SLA-Based Management of Human-Based Services in Business Processes for Socio-Technical Systems

Mirela Riveni[1(✉)], Tien-Dung Nguyen[2], and Schahram Dustdar[1]

[1] Distributed Systems Group, TU Wien, 1040 Vienna, Austria
{m.riveni,dustdar}@infosys.tuwien.ac.at
[2] International University National University,
Ho Chi Minh city, Vietnam
ntdung@hcmiu.edu.vn

Abstract. Research and industry are focused on Collective Adaptive Systems (CAS) to keep up with the social changes in the way we work, conduct business and organize our societies. With advances in human computing, we can strengthen these systems with a crucial type of resources, namely *people*. However, while other resources in CAS are managed by Service Level Agreements (SLAs) in an automated way, human-based services are not. Considering the fact that not much work has been reported on SLAs in settings where human computation is an integral part of a process, this paper investigates SLAs for social computing, including non-functional parameters for human-based services, as well as privacy constraints. We investigate and evaluate SLA changes at runtime, which in turn influence elastic runtime social-collective adaptations, in processes designed to allow for elastic management of social collectives.

Keywords: Human-centric processes · Collective Adaptive Systems
Service Level Agreements · Human computation
Social Compute Units

1 Introduction

Traditional information systems are not good enough to keep up with the evolving societal needs, including the ever bigger complexities and changing dynamics in how work is organized, businesses are run and societies are governed. Thus, researchers and industry have began to focus on Collective Adaptive Systems (CAS), which are systems that include multiple resource-collectives (or sub-systems). They include heterogeneous type of resources and services that inter-operate and provide seamless service. Typical resources can be software services, Cloud services, different agents, sensors and Internet of Things (IoT) services. However, there is a crucial component of collectives that will make a key difference in CAS, namely people. With the advance of human computation in

E. Teniente and M. Weidlich (Eds.): BPM 2017 Workshops, LNBIP 308, pp. 361–373, 2018.
https://doi.org/10.1007/978-3-319-74030-0_27

general and social computing in particular, today it is possible for human-based services/resources to be part of CAS. Examples of these systems are described in [1,3,19].

While other types of resources in CAS can, and are managed by automated Service Level Agreements (SLAs), human-based services are not. People providing their skills as services online are mainly managed by standard (human-language) contracts or no contracts at all. Nevertheless, worker performance should be monitored and managed in processes in an automated way, so that task management can be efficient, and both customer and worker needs can be fulfilled. Thus, the challenges that we tackle in this paper are modeling SLAs for human computation and process-based SLA adaptation mechanisms. To approach them, we utilize the term Social Compute Units (SCUs) introduced in [4]. SCUs represent human collectives, who act as resources and/or provide their skills as online services for processing tasks that can not be processed by software. We name individuals who work within these collectives, Individual Compute Units (ICUs). SCUs are formed mostly on customers' request and with customer-set constraints but they can also be formed voluntarily and in an ad-hoc way. They are invoked to reach a certain task-oriented goal, with a specific quality and in a specific context. Thus, they have their own execution lifecycle that we have previously discussed in [14].

Dustdar et al. in [5] have introduced elastic processes for cloud and human computation, and their fundamental principles including *resource elasticity*, *cost elasticity* and *quality elasticity*. In the particular context of elasticity for collectives such as SCUs, SLAs play a crucial role in managing their execution and controlling the quality of the offered services and thus the returned results. However, as far as we know, SLAs in human computation in general are not explicitly investigated in the context of processes with which collectives are managed elastically together with SLAs based on specific metrics designed specifically for collectives such as SCUs. In the paper at hand we investigate how SLAs fit in an SCU provisioning platform. In addition, we describe and present our experiment results of an SCU elastic management strategy based on SLAs, which is focused on cost optimization of SCUs while they are being elastically adapted at runtime. The contribution of this work is to show that the design and implementation of social-computing processes need to allow for elasticity regarding resources, customer requirements, and worker/ICU profile parameters, in order to be efficient. We justify this along with the justification for the need to design systems that provide the possibility of having elastic SLAs, the changes of which trigger elastic social-collective adaptations.

The remainder of this paper is organized as follows: In Sect. 2 we describe real-world motivation scenarios, and discuss an SCU provisioning-platform model including SLAs. In Sect. 3 we discuss parameters for modeling SLAs for human computation. In Sect. 4 we describe a proof-of-concept prototype demonstrating a process-based SCU adaptation, based on SLA changes. Section 5 presents related work, and we conclude the paper in Sect. 6.

2 Motivation and SCU Environment Setting

2.1 Illustrative Scenarios

Facility Management. A typical example of a Collective Adaptive System (CAS) is a smart city. Let us consider the case of facility management in a smart-city. To maintain multiple smart buildings and manage utilities effectively, such as power management, water management, temperature, and security management, the managing enterprise/organization utilizes applications, platforms and infrastructure, thus working with different software services, Cloud platforms and infrastructures, as well as different sensors and monitoring equipment (IoT). Data gathered from monitoring equipment needs to be analysed in real time so that actions can be taken in real time. The following human-based collective-types (SCUs) could included in the facility-management process: (a) *SCUs for data-analysis*, (b) *SCUs with domain experts*, and c) *SCUs with on-site technical-experts*.

Language Translation. Recently we have seen an increase in online platforms for translations e.g., documents, books, documentary(film) subtitles and available languages for software applications. Let us consider a publishing company that needs to translate a book within a short time in a specific language for which it does not have suitable translators or contractors. Today it is feasible to hire multiple translators online. Assuming that the book is submitted to be translated as a task to a Socio-technical CAS, it is possible that the platform first uses software translation-services as a first round of translating and then assigns appropriate people/ICUs to fix the software translation, and lastly it assigns an ICU or a collective of ICUs to do the final editing. Another possibility, is that the system just assigns the translation task to a set of ICUs (an SCU) from start and then in a second iteration assigns the translated material to a set of ICUs for reviewing and final editing. Two SCUs would be formed in this scenario: (a) a *language-translation SCU*, and (b) a *reviewing and final-editing SCU*.

The aforementioned scenarios validate the need of humans-in-the-loop within socially-enhanced CAS. The main concerns in these type of systems are the type of SCUs that are needed, with which capabilities, and more importantly, how can they be process-managed at runtime to keep up with clients' requests and requirements as well as worker needs and conditions. Meeting customer requirements as well as worker conditions in human computation platforms can be done more efficiently if there is a negotiation defining clear terms and conditions in the form of SLAs. To the best of our knowledge, there is no research in SLAs for human-computation specifically, which goes along with research on how to manage online work based on metrics. The current state of the art engages people in human-computation based on natural language contracts or no contracts at all. However, research in managing mechanisms, such as, collective formation, task delegations, proactive and elastic adaptation of human collectives online, need to address the challenge of SLAs because of the unpredictable nature of humans. To clarify, we identify some of the core characteristics of human-based resources that would impact SLA specification and management as follows:

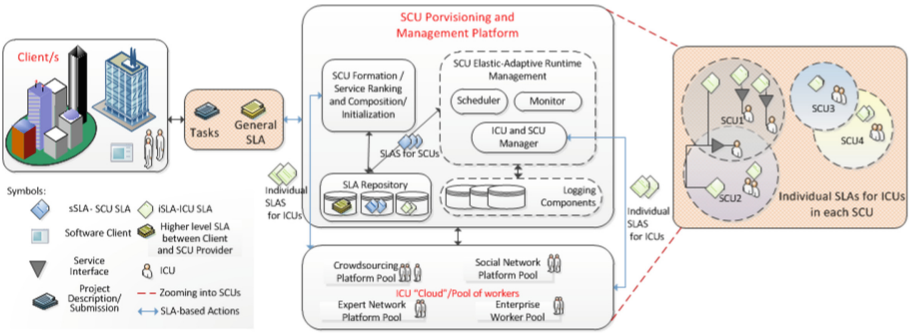

Fig. 1. A conceptual overview of an SLA-based ICU-SCU Provisioning Platform

- Human performance is a function of monetary or non-monetary *rewards* and *penalties*. Thus, human misbehavior can be prevented or managed by *incentives* respectively *sanctions*.
- People are *paid/rewarded* by the service provisioning platform as an intermediary and/or by the platform-customers; the platform in turn is paid by its customers. Thus, in human computation, contract negotiation is hierarchical, between a customer and the service-provisioning platform, and between the platform and people who provide their skills online.
- People need to be managed with their consent and guided by human-rights principles. Hence, *privacy* is a crucial element to be considered during system design, which is unfortunately often overlooked.

2.2 SCU Computational Environment Setting

In Fig. 1 we give an abstract overview of an SCU provisioning platform. The figure shows that SLAs are established between a customer and the platform for a specific project that requires a specific SCU, so there are SLAs for SCUs. The platform can run a ranking algorithm to rank ICUs from an ICU Pool (a proprietary one, or from another pool that it has access to) according to the requirements within the SCU SLAs. In the next step, a negotiating component negotiates individual SLAs with ICUs in the ranked list and forms an appropriate SCU. All negotiations between all parties should be automated so that SLAs can be automatically changed at runtime if customers or ICUs decide to change specific SLA parameters. From the aspect of an SCU execution that is controlled by SLAs, an SCU may elastically be adapted, such that: (a) *compliance with runtime changes in customer and/or worker requirements is achieved*, and (b) *an SLA violation is prevented or managed.* A customer might want to change his/her requirements at runtime, in this case the platform should enable automated SLA changes at run-time. On the other hand, when adapting an SCU such that an SLA violation is prevented, e.g., by substituting a misbehaving ICU with another more appropriate one, from the customers requirements perspective the

Table 1. Notation and description of basic parameters for SCU SLAs

Properties	Description	Value
Consistency	Executed tasks as promised, e.g., via feedback messages	[0..1]
Socio-technical trust score [15]	Includes other metrics: social scores (subjective), and monitored performance metrics(objective), e.g.,: productivity, effort, success rate, nr. of successfully executed delegations, etc	[0..1]
Reputation	An average of the performance and social scores of ICUs across all SCUs in which it was invoked	[0..1]
Average queue time	An average of task waiting time in an ICUs queue; for tasks with specific skill requirements and comparable complexities (e.g., a page for translation)	[0..1]
Result timeliness	An average of on-time result-delivery score of ICUs	%
Availability	ICU availability (as time intervals.)	hour
Budget	The max price that the customer may pay	currency

same SLA terms and conditions for the substituted ICU must be valid. However, the new ICU may have its own requests. In these cases negotiations should be conducted at run-time as well. In consequence to these two cases, to best reflect human behavior we argue the need of *elastic SLAs*, because of the mere nature of people and their working dynamics. Thus a provisioning platform supporting elastic SCUs governed by SLAs, should also support elastic SLAs.

3 Modeling SLAs for SCUs

3.1 Human-Centric Properties and Metrics

Non-functional properties (NFPs) as key indicators of service performance are the parameters that are the most important in SLA negotiations. In Table 1 we list relevant metrics from our previous work [14] and some new ones that we have identified as crucial for SCUs, and which can be used as Service Level Objectives (SLOs) in SLAs. We denote an ICU as a member of an SCU with s_i. An ICU as a member of SCU, has the following set of properties, $ICU^{s}_{prop} = \{id, skill, metrics, iSLA\}$, while an SCU is defined as $SCU_{exec} = \{state, structure, metrics, sSLA\}$. For the mathematical definitions of *Socio-Tecnical Trust Score* and *Reputation* metrics we advise you to check our previous work presented in [15]. Currently, platforms in industry that involve social-computing, all have a common flaw, the lack of privacy. Apart form performance parameters for workers, the inclusion of privacy-related constraints for systems will provide systems that respect both workers and customers. While we leave the investigation of privacy-related parameters for future work, identified privacy-related conditions that can be included in SLAs are: *software to be used for*

task execution and communication, time allowed to keep the collected data (e.g. location), audit periods for third-parties to check if obligations are fulfilled, time period until an audit report is received, deadline for informing users after a security breach has happened, and privacy-breach penalties for the system.

3.2 Penalties

We consider two cases regarding penalties: (1) when SLA parameters are changed at runtime, and (2) when there has been an SLA violation. We have previously defined a metric that we call Socio-technical Trust for ICUs (in [15]). This metric is comprised of two complex metrics, namely (1) *Performance trust*(PT) that we define and calculate with the help of other performance-measuring metrics, and (2) Membership Collaboration Trust Score (MCTS), which we defined as a weighted sum of satisfaction scores from each worker to another within the same collective, representing a *Social trust* metric in our model. The Socio-technical Trust score is calculated as: $STT(s_i) = w_{pt} * PT(s_i) + w_{mcts} * MCTS(s_i)$ - our equation from [15], where s_i denotes an ICU and $w(x)$ is the weight of a metric x. Now we extend this formula and define it as: $STT(s_i) = w_{pt} * PT(s_i) + w_{mcts} * MCTS(s_i) + w_{cps} * CPS(s_i)$, where CPS is the satisfaction score comprised of a satisfaction score from the customer and the platform (measuring consistency of behavior). We define CPS as: $CPS(s_i) = w_{css} * CSS(s_i) + w_{pss} * PSS(s_i)$, where CSS is Customer Satisfaction Score and PSS is Platform Satisfaction Score.

Let us take a concrete example, when an SLA change occurs at runtime, e.g., a customer requests a cost-change for a specific skill-type. In this case, ICUs already within the collective can be sent a notification message for the newly changed fee. In this example, those ICUs that are already within the SCU and do not accept the changes are excluded from the SCU and their Platform Satisfaction Score (PSS) is lowered by a preset value ∂ as a penalty (lines 10–11 in Algorithm 1). The tasks for the changed cost are executed by other appropriate ICUs that are newly included in the SCU from the available pool of ICUs. ICUs should always be informed that they will get lower scores for certain properties, which will influence their overall reputation score, so that this knowledge can also serve as an incentive for future behavior. The same penalties can be applied in SLA violations. In these cases, in addition to lowering performance values and satisfaction scores, there can be monetary penalties. As an extension to our cost model from previous work [14], the new cost of the SCU after adapting it because of a violation by ICUs will be: $Cost_{adapt}(scu_i) = Cost_{old}(scu_i) - \sum_{i=1}^{m}\sum_{x=1}^{j} c(s_i, t_x) + \sum_{i=1}^{m}\sum_{x=1}^{j} c_(s_i^{nw}, t_x) - \sum_{i=1}^{m} c(s_i, vSLA_i)$, where $\sum_{i=1}^{m}\sum_{x=1}^{j} c_(s_i^{nw}, t_x)$ is the cost of the newly added ICUs, and $\sum_{i=1}^{m} c(s_i, vSLA_i)$ is the cost for SLA violation by ICUs; t_x denotes a task executed by an ICU s_i.

4 Programming Adaptations Based on Elasticity Principles

4.1 Algorithm: Runtime SLA-changes Invoking SCU Adaptation

In Algorithm 1 we describe an example of adapting an SCU at runtime when a specific SLA Parameter is changed by the customer at runtime, namely cost. The algorithm has a ranked pool of resources at its disposal, where ICUs are ranked by requirements pre-set from the customer. In addition, the algorithm supposes that an SCU is already formed from that ranked list, based on the type of skills needed for separate tasks. At runtime, in the event that a customer changes the budget for the SCU, the algorithm calculates new costs for each skill-type and sends a notification request-message to each ICU to get back an approval or rejection of the changes (lines 1–5 of Algorithm 1). In the case that not all ICUs approve the changes, new ICUs with the cost parameter fitting the request and the skill-type for which the cost was changed, are selected from the ranked list, and individual SLAs are set with them. They are then added to the SCU, and tasks which were previously assigned to ICUs in the SCU that rejected the changes are delegated to the new ICUs. Hence, the SCU is adapted. The Platform Satisfaction score, is lowered for those ICUs that have rejected the payment changes, and those ICUs are excluded from the collective (lines 7–13).

4.2 Prototype Implementation and Experiments

We ran a Java-based experiment to evaluate runtime adaptations of SCUs when requirements for parameters are changed at runtime. The goal of our experiments is to show that processes allowing for elastic adaptation of SCUs along with runtime SLA changes are more efficient, as workers can be added or excluded from execution as needed, and customers can get faster results with lower cost. We show this by showing that SCU performance is higher with each adaptation point (Fig. 2), as well as the time for total-task execution is lower with each elastic adaptation of social collectives (Fig. 3). Our proof-of-concept prototype consists of the following components: (1) *a model of ICUs, SCUs, ICU and SCU metrics, and Tasks*; (2) a component for ICU *ranking*, according to specific metric constraints from the customer; (3) *adaptation algorithms* that we feed to our process engine; (4) *a process execution engine*, with which we run the process described in Algorithm 1. More specifically, we designed and implemented ICUs with a single skill, for better overview of the results, and cost-per-task as pre-set properties. In addition, we designed performance metrics [15]. The following are only some of them: availability, total assigned tasks, delegated tasks, success-rate, productivity, reliability, social trust, performance trust, socio-technical trust. We designed tasks with skill and cost requirements, and SCUs with metrics specific to them. The source code for the implementation can be found at: https://mirusx@bitbucket.org/mirusx/ulyss.git.

Algorithm 1. Elastic Adaptation of SCUs based on SLA changes

Require: icu member of SCU, icu member of ICU Pool

if $scu.currentBudget == (scu.currentBudget - amount)$ **then**
  for all icu in SCU **do**
    calculateNewFees(taskList)
    sendFeeChangeNotification()
    $icuApprovalList \leftarrow getApproval(icu)$
  /* Adapt the SCU if not all ICUs approved the agreement changes */
  for all icu in icuApprovalList **do**
    while icuCurrent.Approval()== false **do**
      $icuNeededSkillsList \leftarrow icuCurrent.getSkill()$
      $icuCurrent.icuMetrics.PSS = getPSS(icuCurent) - \partial$
      $updateAgreementReciprocity(icuCurrent)$
      $icuCurrent.updateMetrics()$
      $SCU \leftarrow removeICU(icuCurrent)$
  /*Add the next matching ICU from the ranked list that approves the SLA*/
  for all icuNeededSkillsList **do**
    for all icu in rankingList **do**
      if $icu.Approve() == true$ **then**
        $icuCurrent = icu$
        $SCU \leftarrow addICU(icuCurrent)$
      break

SCU Adaptation with SLA parameter changes at runtime. We implemented a variation of Algorithm 1 where the cost is the parameter to be changed, but instead of lowering the total budget of the SCU we lowered the fee for tasks of specific skill types. Thus, we simulated time points at which changes for specific skill-types are required as an input. Already assigned tasks for which the fee was lowered were delegated to other ICUs with matching skill and (lower) cost. In addition, tasks were also delegated at points when they were assigned but were not executed up until a pre-set time-threshold. In both cases tasks were delegated to ICUs already within the SCU or new ICUs were invoked form an ICU pool if no matches were found. Let us examine the results, which are shown in Fig. 2. Figure 2 (a) shows the socio-technical trust scores of four SCUs that we selected (as interesting cases to analyze). We selected to show the socio-technical trust score (STT), as it is a metric that encompasses all other important metrics, such as success rate, performance, productivity of ICUs as well as social-trust scores that ICUs assign to each other according to their collaboration satisfaction. We randomized the assigned social-trust scores, but biased in a way that we assigned higher scores to ICUs with higher number of executed tasks and lower scores to those that delegated their tasks. In Fig. 3(a) we can see that the trust scores are generally rising for all SCUs, but some low points exist. The cost change events for SCU1 and SCU2 happen at the last adaptation point, while the cost changes for SCU3 and SCU6 are at the sixth point of adaptation. Let us examine these cases more closely. SCU1 has a high STT after the end of the

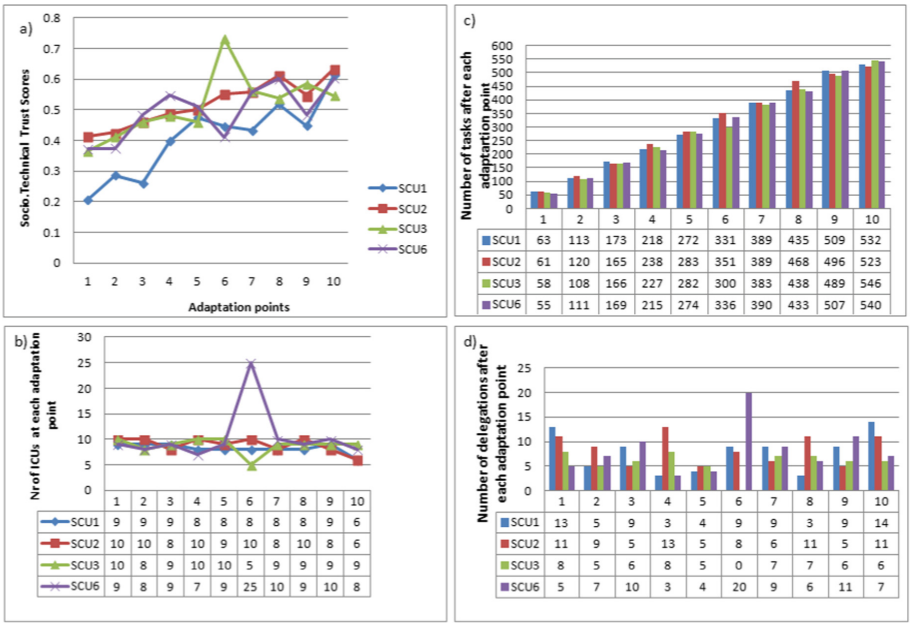

Fig. 2. Elastic SCU adaptation with SLA cost changes

last adaptation, and if we examine sub-figure (b) we will see that at that point it had a low nr of ICUs (6) but high delegations (sub-figure d). Thus, STT is high because the SCU performance is high, as a consequence of the fact that at the last adaptation point some ICUs are excluded form the SCU and more tasks are executed with lower number of people. Examining the case of SCU3 we see that at the sixth adaptation point, it has a spike on STT scores shown in sub-figure (a), while sub-figure (b) clearly shows that the number of ICUs at point 6 is five, and we do not have delegation of tasks at point 6 (shown in sub-figure (d)). In this case the cost changes were made at point 6, and all the new tasks to be assigned with those changes were accepted by ICUs already in the SCU. SCU6 on the other hand, has a low score of STT at point 6 in sub-figure (a). Sub-figure (b) shows that the number of its ICUs is considerably high, 25 ICUs at point 6, and the number of delegated tasks at point 6 is 20 (sub-figure d). We noticed that 18 new ICUs were added to adapt to the cost change requirements as we gave as an input a very low fee for two skill types. In addition, when ICUs are newly included in the SCU their social trust score is lower (we count invocations within an SCU at each adaptation point), which also ads to the lower value of the STT. In summary, this kind of flexibility at runtime is not possible without having a platform design that provides the possibility to change SLAs at run-time, but most importantly to include ICUs in negotiations when SLAs change, so that collective adaptations at run-time are more efficient and agreeable for all parties.

Comparison of elastic and fixed SCU adaptation. We ran another experiment comparing the variation of Algorithm 1 and a Base-Algorithm with which the SCU was adapted as tasks were scheduled and assigned according to skills, and they were delegated when they reached a time-threshold, as well as when a cost change requirement was given as an input at runtime. However, in the base-algorithm tasks are delegated only to ICUs that are already within the SCU, and no new ICUs were added to the SCU. Thus, in the variation of Algorithm 1 the adaptation is elastic, we exclude and add ICUs from the pool of ICUs, while in the base algorithm after the SCU is formed, ICUs are not changed. We compared the two algorithm in terms of time efficiency. Figure 3 shows the results. At each adaptation point a new bag of tasks is assigned to SCUs, sub figure (a) shows the number of tasks with each adaptation point. Sub-figure (b) shows the time units after each adaptation point, these time units are calculated as the time to execute the bag of tasks for (and after) each adaptation point, including the delegated tasks. Sub-figure (b) clearly shows that even if the task number assigned at each adaptation point is similar in both SCUs, the time to execute the tasks is not. In the fixed SCU the changes in cost for a skill resulted in delegating multiple tasks to a single ICU, thus the waiting time in queue and the total execution time of tasks has a higher value. On the other hand, in the elastic adaptation algorithm we scheduled each task for which the cost is changed to a new and different available ICU from outside the SCU and included them in the SCU at the needed adaptation points and excluded them when no longer needed. This clearly lowered the time to execute tasks,as shown in sub-figure (b).

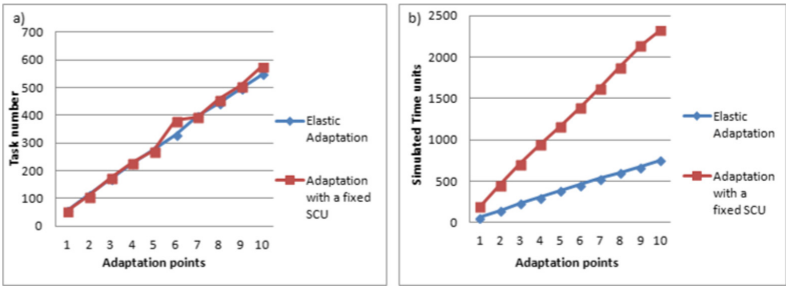

Fig. 3. Task-execution time-comparison in elastic and fixed adaptation algorithms

5 Related Work

The fundamental work that first introduced Social Compute Units is presented in [4], whereas [18] discusses a specific scenario detailing further the way of SCUs utilization and its benefits with a concrete real-life scenario. Proposals for provisioning human skills with the service oriented model are presented in [6].

There is very little work on SLAs in human computation environments. Initial work which concerns crowdsourcing environments in particular is presented

in [8]. The authors give an example of an SLA that may be exchanged between customers and crowd-provider platforms and also present a few crowdsourcing specific SLOs concerning worker skills, quality of executed tasks and customer fees. Another work considering SLAs for human computation, is presented in [13] by Psaier et al. The authors discuss scheduling mechanisms for crowdsourcing environments, which will meet the requirements of contracts, between multiple entities/roles. Schall in [16] discusses protocols for human computation and supports our claim that platforms for provisioning social computation need to work with SLAs, for better task and resource management, just as in software services. A description of requirements and considerations for platforms that support QoS management with SLAs in human computation is introduced in [7]. Elastic management of properties such as cost and workload are presented in [17], where elasticity is used to define restrictions for performance metrics defined in SLA guarantee terms. On the other hand, because human behavior is highly unpredictable, the strict definition of time-related constraints is crucial for SCUs. The temporality aspect for SLAs is investigated in [12]. Müller et al. in [11] have presented a formalization model for creating SLAs with compensations such as rewards and penalties. The compensation functions that they propose can be used in Compensable Guarantees (a term the authors coin for Guarantee Terms with compensation functions). Interesting for our work is that this work concerns Cooperative Information Systems and the models would also be fit to be used for SLAs for human-based services. Authors of [10] also discuss a contract model and a negotiation process considering a social computing scenario where rewards and penalties are not only monetary. Key Performance Indicators mapped to SLOs as metrics for business processes, and specific metrics for services in an orchestration, aggregated in SLAs are discussed by Wetzstein et al. in [20]. Social BPMN is discussed in [2], as a mechanism that integrates social interactions in standard organizational business processes. An example of a process-based programming model for crowdsourcing is presented in [9].

6 Conclusion

We investigated process-based SLAs and human-based collective management, by providing key parameters to be included in SLAs for human computation. In addition, we provided an example mechanism for SCU adaptation based on SLA changes at runtime, and provided a prototype for process-based management of SCUs. Our experiments with a processed-based algorithm showed that social-computing mechanisms designed with elasticity in mind where SLA parameters can be changed at runtime and SCUs can be adapted based on those changes, are more efficient than traditional processed with fixed-resource management. In future work we plan to investigate privacy-enforcing parameters for SLAs, and the way that they can be included in processes for SCU management.

References

1. Andrikopoulos, V., Saez, S.G., Karastoyanova, D., Weiß, A.: Collaborative, dynamic & complex systems - modeling, provision & execution. In: CLOSER 2014 - Proceedings of the 4th International Conference on Cloud Computing and Services Science, Barcelona, Spain, 3–5 April 2014, pp. 276–286 (2014)
2. Brambilla, M., Fraternali, P.: Human computation for organizations: socializing business process management. In: Michelucci, P. (ed.) Handbook of Human Computation, pp. 255–264. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-8806-4_21
3. Bucchiarone, A., Dulay, N., Lavygina, A., Marconi, A., Raik, H., Russo, A.: An approach for collective adaptation in socio-technical systems. In: Proceedings of the 2015 IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW 2015, pp. 43–48. IEEE Computer Society, Washington, DC (2015)
4. Dustdar, S., Bhattacharya, K.: The social compute unit. IEEE Internet Comput. **15**, 64–69 (2011)
5. Dustdar, S., Guo, Y., Satzger, B., Truong, H.L.: Principles of elastic processes. IEEE Internet Comput. **15**(5), 66–71 (2011)
6. Dustdar, S., Truong, H.L.: Virtualizing software and humans for elastic processes in multiple clouds- a service management perspective. IJNGC **3**(2), 109–126 (2012)
7. Kern, R., Zirpins, C., Agarwal, S.: Managing quality of human-based eServices. In: Feuerlicht, G., Lamersdorf, W. (eds.) ICSOC 2008. LNCS, vol. 5472, pp. 304–309. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01247-1_31
8. Khazankin, R., Psaier, H., Schall, D., Dustdar, S.: QoS-based task scheduling in crowdsourcing environments. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 297–311. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25535-9_20
9. Kucherbaev, P., Tranquillini, S., Daniel, F., Casati, F., Marchese, M., Brambilla, M., Fraternali, P.: Business processes for the crowd computer. In: La Rosa, M., Soffer, P. (eds.) BPM 2012. LNBIP, vol. 132, pp. 256–267. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36285-9_31
10. Michalk, W., Haas, C.: Incentives in service level agreement establishment the case of economic and social aspects. In: 2011 First International Workshop on Requirements Engineering for Social Computing, pp. 30–33, August 2011
11. Müller, C., Gutiérrez, A.M., Martín-Díaz, O., Resinas, M., Fernández, P., Ruiz-Cortés, A.: Towards a formal specification of SLAs with compensations. In: Meersman, R., Panetto, H., Dillon, T., Missikoff, M., Liu, L., Pastor, O., Cuzzocrea, A., Sellis, T. (eds.) OTM 2014. LNCS, vol. 8841, pp. 295–312. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45563-0_17
12. Müller, C., Martín-Díaz, O., Ruiz-Cortés, A., Resinas, M., Fernández, P.: Improving temporal-awareness of WS-agreement. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 193–206. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74974-5_16
13. Psaier, H., Skopik, F., Schall, D., Dustdar, S.: Resource and agreement management in dynamic crowdcomputing environments. In: 2011 15th IEEE International, Enterprise Distributed Object Computing Conference (EDOC), pp. 193–202, August 2011

14. Riveni, M., Truong, H.-L., Dustdar, S.: On the elasticity of social compute units. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 364–378. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07881-6_25
15. Riveni, M., Truong, H.L., Dustdar, S.: Trust-aware elastic social compute units. In: 2015 IEEE Trustcom/BigDataSE/ISPA, vol. 1, pp. 135–142. IEEE (2015)
16. Schall, D.: Service oriented protocols for human computation. In: Michelucci, P. (ed.) Handbook of Human Computation, pp. 551–559. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-8806-4_42
17. Schulz, F.: Elasticity in service level agreements. In: 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 4092–4097. IEEE (2013)
18. Sengupta, B., Jain, A., Bhattacharya, K., Truong, H.-L., Dustdar, S.: Who do you call? Problem resolution through social compute units. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) ICSOC 2012. LNCS, vol. 7636, pp. 48–62. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34321-6_4
19. Truong, H.-L., Dustdar, S.: Context-aware programming for hybrid and diversity-aware collective adaptive systems. In: Fournier, F., Mendling, J. (eds.) BPM 2014. LNBIP, vol. 202, pp. 145–157. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15895-2_13
20. Wetzstein, B., Karastoyanova, D., Leymann, F.: Towards management of slaaware business processes based on key performance indicators. In: 9thWorkshop on Business Process Modeling, Development and Support (BPMDS 2008) - Business Process Life-Cycle:Design, Deployment, Operation Evaluation (2008)