditional way to handle this issue is to send dedicated inspectors for regular inspections. However, such approach can be in-effective for a large maintenance area. Therefore, collaborative citizens are also engaged for revealing issues in places where hardware sensors are not feasible. These so-called human sensing services are coordinated by a human-based computing platform, e.g., a crowdsourcing platform. This platform generates human-based tasks, such as data collections and assessments, and disseminates the tasks to the participating citizens according to their availabilities and locations. An example of such crowdsourcing system has been presented in CrowdSC platform [18]. For places where citizen participations are high, we may no longer need to employ professional inspectors. Hence, the provisioning of human-based collaborations can be made on-demand.

In this scenario, we can define an HCU as a set of collaborative compute units for detecting a particular facility breakdown, which is fulfilled from the available resources pools. Hence, an HCU for a particular building may consist of a set of sensors, participating citizens who live or work in the building, a standby dedicated inspector, and a stream analytic service. In this context, HCUs are said to be reliable when they correctly detect breakdowns.

Here, a reliability analysis is very important and useful for improving the reliability of the collaboration. For example, we can use the reliability analysis to identify whether the increase of citizen participations due to particular incentives is really beneficial. Also, we can identify which HCU formation strategy is more effective to obtain more reliable HCUs.

C. Research Problems

a) Background and Scopes: In machine-based computation, failures are typically caused by natural- or design-faults [4]. However, for human compute units the nature of the faults is different. Humans are prone to execution error [5]. When a human performs a task, it is natural he/she performs an error, which leads to failure. Also, same tasks executed by the same worker on different times may give different results.

In general, reliability models can be categorized into black box and white box models [4]. For human compute units, it is complex to model the internal functioning of a human work using a white box model. Black box models, such as based on interpolation or parameter estimation using historical data, can be used for predicting the individual reliability. Various influencing factors, such as trust, skills, connectedness of the collaboration, as well as past success rates, may affect the reliability of individuals. However, problems may arise for a new unit with no historical data. To this issue we point to approaches for predicting reliability based on similarity such as found in [19]. Our work presented in this paper focuses on the issues of the reliability analysis for mix human-machine

collaborations using black-box models with *a priori* known factors.

b) Problem 1: Traditionally, the notion of reliability is expressed as a function in a continuous time space. However, for human-based computing, this approach is not suitable, since most human-based compute units do not operate continuously. For HCUs, where human-based compute units are involved, we need to model the reliability on a task basis. In Section III-A, we discuss this issue.

c) Problem 2: The reliability of a system depends largely on the inter-dependencies between its elements. In HCUs, the dependencies can be inferred explicitly from the process model, e.g., a workflow, if it is available. However, in many cases the collaboration inside an HCU can be ad hoc. Hence, we need a model to describe the dependency in an HCU in a more agile and flexible way. We approach this issue in Section III-B and use it for HCUs reliability analysis in Section IV.

d) Problem 3: With the advent of the cloud computing, the provisioning of both machine-based and human-based compute units can be made on-demand from a virtually large pool of available resources [20]. In most cases, when a failure occurred on a running unit, another unit can be selected from the cloud to replace. The reliability analysis for cloud-based HCUs must take into account this provisioning model. We propose a solution for this issue in Section IV-A2.

III. Models

A. Reliability of Individual Units

a) Reliability of Machine-Based Units: Measuring the reliability of machine-based units is a well-researched problem [21], [22], [23]. Generally, it can be summarized as follows. Let T be a continuous random variable that represents the time elapsed until the first failure occurs. And let $f(t)$ be the probability density function of T, and $F(t)$ be the cumulative distribution function of T. Traditionally, $F(t)$ represents the unreliability of the system, i.e., the probability that the system fails in time interval $[0, t]$. The reliability, $R(t)$, of the unit is the complement of $F(t)$, i.e., $R(t) = 1 - F(t)$ [21].

b) Reliability of Human-Based Units: In human-based tasks, we do not deal with the exact time when a particular human-based compute unit fails, instead we are more interested in whether a particular task execution is likely successful. Furthermore, in the execution of human-based tasks, the active execution time of the human-based compute units is not continuous, i.e., people may take a break, eat, and sleep. Therefore, in our model, we approach the reliability of human-based compute units using a discrete time space.

Let K be a discrete random variable which represents the number of consecutive successful task executions by a particular human-based compute units until a first failure occurs. Let $f(k)$ be the probability density function of K which also represents the probability of the first failure occurs at k-th task execution. Let $F(k)$ be the cumulative distribution function of K. $F(k)$ represents the unreliability of the human-based compute unit, i.e., the probability that the unit fails at least once in execution $[1, k]$. The reliability, $R(k)$, defines the reliability of the human-based compute units for the execution of all k tasks. Hence, we have

$$\begin{aligned} f(k) &= Pr(K = k) \\ &= Pr\{task_k \text{ fails } \mid task_1, task_2, ...task_{k-1} \text{ succeed}\}. \end{aligned} \quad (1)$$

Depending on the problem domain and the underlying human-computing systems, different discrete distributions can then be employed to define $f(k)$. Note that the distribution parameters of such failure probability may also dynamically change from time to time, e.g., due to human skill evolutions [24]. To exemplify this model, in our experiments described in Section V-B, we approach $f(k)$ using a geometric distribution with non-dynamic parameters.

This model extends models proposed in human reliability analysis and task quality measurement techniques, e.g., [5], [6], [7], [8], [11], where the reliability property, e.g., with respect to the failure/success probability, can be taken for granted. However, instead of using only a single value of failure/success probability for the next human task execution, our model allows the estimation of the reliability as a cumulative probability of failure/success within a set of consecutive task executions. Hence, together with the traditional reliability measurement of machine-based units we could derive the reliability of HCUs in a discrete time space.

B. Collective Dependencies

Members of an HCU depend on each others in order to collaborate effectively. When running a particular task, each member units participate in a certain role by executing the assigned activity. In our work, we propose a model based on the the dependencies among units while performing activities to define the interrelationships between units.

To define these dependencies, we introduce the notion of *inter-dependent collective activities* (*c-activities* for short) on which participating roles perform their actions. Each c-activities provides a deliverable that can be consumed by other c-activities. Hence, each c-activity depends on all of its dependencies so that it can be successfully accomplished. Furthermore, we also introduce alternate dependencies, where a c-activity can be accomplished after at least n of its dependencies have provided the required deliverables.

We define a collective dependency graph as an acyclic graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$, where $\mathcal{A}$ is the set of c-activities executed by the HCU, and $\mathcal{E}$ is the set of dependencies between c-activities in $\mathcal{A}$. Furthermore, for each c-activity we define the roles associated to the c-activity, and for each role we define the units assignments. Similarly with an alternate dependency, an alternate assignment can also be defined, where at least m assigned units must successfully perform the role.

The mechanism to obtain the collective dependency for a particular system is domain-specific. The application designer can define the collective dependency from the ground up, but it can also more practically be implied from the application design. For example, in a process-based application, such dependency can be inferred from the workflow. In a crowsdource-based application, the dependency can be deduced from the relationships between the microtasks, e.g., [25].

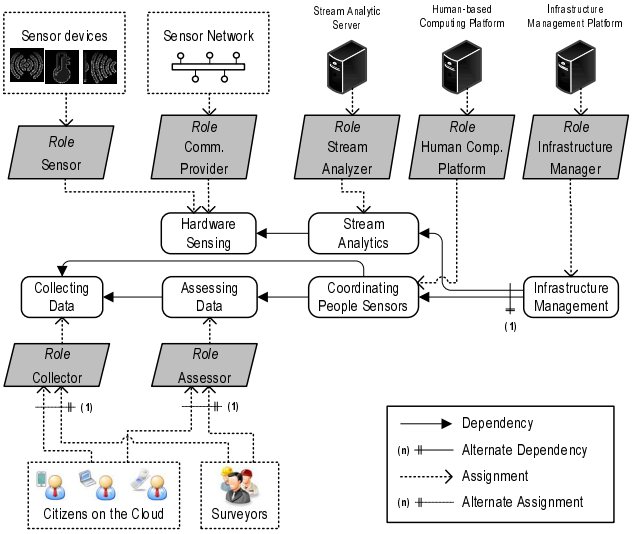


Fig. 2: Collective Dependency

Returning to our previous scenario in infrastructure maintenance, in Fig. 2 we show an example of a collective dependency for detecting facility breakdown as well as the associated roles and possible units assignments. As we will show later, this collective dependency is useful to obtain the execution spanning tree for reliability analysis (Section IV-B2).

C. Reliability of HCU Collective

We define the reliability of an HCU as the reliability of the task execution performed by the HCU, i.e., the probability that the HCU successfully execute tasks. As discussed in the following section, the reliability of an HCU to execute a task depends on the reliability of the individual units involved and the structure of the HCU represented by the collective dependency.

IV. Reliability Analysis Framework

Here we present a framework that provides features to evaluate the reliability of HCUs. The goal of our framework is to measure the reliability of the system consisting HCU instances to execute tasks. More specifically, given a set of k consecutive tasks $T = \{t_1, t_2, ..., t_k\}$, we measure the reliability of all HCU_i provisioned by the system to execute $t_i \in T$. This framework takes the following as inputs: the profiles of the units [9] to determine their individual reliability (Section III-A), and the collective dependency of the task type (Section III-B). Our proposed analysis approach yields the reliability of HCUs provided by the system to execute a particular task type. We retain the aggregation analysis to measure the overall system reliability for various task types as a future work.

A. System Overview

1) HCU Provisioning: In hybrid collaborations, we deal with HCUs provisioned on-demand to execute tasks. Here we introduce a generic HCU provisioning model, as shown in Fig. 3, upon which we could perform a reliability analysis.

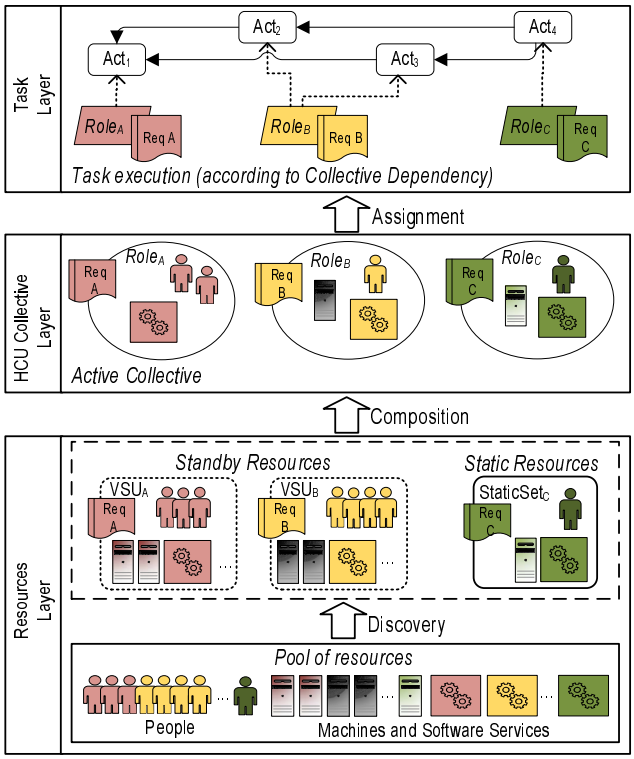


Fig. 3: HCU Provisioning Overview

The requested task contains a set of required roles, $Role_x$, which need to be fulfilled. Each roles execute certain c-activities for the task, Act_x, according to the collective dependency described in Section III-B. For each role, a set of requirements, Req_x, can be defined to guide the provisioning of the units. For example, the requirements may contain a set of qualifications for discovering units. Qualified units are composed to form an HCU to fulfill each roles defined in the task request. Our previous work in [20] discusses some methodologies for this composition.

Units qualified to perform a particular role are discovered from diverse resources pools. These discovered units may represent a static set of resources, e.g., in the case of in-house provisioning. However, in the case of on-demand provisioning, e.g., cloud-based services provisioning or crowdsourced human-based units provisioning, the discovered units represent a set of virtually standby units, VSU_x (Section IV-A2), which can be assigned to a particular role on-demand.

2) Virtual Standby Units: One of the main challenges in a cloud-based hybrid collaboration is to deal with large numbers of units. For example, the number of people participating in a crowdsourcing platform can be very large (e.g., in a smart city). However, since HCUs are task-oriented and provisioned on-demand, we can abstract these large pools of units that will likely be included in the assembled HCU. We call these sets of units as *Virtual Standby Units* (VSUs). Hence, an VSU is a subset of the units pool consisting units qualified to perform a particular role.

With this approach, only compute units providing resources

required for the process of concern are considered for analysis. However, the construction of VSUs should consider not only static profiles, but also the dynamic changes of functional and non-functional properties of the units. For example, in our infrastructure maintenance scenario, we may want to analyze the reliability of the facility sensing capability against a particular building at a particular time slot; therefore, we can utilize the participants profiles such as time availability and location history to decide whether he/she should be included in the VSU for that particular task.

More formally, let t be a task with a set of functional and non-functional requirements $\mathcal{R}_t$. Given a set of all units $\mathcal{U}$, we can define the VSU for t as $VSU_t = \{u_i | u_i \in \mathcal{U} \wedge Sim(\mathcal{R}_t, \mathcal{P}_{u_i})\}$, where $\mathcal{P}_{u_i}$ is the profile of u_i, and Sim is a predicate representing a similarity match. Details of the similarity matching are domain-specific and beyond the scope of this paper. An example of such matching is provided in [26].

B. Reliability Calculation

We employ the following procedures to estimate the reliability of HCUs:

1) We calculate the reliability of individual units based on their profiles (see Section III-A).
2) We determine the reliability for each group of units assigned for a particular role. The assigned groups of units can be in the form of static sets of units (or a single unit) or in the form of virtual standby units (VSUs).
3) We calculate the reliability of the executions of task instances for a particular task type based on the reliability of the group of units assigned for each task roles.

In the following we discuss these procedures in detail.

1) Reliability of Role Assignments: Before we can measure the reliability of the task executions, we need to measure the reliability of the units assigned for a particular role. The reliability for each role assignments in an HCU is defined according to the reliability of the set of units assigned for the role. The member units of a role can be fulfilled either (i) from a *static set of units* (e.g., as in Role Stream Analyzer, Role Human Computing Platform, Role Infrastructure Manager, and Role Communication Provider in Fig. 2) or (ii) from an *VSU* (e.g., as in Role Collector, Role Assessor, and Role Sensors).

a) Reliability of Static Sets of Units: A static set of units may employ only a single unit (simplex), or a certain basic structure such as the parallel structure, where we distribute a task in parallel and expect at least one unit returns a result, or the series structure, where we expect all assigned units provide results correctly. A more complex structure can also be formed from these basic structures. A static set of units may also employ a *static redundancy* for masking faults. One of well-known approaches for a static redundancy is the *M-of-N* redundancy, which consists of N units and requires at least M of them to function properly. For example, in human-based computing, this *M-of-N* redundancy can be in the form of assignments of the same task role to N people, where we expect at least M people provide the correct result reliably. The calculation of the reliability of such static structures are well known and can be found in [23].

b) Reliability of VSUs: When a role of a task is fulfilled using units from an VSU, it resembles the structure of a set of active units accompanied by standby spare units. If any of the active units fails, a standby unit is activated for a replacement. This resilience approach is traditionally called *hybrid redundancy* (or simply *dynamic redundancy* when only a single unit is active), where we dynamically detect (or predict) faults and reconfigure the structure of the running HCU to correct (or anticipate) the faults. In this case, we also need to take the reliability of the detection and reconfiguration component into account.

If the active units from an VSU are assembled to use *M-of-N* redundancy approach, we would need at least M units to function properly. Let L be the number of standby spare units, the reliability of the VSU is given by the probability that at least M units out of $L + N$ units are functioning correctly. Hence, given the reliability of the detection and reconfiguration component R_{DR} and the uniform reliability of each units R_u the reliability of an VSU is given by

$$R_{VSU} = R_{DR} \cdot \sum_{i=M}^{L+N} \binom{L+N}{i} R_u^i \left(1 - R_u\right)^{L+N-i}.$$

For non-uniform R_u, an analytical probability calculation based on each individual unit reliability can be performed.

2) Reliability of Task Executions: When an HCU is assembled, its assigned units form a configuration that fulfills a set of required dependencies as defined by the task's collective dependency model. Due to the flexibility of HCUs, i.e., defined by alternate dependencies and alternate assignments in the collective dependency model, different HCU configurations may be composed for different task instances. We use the concept of *execution spanning tree* (EST) to identify various possible HCU configurations for the task type.

We define that an EST contains the inter-dependent static sets of units and/or VSUs such that its vertices (the static sets of units/the VSUs) are capable to execute a set of required c-activities. That is, given a collective dependency graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$ and static sets of units/VSUs $\mathcal{V}$, we can have an EST $\mathcal{S} = (\mathcal{V}', \mathcal{E}')$, where $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}'$ is the dependency of $\mathcal{V}'$ according to $\mathcal{E}$, such that $\mathcal{V}'$ and $\mathcal{E}'$ encompass one possible alternative dependency set in $\mathcal{G}$.

To obtain ESTs, we could derive the dependencies between the units from the collective dependency. Algorithm 1 presents a procedure to transform a collective dependency into a set of ESTs. For example, let the sensors (Se), citizens (Cz), and inspectors (In) in our infrastructure maintenance scenario are constituted as VSUs, and both citizens and inspectors may provide services for collector ($Coll$) and assessor ($Asses$) roles, hence we have the following VSUs: VSU_{Se}, VSU_{Cz}^{Coll}, VSU_{Cz}^{Asses}, VSU_{In}^{Coll}, VSU_{In}^{Asses}. Let us assume that the Infrastructure Management Platform (IMP), the Stream Analytic Server (SAS), the Human-based Computing Platform (HCP), and Sensors Network (SN) are static sets of units. Then, given a collective dependency graph in Fig. 2, we can obtain a set of possible ESTs as follows:

- IMP, SAS, VSU_{Se}, SN
- $IMP, HCP, VSU_{Cz}^{Coll}, VSU_{Cz}^{Asses}$

Algorithm 1 EST Generation Algorithm

```
1: function GENERATEEST(dependencyGraph)
2:     ESTList ← ∅
3:     for all root ∈ dependencyGraph.getRoots() do
4:         est ← GENERATE(root)
5:         ESTList ← COMBINE(ESTList, est)
6:     return ESTList
7:
8: function GENERATE(node)
9:     ESTList ← node.generateResourcesEST()
10:    for all branch ∈ node.getBranches() do
11:        if branch.isAlternating() then
12:            childESTList ← ∅
13:            for all altNode ∈ branch.getAltNodes() do
14:                alternateEST ← GENERATE(altNode)
15:                childESTList.add(alternateEST)
16:        else
17:            branchNone ← branch.getNode()
18:            childESTList ← GENERATE(branchNone)
19:        ESTList ← COMBINE(ESTList, childESTList)
20:    return ESTList
21:
22: function COMBINE(list1, list2)
23:    if list1 = ∅ then
24:        return list2
25:    else if list2 = ∅ then
26:        return list2
27:    else
28:        resultList ← ∅
29:        for all s1 ∈ list1 do
30:            for all s2 ∈ list2 do
31:                resultList.add(s1 + s2)
32:        return resultList
```

- $IMP, HCP, VSU_{Cz}^{Coll}, VSU_{In}^{Asses}$
- $IMP, HCP, VSU_{In}^{Coll}, VSU_{Cz}^{Asses}$
- $IMP, HCP, VSU_{In}^{Coll}, VSU_{In}^{Asses}$

For an HCU to execute a task reliably, at least one EST must successfully accomplish the task. The failures of all possible ESTs result to the failure of the HCU to execute the task. Therefore, given a task t and its set of EST $\mathcal{S}_t$, we can define the reliability to execute the task t, R^t, as the probably of having at least one EST of $\mathcal{S}^t$ working properly:

$$R^t = Pr\{\exists EST, EST \in \mathcal{S}^t \wedge EST \text{ works properly}\}.$$

Let E_i be the event that $EST_i \in \mathcal{S}^t$ operates properly, then the reliability to execute the task t is given by

$$R^t = Pr\left\{\bigcup_{i=1}^{|\mathcal{S}^t|} E_i\right\}. \tag{2}$$

The calculation of probability of such events should consider the fact that E_i may be correlated, i.e., the inclusion of VSUs in ESTs are not exclusive. Several works, e.g., [27], [28], propose some techniques to calculate such probability.

V. Implementation and Experiments

A. Implementation

We have prototyped our reliability framework and integrated it into our platform, *Runtime and Analytics for Hybrid Computing Systems* (RAHYMS) [1]. This platform is open-sourced and implemented using Java and provides tools for simulating hybrid collaboration based on GridSim toolkit [29].

The platform features the following capabilities:

- Simulate a pool of machine-based and human-based compute units with statistically distributed profiles, e.g., to resemble a crowdsourcing marketplace or a social network.
- Simulate the generation of task requests with their associated requirements, e.g., skill requirements, cost limit, deadline, and units' connectedness.
- Provide various strategies for the formation of HCUs. Several formation strategies are provided based on our previous work [20], e.g., greedy approach, and optimized formations using Ant Colony Optimization. Further formation strategies can be implemented and plugged into the platform.
- Provide a tool for performing reliability analysis on the running HCU, which can then be aggregated to represent overall system's reliability.

The tool can be configured using user-defined JSON configurations, which allow different scenarios to be simulated. These configurations can be specified to govern the generation of the pool of compute units with profiles that are statistically distributed. For example, in the infrastructure maintenance scenarios as depicted in Fig. 1, we simulate a pool of citizens participating in the process for detecting infrastructure breakdown. In this process, some citizens can be nominated to have data collection capability while others may be credited for their assessment skills. Fig. 4 shows a snippet of a configuration for generating citizens as compute units, which provide *DataAssessment* service and have a set of generated skills and properties.

Also, configurations for the tasks generator allow customization of the task requests, e.g., to define roles and collective dependencies. They can also be used to define statistically distributed functional and non-functional requirements of the tasks. During the formation of the HCU collective for executing the task, these requirements will be matched with the profiles of the generated compute units. We include some pre-configured scenarios in the above-mentioned source code repository.

B. Experiments

In the following, we apply our model by exemplifying some reliability analyses on different scenarios. Our goal here is to show how our model can be used to measure the reliability of task executions, R_{task}, and how we can get insights from the reliability analysis. The purpose of this experiment is not to model a true-to-life scenario. However, we want to show how

[1] https://github.com/tuwiendsg/RAHYMS

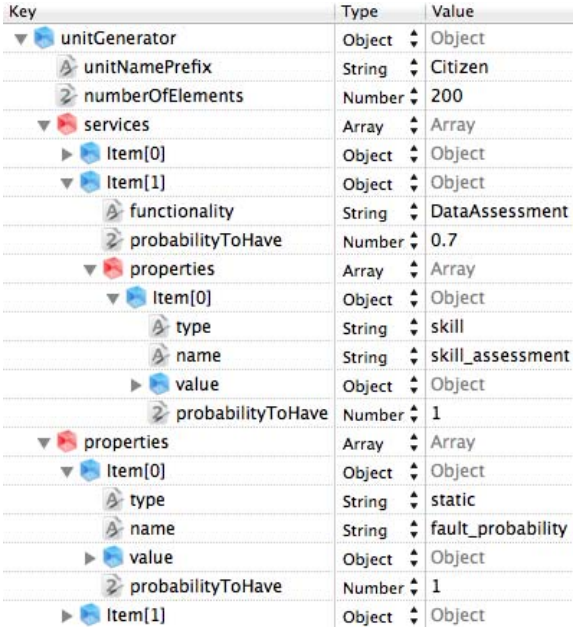


Fig. 4: Configuration snippet for units generator

our tool can be used to model and tune different scenarios, and how the reliability analysis can be used as a feedback for improvements.

In our experiments, we use the infrastructure maintenance scenario as depicted in Fig. 1 and Fig. 2. We define the task in our experiments as the task for sensing facility breakdown. Each instantiation of a task is implicitly associated with an occurring breakdown event. A task execution is said to be successful when the breakdown is correctly detected.

Without loss of generality, in this experiments we model the probability of failed executions of each individual unit in discrete time space using the geometric distribution. Let p be the failure probability of executions by an individual unit. Assuming that p is constant and independent of the execution time, we could have

$$\begin{aligned} f(k) &= (1-p)^{k-1}p \quad \text{, and} \\ F(k) &= 1-(1-p)^{k} \quad \text{, therefore} \\ R(k) &= (1-p)^{k}. \end{aligned} \tag{3}$$

An estimation of the distribution parameter p can be derived from the task execution data of each individual units. For a known unit u, let $e = e_1, e_2, ..., e_n$ be a set of result execution samples. The value of e_i may be a binary, 1 for a successful execution and 0 for a failure execution, or a floating number $[0..1]$, which represents the result quality of the execution. The distribution parameter p of unit u can be estimated by

$$\hat{p_u} = 1 - \frac{\sum_{i=1}^{n} e_i}{n}. \tag{4}$$

a) Experiments Setup: Assuming the Infrastructure Management Platform (IMP), the Stream Analytic Server (SAS), the Human-based Computing Platform (HCP), and Sensors Network (SN) are static sets of units (Fig. 2), we

Scenarios	Goals: to study	Configurations	Variants
Exp. 1	reliability changes over time or executions	$N_{citizens} = 200$ $N_{inspectors} = 10$ $N_{sensors} = 50$ $\bar{p}_{citizens} = 0.3$ $\bar{p}_{inspectors} = 0.05$ $\bar{\lambda}_{sensors} = 0.02$ $\lambda_{DR} = 0.001$	$k = [1..10000]$
Exp. 2	effect of different sizes of resources pools on reliability	$\bar{p}_{citizens} = 0.3$ $\bar{p}_{inspectors} = 0.05$ $\bar{\lambda}_{sensors} = 0.02$ $\lambda_{DR} = 0.001$ $k = 2500$	$N_{citizens} = [0..300]$ $N_{inspectors} = [0..20]$ $N_{sensors} = [0..250]$
Exp. 3	effect of different HCU formation strategies on reliability	$N_{citizens} = 200$ $N_{inspectors} = 10$ $N_{sensors} = 50$ $\bar{p}_{citizens} = 0.3$ $\bar{p}_{inspectors} = 0.05$ $\bar{\lambda}_{sensors} = 0.02$ $\lambda_{DR} = 0.001$ $k = 2500$	strategies: - uniform distribution - fastest response - greedy (cost optimized)

TABLE I: Experiment scenarios

focus our experiments on studying the variability of VSUs configuration of machine-based sensors, as-well-as human citizens, and human inspectors in fulfilling *sensor*, *collector*, and *assessor* roles. Citizens and inspectors may be assigned to the collector and assessor role, while machine-based sensors are assigned to fulfill the sensor role. Each machine unit (i.e., a sensor) has a randomly generated continuous failure rate λ, and the reliability at a particular time t is given by $R(t) = e^{-\lambda t}$ [21]. Each human unit (i.e., a citizen or an inspector) has a randomly generated probability of failure p, and the reliability at a particular execution k can be measured using Equation 3. We perform three sets of experiments to study different aspects of reliability in HCUs with different configurations as shown in Table I. These experiments are discussed as follows.

b) Experiment 1: In this experiment, we study how the reliability of the task executions changes over time. We generate a fix number of units and generate statistically distributed failure probabilities and failure rates for each units as shown in Table I. We employ fix reliability configurations: for citizens, when they are assigned to a task, at least 2 of 3 assigned citizens must be working properly; for inspectors and sensors, we require only 1 working unit. We simulate the detection and reconfiguration of faulty units using software-based components with a failure rate $\lambda_{DR} = 0.001$. We generate 10,000 tasks with a task rate of 30 tasks per time unit. For each task instance, members of the VSUs are then assigned and activated for executing the task.

During the experiment we measure the average reliability of individual units, as well as the reliability of VSUs and the aggregated reliability of the task executions, R_{task} (given by equation 2), as shown in Fig. 5. The reliability of VSUs are affected by the number of units as well as the reliability of each units. $R_{VSUcollectors}$ is higher than $R_{VSUassessors}$ because in our experiments the number of generated units qualified for doing data collection task is around 50% more than the the number of data assessment qualified units. Furthermore, the average reliability of sensor units is calculated as a function of t; hence, the slope of its reliability is also affected by the task rate, here (as well as in other experiments) we use $t = \frac{k}{30}$, i.e., 30 tasks per time unit.

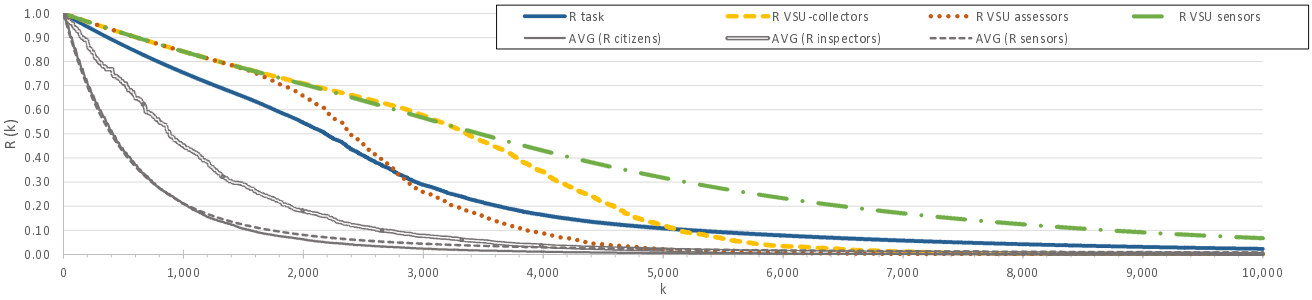


Fig. 5: Reliability on task executions, $R(k)$

The reliability of VSUs are significantly higher then the average reliability of individual units, since VSUs employ dynamic/hybrid redundancy. The reliability of any computing systems always decreases over time. However, the decrement slope of an VSU is not as steep as its individual units. On low k value, the reliability of the whole system and VSUs are mainly affected by R_{DR}. In fact, in this setup if we simulate a perfect detection and reconfiguration component, i.e., $\lambda_{DR} = 0$, we will have $R_{task} \simeq 1$ until $k \approx 1000$. And R_{task} will drop below the average reliability of all individual units when $\lambda_{DR} > 0.0058$.

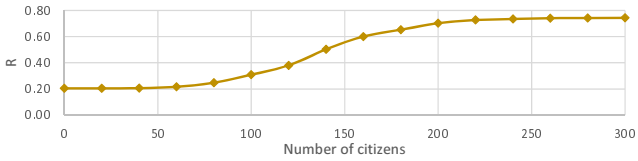
Therefore, to design reliable HCUs, we posit that the application designer should pay attention not only to the reliability of individual units, but also consider the structure of standby units, e.g., how they can be effectively discovered, and also the size of the available standby units. Furthermore, it is also important to design a highly reliable detection and reconfiguration component, otherwise the redundancy structure of the standby units will render useless.

c) Experiment 2: Compute units in HCUs may come from different pools of resources with varying quality and sizes. Our next experiments study how R_{task} is affected by the size of resources pools. Such experiments may assist the resource platform providers to decide whether adding more resources is beneficial to improve the reliability of the HCUs.

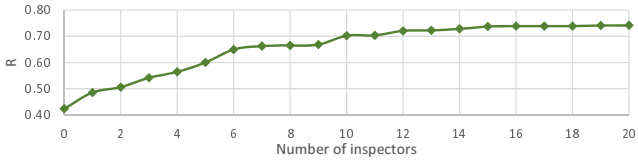
We use the same values of p and λ, and employ the similar reliability configurations as the previous experiments. We experiment with 2500 generated tasks, i.e., $k = 2500$. Fig. 6 depicts how R_{task} changes with the varying number of citizens, inspectors, and sensors.

In these figures we can see that R_{task} values have upper limits due to the fact that other unit types (as well as other static components) are not being improved. By studying these R_{task}, we could recognize the sweet spots on which adding more units could effectively increase R_{task}. For example, the increment of the number of citizens between 80 to 220 on our setup effectively improve R_{task}, while adding more citizen units beyond 220 is fruitless. Hence, the importance of adding more units to increase the reliability (e.g., recruiting more citizens) must be balanced out with the recruitment cost.

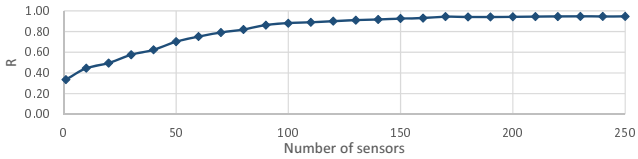
As shown on Fig. 6a and Fig. 6b, the effect on R_{task} is greatly determined by the failure rate or failure probability. We require less additional recruitments of inspectors to improve the reliability due to $\bar{p}_{inspectors} \ll \bar{p}_{citizens}$. However, the



(a) Reliability on varying number of citizens



(b) Reliability on varying number of inspectors



(c) Reliability on varying number of sensors

Fig. 6: Reliability on varying size of resources pools

structure of the corresponding VSUs also impacts the changes of R_{task}. In our setup, the role of the dedicated inspectors in $VSU_{collectors}$ and $VSU_{assessors}$ can also be replaced by citizens. Hence, we don't need many inspectors additions, compared to sensors additions, to improve R_{task}.

d) Experiment 3: Different systems may employ different strategies for the formation of HCUs collaboration, which eventually affect the reliability of the HCUs as well as their non-functional properties. Here we experiment with three formation strategies: (i) the *uniform distribution* strategy, where the tasks are uniformly assigned to qualified units, (ii) the *fastest response* strategy, where the tasks are assigned to the qualified units that provide fastest response times (e.g., depending on the unit's job queue and performance rating), and (iii) the *greedy* strategy, where we employ a greedy optimization algorithm to minimize the execution cost of the HCU [20]. The performance rating and the execution cost of each individual unit are statistically distributed during units generation based on the generator configurations.

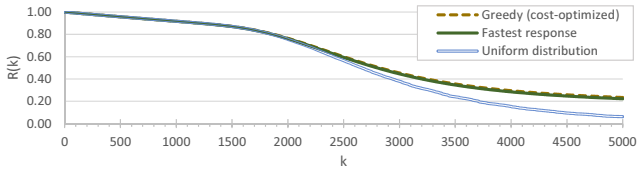


Fig. 7: Reliability on different HCU formation strategies

HCU Formation Strategies	$avg(cost)$	$avg(response\ times)$
Uniform distribution	7.20	13.585
Fastest response	7.92	11.775
Greedy (cost-optimized)	6.60	12.276

TABLE II: HCU cost and response times

We use similar configurations as in the first experiment with 5000 tasks. As we can see on Fig. 7, the reliability of the *fastest response* strategy and the *greedy (cost-optimized)* strategy are similar. However, we observe that the reliability of the *uniform distribution* strategy is lower than the other two, especially on higher k. This is due to the fact that the *fastest response* strategy and the *greedy (cost-optimized)* strategy tend to select a particular set of units with better performance rating and cheaper cost, respectively; hence, they yield more standby units with less utilization. On the individual level, a unit with less utilization has a higher reliability for the next assigned task, i.e., for each unit i, $R_i(k_i) > R_i(k_i+x), \forall x \in \mathbb{R} \mid x > 0$. Therefore, the reliability of the VSUs will also be higher, and consequently that yields higher HCU reliability.

Furthermore, different HCU formation strategies also result different non-functional properties of the formed HCU. In Table II, we show the average cost and response time of the HCUs obtained from the three strategies. Here, the cost is defined as the sum of the execution cost of all members units in each HCU, while the response time is defined as the duration since the task is assigned to the created HCU until all members units of the HCU finish their roles. In these experiments, the *greedy (cost-optimized)* strategy provides 16.70% and 8.35% cheaper HCUs compared to the *fastest response* strategy and to the *uniform distribution* strategy respectively. For the response time, the HCUs provided by the *fastest response* strategy perform the tasks 13.32% and 4.08% faster compared to the HCUs provided by the *uniform distribution* strategy and the *greedy (cost-optimized)* strategy respectively.

Each problem domain has its own requirements with respect to the non-functional properties. Such analysis with different HCU formation strategies help application designers to decide the desirable trade-offs between the reliability and other non-functional properties gained by certain strategies.

VI. Related Work

a) Quality-Aware Human-Based Computing: The quality control has become one of the challenging obstacles in the human-based computing, especially in the advent of crowdsourcing models [9]. Many approaches have been proposed to improve the reliability of human-based computing systems, especially with respect to the quality of results, e.g., [30], [31], [11], [30]. These works deal with the reliability-improving approaches for simple tasks that can be assigned to individuals. Our work focuses on more complex tasks executed by a collective of humans and machines, and how to measure the reliability of the task execution.

In the context of processes with human tasks, several techniques have been proposed to measure reliability property. In [7] and [8], the authors proposed a mathematical model to compute the quality of services by applying reduction rules to a workflow until an atomic task is obtained. This value only provides an estimation of reliability for the next task execution, while our approach provides a mechanism to estimate the reliability in a discrete time space. Furthermore, our framework allows reliability analysis for hybrid compute units obtained from a large pool of resources.

b) Reliability of Human Resources: Several techniques for Human Reliability Analysis (HRA) have been developed in other disciplines such as safety and ergonomic engineering using a probabilistic model, e.g., [5], or using a cognitive theory, e.g., [6]. More advance approaches, e.g., [32], propose techniques to measure human performance reliability in real-time and on-line manner. Several works have also been conducted to formally model human behavior in computing systems, e.g., [33]. In our proposed framework, we use a technique to measure the reliability of individual units on a task-basis using probability distributions with certain parameters. These parameters can be obtained from these HRA techniques.

c) Reliability of Large Scale Systems: Research on the reliability of large scale systems, such as grid systems, e.g., [28], [34], and cloud services, e.g., [35], [36], has gained a lot of interest. These works proposed some models to analyze the reliability of hardware and software systems and proposed techniques to improve systems' fault tolerance. Some algorithms have also been proposed, e.g., [27], [28], to solve non-trivial reliability equations as discussed in Section IV-B2.

Several techniques have also been proposed to improve the reliability of large scale systems. For example, in [37], a combination of game-theoretic approach and classical voting approach is proposed to improve the reliability of Internet-based master-worker computations, so that to enable a master to reliably obtain a result despite of the coexistence of unreliable workers.

To the best of our knowledge, currently there are no published works that provide models for the reliability analysis of hybrid human-machine systems as proposed in this paper.

VII. Conclusions and Future Work

In this paper we present our approach to analyze the reliability of collaborations that consists of human- and machine-based compute units to execute tasks. Our framework is capable to deal with the reliability measurement of collaboration collectives, i.e., Hybrid Compute Units (HCUs), which are dynamically provisioned on-demand using various strategies from large-scale Internet-based human and machine resources pools. We first discuss models for measuring the reliability of individual units on a task basis. Then we present the underlying models of HCUs. Based on these models we propose a framework to measure the reliability of HCU.

Our approach centers around the structure of collaborations modeled using the collective dependency. Such dependency

can be defined by the application designer from the ground up, or it can also be inferred from the application design, e.g., from the workflow structure. This implies that our approach is suitable for collaborations that have known structures. We retain problems of analyzing the reliability of unstructured collaborations as future works.

We present our HCU simulation tool, and exemplify our reliability analysis approach using infrastructure maintenance scenarios. The results of our experiments show that our framework is beneficial for hybrid collaborative application stakeholders, e.g., application designers, resource providers, and task owners, to measure the reliability of the collaborations and to obtain insights for improving the collaboration quality. Although our tool currently focuses on off-line reliability simulation, we aim to have an online reliability measurement tool based on the proposed reliability analysis framework.

Our work presented in this paper is part of our ongoing research on dependable hybrid human-machine computing. Future works include modeling other dependability metrics such as availability, performance, and quality of results.

REFERENCES

[1] A. Agrawal *et al.*, "WS-BPEL extension for people (BPEL4People) version 1.0," 2007.

[2] Y. Pan and E. Blevis, "A survey of crowdsourcing as a means of collaboration and the implications of crowdsourcing for interaction design," in *Collaboration Technologies and Systems (CTS), 2011 International Conference on*. IEEE, 2011, pp. 397–403.

[3] H.-L. Truong, H. K. Dam, A. Ghose, and S. Dustdar, "Augmenting complex problem solving with hybrid compute units," in *Service-Oriented Computing–ICSOC 2013 Workshops*. Springer, 2014, pp. 95–110.

[4] I. Eusgeld, F. Freiling, and R. H. Reussner, *Dependability Metrics*. Springer, 2008, vol. 4909.

[5] J. Williams, "Heart–a proposed method for assessing and reducing human error," in *9th Advances in Reliability Technology Symposium, University of Bradford*, 1986.

[6] E. Hollnagel, *Cognitive reliability and error analysis method (CREAM)*. Elsevier Science, 1998.

[7] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut, "Quality of service for workflows and web service processes," *Web Semantics*, vol. 1, no. 3, pp. 281–308, 2004.

[8] P. Bocciarelli, A. D'Ambrogio, A. Giglio, and E. Paglia, "Simulation-based performance and reliability analysis of business processes," in *Simulation Conference (WSC), 2014 Winter*. IEEE Press, 2014, pp. 3012–3023.

[9] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. Motahari-Nezhad, E. Bertino, and S. Dustdar, "Quality control in crowdsourcing systems: Issues and directions," *IEEE Internet Computing*, vol. 17, no. 2, 2013.

[10] P. Ipeirotis, F. Provost, and J. Wang, "Quality management on amazon mechanical turk," in *Proceedings of the ACM SIGKDD workshop on human computation*. ACM, 2010, pp. 64–67.

[11] S.-W. Huang and W.-T. Fu, "Enhancing reliability using peer consistency evaluation in human computation," in *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 2013, pp. 639–648.

[12] OMG, "Business process model and notation 2.0," 2011.

[13] "Amazon mechanical turk," Website, 2015, http://www.mturk.com/.

[14] C. Petrie, "Plenty of room outside the firm [peering]," *Internet Computing, IEEE*, vol. 14, no. 1, pp. 92–96, 2010.

[15] J. G. Breslin, A. Passant, and S. Decker, "Social web applications in enterprise," in *The Social Semantic Web*. Springer, 2009, pp. 251–267.

[16] S. Dustdar and W. Schreiner, "A survey on web services composition," *International journal of web and grid services*, vol. 1, no. 1, pp. 1–30, 2005.

[17] F. Giunchiglia, V. Maltese, S. Anderson, and D. Miorandi, "Towards hybrid and diversity-aware collective adaptive systems," 2013.

[18] K. Benouaret, R. Valliyur-Ramalingam, and F. Charoy, "Crowdsc: Building smart cities with large-scale citizen participation," *Internet Computing, IEEE*, vol. 17, no. 6, pp. 57–63, 2013.

[19] Z. Zheng and M. R. Lyu, "Collaborative reliability prediction of service-oriented systems," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*. ACM, 2010, pp. 35–44.

[20] M. Z. Candra, H.-L. Truong, and S. Dustdar, "Provisioning quality-aware social compute units in the cloud," in *Service-Oriented Computing*. Springer, 2013, pp. 313–327.

[21] I. Eusgeld, B. Fechner, F. Salfner, M. Walter, and P. Limbourg, "Hardware reliability," *Dependability metrics*, pp. 59–103, 2008.

[22] I. Eusgeld, F. Fraikin, M. Rohr, F. Salfner, and U. Wappler, "Software reliability," *Dependability metrics*, pp. 104–125, 2008.

[23] I. Koren and C. Krishna, *Fault-tolerant systems*. Morgan Kaufmann, 2010.

[24] B. Satzger, H. Psaier, D. Schall, and S. Dustdar, "Stimulating skill evolution in market-based crowdsourcing," in *Business Process Management*. Springer, 2011, pp. 66–82.

[25] A. Kulkarni, M. Can, and B. Hartmann, "Turkomatic: automatic recursive task and workflow design for mechanical turk," in *ACM SIG CHI '11*. ACM, 2011.

[26] M. Maybury, R. D'Amore, and D. House, "Expert finding for collaborative virtual environments," *Communications of the ACM*, vol. 44, no. 12, pp. 55–56, 2001.

[27] X. Zang, H. Sun, and K. Trivedi, "A bdd-based algorithm for reliability analysis of phased-mission systems," *Reliability, IEEE Transactions on*, vol. 48, no. 1, pp. 50–60, 1999.

[28] Y. Dai, M. Xie, and X. Wang, "A heuristic algorithm for reliability modeling and analysis of grid systems," *Systems, Man and Cybernetics*, vol. 37, no. 2, pp. 189–200, 2007.

[29] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and computation: practice and experience*, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[30] L. R. Varshney, A. Vempaty, and P. K. Varshney, "Assuring privacy and reliability in crowdsourcing with coding," in *Information Theory and Applications Workshop (ITA), 2014*. IEEE, 2014, pp. 1–6.

[31] R. Blanco, H. Halpin, D. M. Herzig, P. Mika, J. Pound, H. S. Thompson, and T. Tran Duc, "Repeatable and reliable search system evaluation using crowdsourcing," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 2011, pp. 923–932.

[32] W. J. Kolarik, J. C. Woldstad, S. Lu, and H. Lu, "Human performance reliability: on-line assessment using fuzzy logic," *IIE transactions*, vol. 36, no. 5, pp. 457–467, 2004.

[33] R. Rukšėnas, J. Back, P. Curzon, and A. Blandford, "Formal modelling of salience and cognitive load," *Electronic Notes in Theoretical Computer Science*, vol. 208, pp. 57–75, 2008.

[34] S. Guo, H. Huang, Z. Wang, and M. Xie, "Grid service reliability modeling and optimal task scheduling considering fault recovery," *Reliability*, vol. 60, no. 1, pp. 263–274, 2011.

[35] T. Thanakornworakij, R. F. Nassar, C. Leangsuksun, and M. Păun, "A reliability model for cloud computing for high performance computing applications," in *Euro-Par 2012*. Springer, 2013, pp. 474–483.

[36] N. Yadav, V. Singh, and M. Kumari, "Generalized reliability model for cloud computing," *International Journal of Computer Applications*, vol. 88, no. 14, pp. 13–16, 2014.

[37] E. Christoforou, A. Fernandez Anta, C. Georgiou, M. Mosteiro *et al.*, "Algorithmic mechanisms for reliable master-worker internet-based computing," *Computers, IEEE Transactions on*, vol. 63, no. 1, pp. 179–195, 2014.