

Provisioning and Management Techniques for Elastic Collectives in Human Computation

PhD THESIS

submitted in partial fulfillment of the requirements for the degree of

Doctor of Technical Sciences

within the

Vienna PhD School of Informatics

by

Mirela Riveni, MSc.

Registration Number 1028032

to the Faculty of Informatics

at the Vienna University of Technology

Advisor: Univ.-Prof. Dr. Schahram Dustdar

Second advisor: Priv.-Doz. Dr. Hong-Linh Truong

External reviewers:

Prof. Dr. Harald C. Gall. University of Zurich, Switzerland.

Assoc. Prof. Mehmet S. Aktas. Yıldız Technical University, Turkey.

Vienna, 6th June, 2018

Mirela Riveni

Schahram Dustdar

Declaration of Authorship

Mirela Riveni, MSc.
Vienna, Austria

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

Vienna, 6th June, 2018

Mirela Riveni

To my parents, and my sister

Acknowledgements

First, i would like to thank my advisor, Schahram Dustdar, for giving me the opportunity to work under his mentorship. I feel honored to have been a part of a research group under the guidance of a scientist with admirable achievements and character, and who is a role model for many; so much more because i was continuously encouraged and inspired along the way. My advisor has given me guidance, freedom and trust, and has shown flexibility regarding my research topic, which are crucial values in intellectual endeavor. In addition, I thank Hong-Linh Truong, my second advisor, for the fruitful brainstorming sessions, idea exchanges, and collaboration for parts of this work. Moreover, I thank the PhD School of Informatics for the financial support, and it's staff for the admirable organizational and administrative work. The research process, including the environment I have been in, the wonderful people I was surrounded with and hope to be so in the future, have given me invaluable academic experience, but most importantly an extraordinary life experience. I am grateful for every moment.

I am happy to have met remarkable people, with whom we shared similar exciting research experience. Specifically, I would like to thank Mohamed Radwan, Kseniya Cherenkova, Soheil Qanbari, Peter Kán, Danijel Novakovic, Ognjen Scekcic, Muhammad Candra, Vitaliy Lipchinsky, Andreas Scharf and Shaghayegh Sharif Nabavi, for the interesting conversations within our shared time in Vienna, and their friendship.

Thanks to my collaborators from around the world for the knowledge exchange, constructive comments and contributions. In particular, I thank Mehmet S. Aktas from the Yıldız Technical University, in Istanbul, and his student Muhamed J. Baeth, with whom we did joint work while I was on a short research visit there. Thanks to Tien-Dung Nguyen and Christiaan Hillen as well, for valuable discussions and cooperation. I would also like to thank Aksenti Grnarov, my professor from my BSc studies, who follows my successes. In addition, thanks to my close friends for their cheering.

Finally, I am eternally grateful to my parents, Mübedzel and Namik Riveni, and my sister Laura. My parents have continuously been my greatest support, and above all they are my greatest examples and inspiration in life. They have showed me what is understanding, compassion, trust, patience and love, and provided me with opportunities in life along with showing me the way to create them. Thank you for all your love.

Abstract

Technology and society are in an interminable process of mutual effect on each-other's transformation; thus, computational challenges along evolving social dynamics are getting highly complex. New approaches are needed to tackle this complexity, as some computational tasks cannot be solved solely by software. Consequently, we argue that systems need to be evolving toward the integration of *software services*, *things* and *people* that work together, e.g., as Collective Adaptive Systems. This thesis focuses on the *people* part of these systems, and more precisely on automating the provisioning and management of human-based resources, in scenarios where applications and systems include human-computation.

Human computation has been slowly but firmly gaining its momentum, and software is beginning to be designed and built with the possibility of enabling human task-execution to be provisioned as a service. In this work, we investigate the mechanisms to provision and elastically manage collective and collaborative human computation. We present a report of our investigation of strategies for formation and elastic coordination and management of collectives of experts who provide their skills online as services. With particular focus on elasticity we argue that human computation is more efficient, reliable, on-time and cost-effective when expertise can be scaled in and out, at runtime. Moreover, we argue that trust is highly important due to the many uncertainties that come from the human nature. Hence, we investigated trust metrics and ways of using trust in automated coordination of collectives. Furthermore, we see runtime negotiations as an additional mechanism to guarantee quality of results for online human-task execution and argue that they are as crucial for human-based services as they are for software services. Last but not the least, we investigated the benefits of storing provenance data, along with the challenges that human-computation entails regarding privacy.

Contents

Abstract	ix
Contents	xi
List of Figures	1
List of Tables	2
List of Algorithms	5
Acronyms	7
1 Introduction	9
1.1 Social Compute Units Fundamentals and Motivation	10
1.1.1 Social Compute Units: Basics	10
1.2 Problem Statement and Research Questions	10
1.3 Contributions	12
1.3.1 Publications	13
Publication List	15
1.4 Thesis Structure	16
2 State of The Art	19
2.1 Human Computation: Categories and Definitions	19
2.1.1 Social Compute Units	21
2.2 Service Oriented Computing	21
2.2.1 Human Provided-Services	21
2.3 Frameworks, Models and Platforms	22
2.4 Resource Management	23
2.4.1 Metrics	23
2.4.2 Resource Ranking and Selection Algorithms	24
Individual Resource Selection	24
Team/Collective Formation	25
2.4.3 Runtime Resource Provisioning, Management and Adaptation Tech- niques	25

2.4.4	Elasticity	26
2.5	Incentive Mechanisms and Pricing	26
2.6	Trust and Reputation	27
2.7	Quality of Service and Service Level Agreements	30
3	Elastic Social Compute Units: Provisioning and Management	31
3.1	SCU Preliminaries	31
3.1.1	ICU and SCU Definitions	31
3.2	Motivation Scenario	33
3.3	On the Elasticity of Social Compute Units	34
3.3.1	Definition and Principles of Elasticity	35
3.3.2	ICU/SCU Formal Notation	35
3.3.3	Metrics: Notation and Definitions	36
3.3.4	SCU Execution Model	36
	SCU States	36
	SCU in Execution	37
	Elasticity APIs	39
3.3.5	Elastic SCU Provisioning Platform	39
3.4	SCU Runtime Management: Elastic Adaptation Mechanisms	40
3.4.1	Programming an Elasticity Strategy: A semi-automatic adaptation strategy with human-in-the-loop decision making	41
3.4.2	Experiment: Executing an Elasticity Strategy	44
	Utilizing an Analytic Hierarchy Process (AHP) model for ranking of ICUs	44
	Implementation of Algorithmn 3.1 and results	45
3.5	Related Work	46
4	Trust in Social Computing: Metrics, Model and Algorithms	49
4.1	Background and Motivation	49
4.1.1	Trust for Social collectives	49
4.1.2	Motivation Scenario and Challenges	50
	Observations and Challenges	51
4.2	A Socio-Technical Trust Model for Social Compute Units	53
4.2.1	Modeling Trusted Individual Compute Units	53
4.2.2	Metrics: Notation and Definitions	53
4.2.3	Socio-Technical Trust (STT) Model	54
	Context	56
4.2.4	Modeling Trusted Social Compute Units	56
4.3	Elastic Adaptation Strategies with Trust: Algorithms and Experiments	59
4.3.1	Experiments	59
4.4	Incentive Mechanisms with Trust	64
4.5	Related Work	65
4.5.1	Trust in Social Computing	65
4.5.2	Agent-based Trust	67

5	Team Formation	69
5.1	Team Formation based on trust and multiple interaction types	70
5.1.1	Problem Statement	70
5.1.2	Model	70
5.1.3	Expert role connected to different types of interaction links-Discussion	74
5.2	Programming team formation	74
5.3	Experiments	76
5.3.1	Evaluation with synthetic data	76
5.3.2	Evaluation with real data	78
5.4	Related Work	80
6	The Application of Service Level Agreements for Social Collectives	83
6.1	Motivation Scenario	84
6.1.1	Language translation	84
6.1.2	On the need for SLAs supporting human computation	84
6.2	Computational-Environment Setting	85
6.3	Modeling SLAs for SCUs	87
6.3.1	Human-centric properties and metrics	87
6.3.2	Penalties	87
6.3.3	Enforcing Privacy with SLAs	89
6.3.4	Examples	89
6.4	SLAs and Elasticity	91
6.4.1	Programming SLA Parameter Changes at Runtime	91
6.4.2	Implementation of a Proof of Concept prototype and Experiments	92
6.5	RelatedWork	97
7	Provenance in Human Computation	99
7.1	Motivation	100
7.1.1	Provenance data in social-computing management-mechanisms . .	100
	Individual Task-assignment and Formation of collectives/Social	
	Compute Units	100
	Adaptation mechanisms for Social Compute Units	100
	Misbehavior prevention and False negatives in Misbehavior detection	100
	Incentive mechanisms	101
	Compensations	101
7.1.2	Challenges	101
7.2	SCU Environment and Provenance	102
7.2.1	SCU Environment	102
7.2.2	Modeling Provenance for SCUs	103
7.3	Experiments	104
7.3.1	Setup	104
7.3.2	Dataset	106
7.3.3	Experiment types and Results	107
	Provenance Visualization	107

	Komadu experiments	111
7.4	Provenance-based Inferred Metrics for Social Computing	112
7.5	Privacy Implications: A Discussion	115
7.6	Related Work	117
8	Privacy in Human Computation	119
8.1	Personal Data on Human Computation Systems	119
8.1.1	Collected data	119
8.1.2	Reasons for collecting personal data	120
	Task-Assignment and Formation of collectives	121
	Management Mechanisms	121
	Quality of Service	121
	Misbehavior prevention	121
	Incentive Mechanisms	122
	Payments	122
8.2	Privacy Risks	122
	User Privacy Policy Awareness	122
	Lack of Transparency in Privacy Policies	122
	Profiling	123
	Lack of Control	124
	Lack of Ownership	124
	Lack of Security	124
8.3	Study	124
8.3.1	Method: Survey Design and Distribution	124
8.3.2	Results and Analysis	127
	Demographics	127
	Privacy Awareness	127
8.4	Suggestions	130
8.4.1	Recommendations	130
8.4.2	Research directions	133
	Transparency with rules or SLAs	133
	Privacy preserving workflows	133
	Payment methods	133
	Location	133
	Evaluation methods	134
	Raising people awareness about privacy	134
8.5	Related Work	134
9	Conclusions	137
	Bibliography	139
	Appendices	154

Appendix A	155
Appendix B	157
Appendix C	159

List of Figures

3.1	SCU lifecycle states	32
3.2	SCU basic-working concept	32
3.3	An illustrative example of an SCU in execution: expanding and reducing states	38
3.4	Conceptual platform model supporting elastic SCUs	40
3.5	SCU Productivity and Effort in relation to Task and ICU number	45
4.1	SCU Working Environment in a Predictive Maintenance Scenario	51
4.2	Platform/SCU Management with Trust	52
4.3	SCU Socio-Technical Trust Model	58
4.4	SCU trust metrics and relations updates	59
4.5	SCU metric updates without considering trust for delegations	61
4.6	SCU metric updates considering the STT trust model	62
4.7	Illustrative xml file with ICU metrics	63
4.8	SCU operation and ICU trust updates	64
5.1	The lighter lines represent interactions only in terms of communication, the darker lines represent the presence of two types of interactions: communication and coordination. Edge values represent interaction weights, a single value represents communication weight, while sets of two values represent weights of communication and coordination, respectively. Values in red represent STT scores of experts.	73
6.1	SLA-based ICU-SCU Provisioning and Management Platform	86
6.2	Elastic SCU adaptation with SLA cost changes	92
6.3	Proof-of-concept prototype	94
6.4	Elastic SCU adaptation with SLA cost changes	95
6.5	Time Comparison of elastic and fixed SCU adaptation algorithms	95
6.6	Code snippet from our experiments for delegating a task with a new cost to a new ICU	96
7.1	SCU environment model.	102
7.2	A graphical provenance-model based on the PROV-O specification, focusing on ICU task executions and profile updates.	105
7.3	A specific example focused on ICU task assignments and profile updates.	105

7.4	Code snippet from defining Entities, Activities and Relationships.	106
7.5	Tasks for ICU 16 and 22 at four selected checkpoints during one SCU execution.	107
7.6	Provenance details after a run of an SCU adaptation-algorithm	108
7.7	ICUs and tasks during the execution of one bag-of-tasks. The graph result of checkpoint 7.	108
7.8	Provenance graphs for an SCU after execution of bag-of-tasks at two different checkpoints during run-time	109
7.9	Provenance graphs for an SCU after execution of bag-of-tasks at two different checkpoints during run-time	109
7.10	A sample xml file generated with ProvToolbox.	110
7.11	PROV Agents in our experiments inserted to Komadu	111
8.1	User reports on regulations	131

List of Tables

2.1	Overview of metrics for people in social computing (individuals and collectives)	24
3.1	Notation and description of basic ICU metrics and parameters	35
3.2	SCU metrics to be used in adaptation mechanisms	37
3.3	Fundamental state alternatives of the SCU Execution phase	38
3.4	Example API, abstract methods for ICU manipulation	39
4.1	Notations	57
5.1	Notations	72
5.2	Ranked teams and values of three objectives	77
5.3	Ranked teams and values of three objectives as input for Algorithm 5.2	78
5.4	Ranked teams and values of three objectives at a final run of team generations in Algorithm 5.2	79
5.5	Ranked teams and values of three objectives from a real-world data-set	80
6.1	Notation and description of basic parameters for SCU SLAs	88
7.1	Komadu injection processing time of specific log-data	113
7.2	Komadu injection data for 50 checkpoints, and a total of 2664 activities treated as events	113

7.3	Komadu injection data for 400 checkpoints, and a total of 21780 activities treated as events	113
8.1	Demographics of participants	126
8.2	Most common collected data	126
8.3	Data concerns	128
8.4	Security related survey statements	129
8.5	Opinions on regulations, and approaches in research and industry	131

List of Algorithms

3.1	SCU Adaptation: Task-delegations with ICU-side assurance	43
4.1	A Cost-Effective Algorithm for Elastic Adaptation of SCUs based on ICU Reputation	60
4.2	Membership-Collaboration Trust Update Algorithm as an Incentive Mechanism	66
5.1	Team-formation algorithm utilizing AHP for ranking experts and non-evolutionary Pareto based team selection	75
5.2	Team-formation Genetic Algorithm	76
6.1	Elastic Adaptation of SCUs based on SLA changes	93

Acronyms

ABC Attribute Based Credentials. 120

AHP Analytic Hierarchy Process. 40

BPEL4People Web Service Business Process Execution Language Extension for People.
20

BPMN Business Process Model and Notation. 82

CAS Collective Adaptive Systems. 10, 73

GWAPs Games with a Purpose. 9

HPS Human Provided Services. 20

ICU Individual Compute Unit. 10

NFP Non-functional properties. 20, 45

QOS Quality of Service. 20

SCU Social Compute Unit. 9

SLA Service Level Agreement. 73

SOA Service Oriented Architectures. 20

STT Socio-Technical Trust. 46

WSLA Web Service Level Agreement. 79

Introduction

The fact that a large number of people are present and act online has opened up extensive possibilities of utilizing peoples' capabilities in various areas. Today, power is in the hands of those who know how to utilize the opportunities conferred by the effective use of these skills provided online, regardless if it is the one who offers skills, or the party that requests those skills. Namely, both parties may end up with gains in different contexts, such as social, political and monetary. To avoid the misuse of those gains, it is imperative that human computation is exercised in an ethical manner by both developers and people who consume its benefits. This thesis should be read with a consideration of the fact that everything proposed was investigated, and should be applied with regard for ethical principles, such as transparency and privacy-respecting approaches in application and system design.

Human Computation is a wide concept that encapsulates the online activity of people who use their skills to bring value through content or task execution. It has already been put into practice in various forms. Microblogging, Wiki's, social and expert networks for example enable content sharing and knowledge creation. On the other hand, we see systems that are already enabling online task execution. Games with a Purpose (GWAPs) include human-in-the-loop and engage people in executing tasks that cannot yet be solved by computers. Crowdsourcing enables people to individually execute small and simple online tasks so that vast amount of information can be processed in a short time. In this context, the term *social computing* is also widely used to cover the online activities of people who are socially connected while executing these activities. We use the terms social computing and human computation interchangeably in this thesis, and we elaborate the reason in Chapter 2. In this thesis we focus on a social computing construct named *Social Compute Unit (SCU)*, which is a concept defined by *people who are engaged in complex task execution, while being connected in a certain social context within which they work together in a collective, for a certain common goal*. We give concrete real-life scenarios for SCUs in Chapter 3, 4, and 5.

From the systems perspective, we see the rise of *Collective Adaptive Systems (CAS)* that encapsulate heterogeneous types of resources and services enabling new ways of resource utilization. To specify, these systems are aimed at integrating various types of resources, such as software, Internet of Things and people working together in a collective and being managed in an automated way. Thus, today's systems should be and are already moving toward being socio-technical in character, as are most of our societies. We envision Social Compute Units as part of these systems, providing services offered by a collective of people as compute resources, and in this thesis we investigate mechanisms of their effective provisioning and management, in this way contributing to the effort of building complex socially-enhanced applications and systems.

1.1 Social Compute Units Fundamentals and Motivation

1.1.1 Social Compute Units: Basics

The Social Compute Unit concept is first introduced in [DB11]. An SCU is a construct representing a collective of people who provide their skills as computing resources or services, for execution of human-computation tasks. In our work, a member of an SCU is called an *Individual Compute Unit (ICU)*. In other words, an ICU is an individual who provides his/her capabilities online to execute human-computation tasks. We will use this terminology throughout this thesis. However, we also use the terms resources, services as well as workers, to refer to ICUs where we find it more fitting for the context of the discussion. Social Compute Units are possible today because of the resource pools that are provided by human computation platforms, including crowdsourcing platforms, social networking platforms, expert networks, enterprise networks and similar. In addition, we can see advancements in modeling online human-based task execution under the service oriented model, as presented in [STD08].

SCUs are formed ad-hoc or on customer request, and they have a certain goal. Thus, they have their own life-cycle. We work with them as having a cloud-like behavior, in the sense that during their life-cycle new members can be included and existing members excluded as needed to adapt the SCU. With these collectives, our goal is to manage human-based task execution in a programmatic way and contribute in the effort to make software and human computation work together as discussed in [TDB12]. In particular, in this thesis, we focus on provisioning and proactive management of team-based/collective human computation that we investigate under the term Social Compute Units.

1.2 Problem Statement and Research Questions

Due to the unpredictable nature of human behavior, SCUs bring significant challenges if we want to manage them in a programmatic way. First, SCUs are collectives and thus their provisioning is very different and more complex than that of crowdsourced workers, as workers in crowdsourcing are managed from the perspective of an individual. With SCUs we have complex provisioning of collective work where tasks may be interdependent

in addition to the fact that SCUs are envisioned for more complex tasks than those in crowdsourcing. Second, the definition of SCUs entails that SCUs are managed elastically, as much as this makes them efficient however, their management is more complex than that of online teams that include a fixed number of workers for example. Consequently, as an SCU adapts at runtime, monitoring metrics and ranking algorithms are needed to match a particular task to an appropriate ICU that would be included in the SCU at runtime, or to exclude an ICU from an SCU when it is not needed anymore. Third, along ICU management, task management is crucial. Existing task management solutions for businesses mostly provide task tracking/monitoring and some are even designed with a certain domain in mind, what lacks is automated task management, not only tracking, which at the same time would be generic so it can be used across domains. In relation to all of the aforementioned challenges is the challenge of trusted ICUs and consequently trusted SCUs.

Trust plays an important role in human computation and is used for activities such as raking and decision-making. Selecting not only the appropriate members for tasks based on predefined customer requirements but also the best among the available ones is of paramount importance in forming an SCU and in managing it at runtime. Furthermore, while the utilization of software services is managed by Service Level Agreements (SLAs), human resources/services are not, and we posit that human computation management with automated SLAs would result in more efficient online work and more transparency than what exists in current industry-based platforms. Hence, mechanisms for processes that include automated negotiation-management, where contracts can be adapted at runtime on customer or worker request are a challenge to be addressed. This would bring even more flexibility to online collectives. Last but not the least, when we have to do with people as resources online, it is of utmost importance to have mechanisms that will protect their privacy, as well as ethical approaches in their management such as in incentive and compensation mechanisms. Privacy is a challenge that is mostly overlooked by researchers in human-based computation.

Let us now relate the aforementioned challenges to the systems perspective. Traditional platforms that support virtual fixed-sized collaborations might not be as efficient as those that support SCUs with elastic capabilities that offer opportunities for variable resource numbers with scalable capabilities. There are several reasons for this. First, unexpected tasks might be generated at run-time which may require resources with new type of capabilities that the current collaboration lacks. In fixed-resource collaborations, usually existing members need to learn these tasks and thus the work might be delayed and/or executed with a quality lower than expected. Next, there might be a human-compute unit that is temporarily misbehaving or its performance is degraded. Its exclusion would bring degradation in the performance of the collective, if another appropriate resource is not employed/engaged as a replacement. Furthermore, due to badly planned delegations, it is often the case that some resources are overloaded while others are underutilized. The latter comes as a consequence of the problem of the reliance on human resource *availability* as one of the fundamental ones in social computing. To

sum up, in already existing platforms we have *vertical elasticity*, which means that the optimization and adaptation is executed within the collective and often with a centralized decision-making, while current needs and developments require dynamic and elastic applications where resources/services are utilized on demand. The latter case tells us that in socially enhanced applications and platform we need *horizontal elasticity*, similar to cloud computing. Finally, we focus on the following fundamental research questions in this work:

- *What are the mechanisms that a human computation system needs to deploy so as to provide and manage SCUs with elastic capabilities, in terms of included resources and their dynamic properties, and in terms of task management?*
- *What are some effective ways of collective formation?*
- *Can we come up with a trust model that can be included in mechanisms for elastic management of SCUs, and what would be some conceivable trust-based mechanisms utilizing this model?*
- *How should service contracts for SCUs be defined and how can they be utilized?*
- *How can provenance be utilized in human computation systems?*
- *Which are the privacy implications in human computation systems that need to be addressed?*

1.3 Contributions

The main contribution of this thesis is to provide a set of mechanisms to be used for the full support of the whole lifecycle of SCUs, from SCU provisioning and their effective elastic runtime management to their dissolution. That is why we focus on multiple areas of investigation, such as: metrics in social computation, elasticity, trust, SLAs for SCUs, and provenance. Thus, our goal in this dissertation is to investigate and present a holistic support for the whole lifecycle of SCUs. In order to achieve this:

- We define and present a set of metrics for measuring ICU performance as well as social metrics, both of which we use in ICU and SCU ranking, selection and adaptation algorithms. In relation, we also derive and define SCU related metrics.
- We define elasticity in social computing and describe its principles and properties and ways that these properties can be used to provide elastic SCUs.
- We construct and introduce a novel trust model for ICUs and SCUs that we call a Socio-Technical Trust Model or STT.

- We define and present a set of algorithms for formation of collectives and their elastic adaptation at runtime (some of which are based on our presented trust model). In addition, we present process-based mechanisms with which negotiations can be conducted at runtime and social collectives adapted accordingly.
- Last but not the least, we investigate the use of provenance data in social computing and demonstrate its benefits together with a discussion on privacy implications.
- We discuss privacy implications in social computing systems as well as present results from a study we conducted by interviewing crowdsourcing workers regarding privacy related issues and their concerns.

Our contributions are of benefit to researchers in social computation and in particular for areas such as ranking, recommendation systems for human computation, adaptation mechanisms and incentives in social computing. Moreover, the challenges we address are all related to providing mechanisms that contribute to building platforms that provision and manage human computation.

1.3.1 Publications

We published our research as conference papers, workshop papers and book chapters. Parts of our publications are included in this dissertation in verbatim. In particular, we presented the following contributions:

- Elastic Management of Social Compute Units - The principles of the concept of Social Compute Units were elaborated in [RTD14], published at the 26th International Conference on Advanced Information Systems Engineering(CAISE 2014). In particular, we introduced novel performance metrics for experts providing their capabilities as services as part of an elasticity model for Social Compute Units. We investigated the SCU lifecycle and the different states that it can be at runtime in different situations. Thus, we provided an execution model for Social Compute Units. Furthermore, we introduced an algorithm for a cost-effective elastic adaptation of Social Compute Units using reliable delegations. These contributions are presented in Chapter 3 of the thesis.
- Trust - When people are one of the core resources of a software application, control and management of task execution need to be tuned with the innate uncertainties and unpredictability that comes with the nature of people. Thus, we identified trust as being one of the crucial metrics when managing social computing applications. However, the bulk of existing work is focused solely on the social aspect of a trust metric in areas such as crowdsourcing and web-based collaborations or social/expert networks. We argue that a trust metric for human based computation need to consider metrics that reflect the performance level of people, and that can be measured in an automated way. Consequently, we

worked on an integrated trust model for human based resources, that we call Socio-Technical Trust and we presented our work at the 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-15), [RTD15]. We present our proposed trust model, adaptation strategies and incentive mechanisms in which we apply our trust model in Chapter 4.

- Team Formation - Collective formation and selection presents a multi-objective challenge. The type of skills, performance and interaction dynamics between people are among the indicators for effective teams. While existing work has approached interactions as indicators in combination with other metrics, such as coordination cost, and cost in relation to available team budget, we focus on interaction types in combination with a trust metric. Our approach to the team formation and selection problem is elaborated in Chapter 5. The paper approaching this problem was accepted and presented at the 17th IEEE International Conference on Cognitive Informatics & Cognitive Computing, (won a Best Paper Award), and is due to be published.
- Service Level Agreements - While other resources in CAS are managed by SLAs, human-based services are not managed by machine-readable contracts. Considering the fact that not much work has been reported on SLAs in settings where human computation is an integral part of a process, we investigated SLAs for social computing. We provide an example mechanism for an adaptation of a collective based on parameter changes at runtime, and describe a proof-of-concept prototype for process-based management of SCUs. We argue that social-computing mechanisms designed with elasticity in mind, with which SLA parameters can be changed at runtime in negotiations with people who provide their skills as resources, and with which SCUs can be adapted based on those changes, are more efficient than traditional processed with fixed-resource management. Chapter 6 reports our work on process-based SLAs. We presented this work at the Business Process Management Workshops, 2017, [RND17].
- Provenance - Mechanisms for assessing peoples' behavior, performance and competence depends on historical data. Hence, we look into the possibilities that provenance data offers, which would benefit the aforementioned mechanisms. We present a case study, in which we map social computing terms to provenance terms, and evaluate the use of provenance data through storing this type of data generated from a collective adaptation algorithm. This part of our work is described in Chapter 7. We have presented our work on provenance with a paper at the 13th International Conference on Semantics, Knowledge and Grids (SKG), Beijing, China, [RBAD17], and in [RNAD].
- Other work related to this thesis - We discussed what are the requirements for a simulation system of socially enhanced applications in [RTD12] presented at the 1st International Workshop On Socially Intelligent Computing (SINCOM2012), part

of the OnTheMove OTM Federated Conferences and Workshops 2012 (OTM'12). We also worked on a survey of the state of the art on social interactions in online teams, focusing on research regarding types of online collaborations, monitoring and analysis through different categories of metrics, task types and their online-execution management. This survey was published as part of the Encyclopedia of Social Network Analysis and Mining, in [SRTD14] and in a second edition in [SRTD17]. We use material from the latter mentioned work for our discussion of related work in Chapter 2. Privacy related topics and a survey on privacy awareness is provided in Chapter 8. The paper discussing privacy related issues was accepted and is due to be published in Emerging Research Challenges and Opportunities in Computational Social Network Analysis and Mining, Springer.

Publication List

- M.Riveni, T-D Nguyen, M.S, Aktas, S. Dustdar. Application of provenance in social computing: A case study. *Concurrency Computat Pract Exper.*2018;e4894. <https://doi.org/10.1002/cpe.4894>
- M. Riveni, M. J. Baeth, M. S. Aktas and S. Dustdar. Provenance in Social Computing: A Case Study. 13th International Conference on Semantics, Knowledge and Grids (SKG), Beijing, China, 2017, pp. 77-84.
- M. Riveni, T.-D. Nguyen, S. Dustdar. SLA-Based Management of Human-Based Services in Business Processes for Socio-Technical Systems. *Business Process Management Workshops*, 2017: 361-373
- O. Scekcic, M. Riveni, H. L. Truong, and S. Dustdar. Social interaction analysis for team collaboration. In *Encyclopedia of Social Network Analysis and Mining*, pages 1–16. Springer New York, New York, NY, 2017.
- M. Riveni, H.-L. Truong, and S. Dustdar. Trust-aware elastic social compute units. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 1, pages 135–142. IEEE, 2015
- M. Riveni, H.-L. Truong, and S. Dustdar. On the elasticity of social compute units. In M. Jarke, J. Mylopoulos, C. Quix, C. Rolland, Y. Manolopoulos, H. Mouratidis, and J. Horkoff, editors, *Advanced Information Systems Engineering*, volume 8484 of *Lecture Notes in Computer Science*, pages 364–378. Springer International Publishing, 2014
- O. Scekcic, M. Riveni, H. L. Truong, and S. Dustdar. Social interaction analysis for team collaboration. In *Encyclopedia of Social Network Analysis and Mining*, pages 1807–1819. Springer New York, New York, NY, 2014.
- M. Riveni, H. L. Truong, and S. Dustdar. A simulation framework for socially enhanced applications. In P. Herrero, H. Panetto, R. Meersman, and T. S. Dillon, editors, *OTM Workshops*, volume 7567 of *Lecture Notes in Computer Science*, pages 544–553. Springer, 2012

1.4 Thesis Structure

Chapter 2 covers an extensive survey of related work. Because of the reason that we intend to provide mechanisms for the maintenance of the work of an SCU during its whole lifecycle we cover multiple areas in our work, from the investigation of human behavior from the social perspective, such as finding out metrics that would at least to some extent help us to quantify some characteristics such as capabilities and performance levels, ranking algorithms in web science that we can use in ranking algorithms for ICUs, research on trust as well as recommender systems from multiple areas to help us come up with a trust and reputation model for ICUs and SCUs, service selection and composition mechanisms in service oriented architectures to help us with SCU formation concepts, to the elasticity concept in cloud computing to help us with defining elasticity properties for SCUs. Thus, the state of the art work from these multiple areas and the relation to the work presented in this dissertation is discussed in Chapter 2.

Defining new metrics to quantify and measure a human activity or the quality of an artifact coming from that activity is extremely difficult, especially when they need to be generalized. However, with the clearly defined ICUs and SCUs, as well as with the possible domains of online task execution in mind we have come up and defined metrics for different ICU/SCU characteristics. We use these metrics for ICU ranking based on different criteria, for task assignment, as well as for SCU adaptation mechanisms. As we aim for elasticity in SCU adaptations we define elasticity for SCUs and have presented algorithms for their elastic adaptation. Hence, metrics, an elasticity model for SCUs and a set of elastic adaptation algorithms are presented in Chapter 3.

Based on the understanding of the SCU and the way it can be provisioned and managed, we continued investigating ways for making our adaptation algorithms more efficient. As trust is an important metric in today's human computation platforms, we worked on a feasible trust model for ICUs and SCUs. Seeing that most of the existing work on trust is based on social trust, i.e., how trusted is a person by others with whom he/she has interacted, we realized the lack of a model that considers both the social aspect of trust as well as automatically measured performance metrics in human computation, and investigated a possible one. We present our Socio-Technical Trust Model in Chapter 4. Furthermore, incentive mechanisms in human computation are utterly important to motivate online work. Many human computation platforms have some kind of incentive mechanisms, such as monetary or reputation incentives. We utilized our trust model to provide an incentive mechanism for ICUs as well. Chapter 4 also includes a set of adaptation algorithms as well as incentive mechanisms, based on our trust model.

Chapter 5 reports on our investigation on team formation and selection strategies.

In Chapter 6 we discuss process-based SLA-adaptations at runtime and SLA-based management of SCUs, where SLA parameters can be changed at runtime both by customers and workers.

Chapter 7 discusses the utilization of provenance data for human computation.

We came up with a list of privacy implications in human computation systems by investigating the data that is collected from current existing business platforms and reading their privacy policies. Most importantly, we conducted a survey and interviewed people that use crowdsourcing platforms as well as some domain-based platforms for more complex tasks such as translation and software development, asking them about privacy concerns. We got results that indicate a certain level of awareness for privacy issues in these platforms and got some insight to problems that people face. The results of this investigation are presented in Chapter 8 along with some recommendations for developers of socially-enhanced applications.

We conclude the thesis in Chapter 9 along with some open research questions.

State of The Art

In this work we took a holistic approach to investigate social computing from the aspect of collaborative and highly coordinated online task execution which we model in the form of a construct that we call Social Compute Units. With Social Compute Units we make a clear distinction from individual, non-coordinated and non-collaborative computation (including those such as individual task execution as in crowdsourcing, as well as email, blogs and social networks). Because we take a holistic approach, for our purposes we needed to investigate works related to multiple research areas and did so in the following ones: performance and behavioral metrics, (human resource) management algorithms, scheduling and adaptation algorithms, trust and reputation models, incentive and payment models in social computing, service level agreements, and privacy. Thus, in this chapter we provide our extensive review that we conducted to find works that tackle our aforementioned interest areas and discuss how these works are related to our contributions. In the next section we look into fundamental work that define different types of computation where humans are involved in task execution so as to give an overview and to clarify the place of Social Compute Units, the construct that we work with, among the plethora of terms, definitions, models and systems that include human task-execution.

2.1 Human Computation: Categories and Definitions

As a term, human computation, is considered to be first used by Luis von Ahn in his doctoral dissertation in [VA05]. In it he presents CAPTCHA, a mechanism that utilizes human capabilities in order to differentiate humans from machines, to be used in various applications, e.g., for object recognition in images or reading hard-to read text from scanned books. In addition, he discusses games with a purpose and their use. In [QB11], Quinn and Bederson have provided a classification of human computation systems with a review of the state of the art. In their classification human computation

differs from social computing in the sense that human computation is managed by a software system, whereas social computing implies human behavior that is only mediated by a software system in terms of communication and includes no task-execution but more content-sharing. On the other hand, collective intelligence is a group of people working together to conduct intelligent work (and this includes online as well as offline work). The classification of Quinn and Bederson is cited by many researchers and is an acceptable one. However, there is no formal convention for the use of the terms that include human task-execution and we slightly differ in the definition of the term *human computation*. When referring to *human computation*, we consider several types of task-execution conducted by humans, be it crowdsourcing (see a survey in [YKL11], and an example of crowdsourcing process-approaches under the term human computation in [LCGM10]), or social computation as automated collaborative execution of complex tasks in complex socio-technical systems, and we support this approach (varying a little from the taxonomy presented in [QB11] by Quinn et al.). Some researchers also support this view, e.g., the authors that introduce the term *distributed human computation* in [GRS05], whereas Law in [Law11] intersect human computation with social computing similar to Quinn and Bederson.

To generalize human computation, in a form that fits the topics investigated in this thesis, we define it as *the utilization of human intelligence for tasks, activities and problems that cannot yet be executed and solved by artificial intelligence, where those are tasks, activities and problems in a form that could be solved only by humans who utilize software tools, or by a combination of software and humans in a semi-automated collaboration and coordination*. In addition, we define the term social computing as task execution or problem solving by people connected in a specific social context, and do not imply content-sharing in social networks. Thus, in this thesis we sometimes use the terms human computation and social computing interchangeably.

In the following we give our own definitions (with some adaptations from those in the state of the art) of some of the types of computing concepts with human-in-the-loop, with the aim to further clarify where the concept of SCUs fit within the wide area of human computation.

- Crowdsourcing - is the engagement of a large number of online (anonymous) people in the execution of tasks that are simple for humans but are unsolvable or hard to be solved by computers.
- Social Computing - is the engagement of a group of people in task-execution and content-sharing, for a common goal, where people are connected in a particular social context defined by a common (inter)organizational project, an ad-hoc volunteer project, or a common topic of interest. Hence, the social context of collaboration is one of the important factors that brings people together in social computing.
- Computer Supported Collaborative Work (CSCW) - is the engagement of people in different forms of online team-based work: a) static work, where work is well-defined

by processes, typically within a single organization, b) ad-hoc collaborations where tasks are more complex and team actors cross organizational boundaries, and c) open collaborations where people get together to work on a common goal based on their personal interests (e.g., open source development). (See our encyclopedia article in [SRTD17] for a similar definition).

- Hybrid/Mixed Resource Computations (Mixed Systems) - include human and computer resources working together for a common task. Shahaf and Horvitz for example investigate these types of systems in [SH10]. They present a prototype for language translation, where translation can be conducted only by humans (experts, or crowdsourcing workers), or by hybrid resources where text is first machine-translated and then corrected by human translators. Some other works that include mixed resource computation include [MBO12], [CTD15], [SDC15].

2.1.1 Social Compute Units

In our definitions, Social Compute Units (SCUs) fall within social computing as a subcategory of human computation. However, systems supporting SCUs may fall within social computing systems as well as mixed systems such as collective adaptive systems: in social computing systems because members of an SCU are socially connected within a specific context, be it a specific goal, domain or skills, the same client, or the same enterprise and would use the underlying infrastructure for communication; in Mixed Systems because SCUs are collectives intended for complex problem solving in complex socially enhanced applications and systems that require the use of software services as resources working seamlessly with human resources. The concept of SCU is first presented by Dusdtar et al. in [DB11]. This is the first and fundamental work introducing the SCU, it describes its life-cycle and does not go into details into the SCU execution phase as this was not its aim. SCU execution is tackled in [SJB⁺12], where authors have looked into a specific case of incident management to investigate how SCUs and their evolution (adaptation) perform better over traditional process management. Thus, by discussing a concrete real-life scenario they detail further the ways of SCUs utilization and its benefits. The SCU utilization in dynamic processes is presented in a recent work [FTDC15]. In Chapter 3 we present our investigation of the SCU execution phase and runtime management, and provide mechanisms for elastic runtime management of SCUs.

2.2 Service Oriented Computing

2.2.1 Human Provided-Services

Service oriented computing is a possibility in implementing human computation, and needless to say different than that of how human capabilities are used in computational types such as crowdsourcing, games with a purpose, social and expert networks, and freelance platforms. The WS-BPEL Extension for People Technical Committee within the Organization for the Advancement of Structured Information Standards (OASIS) has

approved the *Web Services Human Task* specification document, version 1.1., [ICK⁺12]. The document specifies human tasks to be executed by people in service oriented applications. More specifically, it provides notations, a language for defining human tasks, defines task behaviors, and provides an API with specific operations to manage tasks. This document is closely related to Web Service Business Process Execution Language Extension for People (BPEL4People), the specification document of which is a guide for defining and specifying human behavior in business processes (see [ICK⁺10]). On the other hand, Schall et al. in [SDB10], and [STD08] discuss the feasibility of provisioning and managing human capabilities as services, which they name Human Provided Services (HPS). In addition to these works, authors in [DT12] investigate and provide concepts and proposals for integrating human capabilities, provisioned as services within Service Oriented Architectures (SOA) so that they can be seamlessly used together with software services in a unified way. The importance of Non-functional properties (NFP), Quality of Service (QOS) models and, trust and reputation models in composing human-provided services in the mentioned works is accentuated equally as for software services. Thus, indeed we can talk about mixed service oriented-systems where human and software services are interacting with each other [SSPD11] and are managed in a similar way.

2.3 Frameworks, Models and Platforms

An earlier work describing research challenges of distributed human computation presented in [GRS05] describes a basic framework with which it would be feasible to provide secure human computation, in the sense that misbehaviors would be avoided, results would be fairly aggregated and payments would be made to the right workers. They include customers with a specific budget, human clients that execute tasks, a broker that manages work and so called storefronts that give some utility to workers needed to complete their tasks (e.g. products, services). A specific domain-based discussion for an architecture of a cloud supported crowdsourcing platform that can support team-based work for software development is discussed in [TWH14]. Specifically, the authors investigate and propose a conceptual architecture where a cloud of people and machines will work together for developing high-quality software products.

Lopez et al. in [VLL10] propose PeopleCloud, a framework intended for enterprise crowdsourcing, where the crowd workers can be from within the enterprise or outsourced. The framework employs an approach of providing the functionality of selecting a number of appropriate crowdworkers according to the parameters that a requester states, including expert discovery and recommendation mechanisms. In connection to our research problem this work is only related with the fact that it allows for collective and collaborative team work, where the platform can dynamically form and suggest teams. The assignment of tasks is done as in traditional crowdsourcing with competitive submissions, accumulative or one submission per task. Our work regarding task assignment differs in that we focus on a push approach when assigning tasks.

Authors of [BCBM12] present *AutoMan* which is a framework for designing human-

computation applications. They present a domain specific programming language, with which programmers can configure the crowdsourcing platform to be used in the backend for invoking workers, they can configure task parameters including the available budget; quality control and scheduling mechanisms are available as well. Collaborative task execution is not mentioned, however, these types of platforms demonstrate the feasibility of utilizing crowdsourcing platforms as pools of resources. Collaborative mechanisms can be built on top of them. Authors of [MB12] present a programming language and framework called CrowdLang for systems that incorporate human computation, and what is of interest to us is that they provide cross-platform integration of resources, in this way making a human cloud possible. Authors in [KPSD11] have presented a platform model for crowdsourcing with monitoring and worker profile management and a skill-based crowd scheduling algorithm. However, their focus is on managing crowd workers and not explicitly SCUs.

From the SOA perspective, Schall et al. in [SDB10], provide a reference architecture for provisioning human provided services(HPS), which includes a registry for service discovery (just as in web services), a services layer that represents HPS that can be discovered, a service bus that provides a messaging infrastructure and a middleware layer which in turn includes a protocol module with rules and patterns as well as a monitoring module. Psaiet et al. have designed and implemented a framework for adaptive service-oriented collaborations and have presented it in [PJS⁺10]. The framework is an integration of the Genesis2 framework, which is a testbed generator for service-oriented environments, and an adaptation framework (named VieCure). The framework presented includes a few services and modules such as, monitoring and logging services, event subscribers to capture e.g., the state of a node, an adaptation module with which are deployed adaptation mechanisms such as changing delegation strategies of nodes based on events. With this work, they make a significant contribution to the available tools and frameworks for designing and developing self-adaptive collaborative applications and systems. Other platform considerations for humans as a service are shortly discussed in [MKGD].

Collective Adaptive Systems are gaining momentum as the possibility of resource diversification in systems are raising with the inclusion of Internet of Things (IoT) and social computation in cloud supported systems [ZSTD15].

2.4 Resource Management

2.4.1 Metrics

In complex adaptive systems that include social computation, metrics are key factors in adaptation mechanisms designed to trigger adaptation events based on constraints, much more because of the unpredictable human nature and behavioral dynamics online. We conducted a survey regarding metrics, mostly used in expert collaborations [SD10](also see our article on social interaction analysis [SRTD17]), but also from (social) networks [New10], crowdsourcing [KK08], [PP11], human-provided services as envisioned within

Service Oriented Architectures [TD09] and multiagent systems [HC08]. Table 2.1, provides our categorization of different metrics regarding people, their interactions and performance within socially enhanced applications and systems. We utilize some of these metrics in our work by providing our own definitions for them, and present novel metrics like *willingness*, for which we also give utilization examples in algorithms.

Profile metrics (individual and SCU level)	Static	Identification parameters
	Dynamic	Skill type/s, skill level
		Homophily(e.g.,topic interests)
		Role, state(e.g.,current availability)
		Reputation
		Price
Structural metrics	Centrality Measures (degree, closeness, betweenness, eigenvector)	
	Structural groups (within the same collaboration context, different SCUs working for a common goal)	
Interaction metrics	Interaction context (common goal)	
	Interaction intensity	
	Interaction availability (in terms of time slots)	
	Responsiveness	
	Social Trust	
Performance metrics (individual and SCU level)	Productivity	
	Effort	
	Reliability (success rate of non-delegated tasks, success rate of delegated tasks, consistency)	
	Willingness, Willingness Confidence-score	
	Performance Trust	
Quality of Results/Data	Completeness (accuracy, freshness, relevancy, consistency)	

Table 2.1: Overview of metrics for people in social computing (individuals and collectives)

2.4.2 Resource Ranking and Selection Algorithms

Individual Resource Selection

A ranking algorithm based on interaction history between people, named DSARank is presented in [SD10]. The algorithm is intended for selecting collaborators in open collaboration environments, and it considers not only the connections and interactions of people who are experts in specific fields with other experts of the same expertise but

also the intensity of those interactions. The HITS method described in [Kle99], which finds the most authoritative nodes in a network is another example, that can be used in expert selection. Ranking algorithms that are used in service selection can also be used in human computation. The Analytic Hierarchy Process (e.g., [CFMT09]) is one such mechanism that provides an opportunity to conduct ranking based on multiple metrics even if they are comprised of multiple atomic ones, so this method is useful when we want to do ranking with complex metrics and we explain this method in more details in Chapter 3. The Logic Scoring Preference methods can also be used in these context (e.g., [YRM08]). In human computation metrics such as interaction intensity, trust and performance based metrics are used in ranking mechanisms. The importance of homophily (e.g., affiliation, gender) as a primary criteria before expertise, is observed by authors in [FZDHC11] for collaboration use cases, which caught are eye as especially interesting.

Team/Collective Formation

Resource discovery in human computation and team formation strategies and algorithms have been the subject of investigation in many works. Anagnostopoulos et.al in [ABC⁺10] present team formation algorithms such as forming teams with minimum size (number of members) that have all required skills, and an algorithm that forms a team with workload optimization, where teams are formed, having minimum size with members that satisfy the required skills and have a pre-specified minimum load. The same authors in [ABC⁺12] present team formation algorithms that while keeping low load also enable low team coordination costs. Authors of [LLT09] present a team formation algorithm that minimizes team communication cost. Dorn and Dustdar in [DD10] have demonstrated the trade-off between expert skills and expert social connections when forming teams of experts, one observation being that high expertise teams may not be the most appropriate ones if team members are socially not well connected within their domain network. The aforementioned algorithms can be utilized for SCU formation, and some also for ICU selection when an SCU needs to be extended via adaptation mechanisms.

2.4.3 Runtime Resource Provisioning, Management and Adaptation Techniques

Task executing collaboration models and runtime collaborations are investigated in works such as [Sag12]. However, the mentioned works focus on fixed teams without elasticity assumptions. Adaptation mechanisms in human computation within (Service Oriented Architectures) SOA, are discussed in [SSPD11]. They focus on adaptations from two perspectives, that of the a) service providers (workers), whose request rate is balanced through automated rejection of requests or delegations, and b) consumers (clients), who are protected by the system by blocking service requests to providers that misbehave or urging them to renegotiate a contract with the same human service provider.

2.4.4 Elasticity

The notion of elasticity is treated in several domains and contexts and has especially gained importance with the advance of cloud computing. In [DT12] authors discuss the reasons, challenges and their approach toward virtualizing humans and software under the same service-based model that will enable elastic computing in terms of scaling both software and human resources. The concept of elasticity in Cloud computing, is being extended to concepts like application [ZKJG11] and process [DGST11] elasticity, e.g., in [DGST11], the authors identify resource, cost and quality elasticity as being crucial in modeling processes in service oriented computing. Mechanisms and a middleware to support scaling services in and out from applications utilizing Software-as-a-Service (SaaS) are presented in [KHCK13].

2.5 Incentive Mechanisms and Pricing

There are two general types of incentives used in social computing: a) intrinsic - in which case people execute tasks based on desire and/or interest, curiosity or willingness to help, e.g., for altruistic reasons and, b) extrinsic - when people execute tasks because they have some gain, e.g., monetary rewards, or reputation. There is much work on incentives, and particularly in the crowdsourcing area. We considered and conducted incentive investigation in crowdsourcing as well, because even though it provides execution of simple-tasks, it still employs human behavior and it is widely investigated both in academia and business. Indeed, it gave us valuable insight. Kaufmann and Schulze in [KS11] present their observation that intrinsic motivation dominate extrinsic incentive types. Authors in [RKK⁺11] have observed that monetary incentives have more effect when tasks are more complex, and the higher the monetary reward the higher the number of completed tasks by workers. However, they also note that when the task description includes a description that implies the task being relevant and important for a social good deed, then workers performed well in terms of accuracy with lower monetary gain.

Authors in [MKC⁺13] compare different payment schemes, but also make observations of non-paid work. They observe that people who are paid complete tasks more quickly than those who work as volunteers. Also, workers that are paid by task spend give more effort than those that are paid on an hourly basis. Authors in [MW09] also observe that while the amount of work increases, the quality/accuracy of work is not increased with higher payments. Hence, research shows that higher intrinsic motivation combined with a (not extremely high) fair monetary gain is a good strategy for accuracy in human tasks. Incentives in enterprise crowdsourcing are investigated in [SHS09]. The authors report that non-monetary extrinsic motivators are important even when the tasks are paid, because they motivate people to become more active and not loose interest in the long term. However, they do state that finding the right motivator is crucial.

Work presented in [STD13] describes an analysis of existing incentive mechanisms, and the part that is of concern to our work is their conclusion and suggestion that for existing business models and more complex social computing such as socially-enhanced

applications (within enterprises) incentives should be dynamic. In other words, systems should be able to adapt incentive and rewarding mechanisms at runtime based on monitoring data, as to lessen (or even avoid) misbehavior by workers. In addition, they describe team-based compensation where the whole team is rewarded for work, the reward being equally split among members or based on member effort. In Chapter 4 we provide an incentive mechanism that tends to motivate SCU workers both on monetary and non-monetary gain. Specifically, we provide an algorithm to pay workers assuming workers are paid by task, and update workers' trust and reputation on a per-task basis. Moreover, we also include a collective trust score that is assigned to all of the workers within the SCU, when all the work is done. And this collective trust score is used to update workers' reputation.

2.6 Trust and Reputation

Trust as a computational concept is first introduced by Marsh in [Mar94]. Trust toward a person, an agent, or any actor in a system, is generally defined and investigated in two contexts:

- **objective trust** - which is based on a person's expertise (area) and competence to execute tasks and can be measured with historical data of task executions, and,
- **subjective trust** - which is based on a subjective impression of the trustor to the trusted person about his/her behavior in the future. This impression can be inferred via different mechanisms, such as based on profile similarities (e.g., common interests) between the trustor and the trustee, based on past interactions, on third party references, and trustee's reputation.

Trust models and mechanisms for trust calculation are investigated in two contexts as well:

- **global trust/reputation**, which are models and mechanisms that compute a global score of trust for people based on their interactions with others and/or their behavior and performance, and
- **local trust**, which are models and mechanisms that compute task from one node (source) to another node (sink) within a network by a specific inferring and/or propagation mechanisms.

In our work we name objective trust as *performance trust* or *technical trust*, and we define it based on metrics related to task-execution which can be calculated in an automated way. On the other hand, we name subjective trust as *social trust*, and we define it based on metrics related to voting mechanisms. We integrate these two types of trust in a trust model for social computing that we present in Chapter 4.

Authors in [SSD10a] investigate trust for human computation from the service-oriented perspective, where human-provided services and software services are utilized seamlessly

in a mixed system. They discuss a trust inference model based on fuzzy theory and different metric categories, as well as composition mechanisms for HPS using trust. In the aforementioned work, trust inference mechanisms are categorized in the following types:

- **direct trust** - is the trust inferred based on direct interactions with another actor in a certain context;
- **trust mapping** - is a mechanism for trust prediction based on a scope(context), ie., if a person is an expert in one field with a certain skill, this trust can be mapped to trust this actor for another skill very closely related to the previous one;
- **recommendation** - is a mechanism that usually implies an aggregation of trust scores of multiple actors about a specific actor, and thus is usually named third-party trust;
- **reputation** - is a global trust inference mechanism, and in many works, as well as in our trust model, reputation is computed as an aggregate score (usually a subjective one), given by all actors that have interacted with the actor for whom this score is computed;
- **trust mirroring** - is trust inferred between actors with similar skills, competencies, and interests; e.g., a person x, will trust a person y for a book recommendation if both have the same interest in books;
- **trust teleportation** - is a trust inference mechanism by which when an actor x trusts another actor y, then it also trusts a third actor z with similar capabilities as y.

Another aspect of trust inference is *trust transitivity*, which is investigated in [FC12] within the multiagent domain. If an actor x, trusts an actor y, and an actor y trusts an actor z, then x may trust z. *Reciprocity* within interactions in a social network as a measure of trust is mentioned in [MMH02]. Example of trust inference algorithms in social networks are works presented in [Gol05], [KG07]. Authors in [BN16] describe their model of trust inference which consists of finding the most trusted path from the source to the sink node, by traversing a path where each edge connects a node with one that has a higher trust rating, and after finding the shortest most trusted path their model calculates trust to the sink node by averaging the sum of the trust ratings of each node on the path weighted by the distance from the source node to each node along the path.

A trust inference algorithm named TidalTrust, is presented by Golbeck in [Gol06], with which trust from a source node to a sink node is inferred by averaging the trust ratings from each node to its neighboring node through a path in a breadth-first-search manner, where a maximum value of trust is set that represents a threshold value that can be utilized as a minimum value for the trust on nodes so that a path from source to sink could be found with a specific trust value. Thus, the model considers the trustworthiness of each neighbor along a path to the sink. It uses a trust accuracy measure by considering the difference between trust ratings from a specific node x and its neighbor to a common neighbor. For example, if x has a neighbor n1, and n2 is a neighbor of both x and n1,

then the difference between the trust rating from x to n_2 , and from n_1 to n_2 is used as an indicator of trust accuracy. If the difference between trust ratings is low then the trust accuracy is high. The network-based experiments in this work have shown that higher trust ratings have a lower difference, and thus higher accuracy for a fixed path length. Hence, higher trust rating along a path provide for higher accuracy in inferring trust from a source node to the sink. To summarize again, in TidalTrust, trust from a source node to a sink node is inferred by averaging the trust ratings from each node to its neighboring node through a path in a breadth-first-search manner, where a maximum value of trust is set that represents a threshold value that can be utilized as a minimum value for the trust on nodes so that a path from source to sink could be found with a specific trust value. Each node is denoted with a maximum trust value on the path from the sink to that node, this maximum value is the minimum of trust rating values from node to node on the way from the sink to the specific node. The maximum trust value is used as a threshold value, and only those paths that bring to the last neighbors of the sink having the trust values at maximum or above are selected to calculate and infer trust from source to the sink. Thus, this algorithm considers the strongest path among the shortest paths from source to sink to infer trust.

MoleTrust presented in [AMT05] is a trust method that as TidalTrust considers path lengths. In particular, the method uses a value called Trust Propagation Horizon to define the distance to which trust is propagated (with the default value being 3), any path going back to the beginning node for example is deleted, and thus the first step of the method is deleting cycles and creating a directed acyclic graph from the network. The second step of the method is computing the trust scores at each hop, for example, if node x is the sink, then the trust scores at distance 1 are considered, then at distance 2 and so on. The authors define a pre-set value of 0.6, which they use to consider only edges coming from nodes with that value and higher. The trust score of the sink node is computed by averaging the trust ratings from incoming edges weighted by the trust ratings of the nodes that give the ratings.

The well-known EigenTrust model, designed for decentralized (rating) systems, is presented in [KSGM03]. It is a model with which a global trust score for peers is assessed by computing the left principal eigenvector matrix of normalized values for local trust values of peers. Thus, trust on a peer is assessed by weighting the opinion of others based on how much the trusting peer trusts them. EigenTrust considers pre-trusted peers (e.g., the network designers), which can be used for trust assessments by peers who enter the network and do not have any friends through which they could infer trust for other peers. The pre-trusted peers are also used as a mechanism to avoid issues with group-based malicious network-members who want to trick the system by rating the group members highly by voting lower for other peers.

The author in [Orm13] presents a Bayesian trust inference method in networks by departing from the more common definition of trust with having the characteristic of transitivity. In contrast, he defines trust as *intransitive*, where a trust relation between x and y , and a trust relation between y and z , does not necessarily mean that x will trust

z. For example, if x trusts y with a sensitive personal information, and y trusts z with his/her sensitive personal information, it does not mean that x will trust z with his/her sensitive information because x and z might have the same friendship-related interest in y, and thus might be in a conflicting relation, rather than a trust relation. On the other hand, the concept of *reliance* is also introduced in the same work, to help with the concept of transitivity. In terms of reliance if x relies on y, and y trusts c, then x will trust c as well, which means that there is a transitivity relation between trust links and reliance links.

Any type of a trust metric depends on the domain and competencies of people in different domains. Thus, trust is context-sensitive, and we treat it as such in this dissertation.

2.7 Quality of Service and Service Level Agreements

There is very little work on SLAs in human computation environments. Initial work which concerns crowdsourcing environments in particular is presented in [KPSD11]. 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. As we mentioned that Social Compute Units should be provisioned elastically so that the performance and time of humans on task execution is utilized optimally for specific fees (or other types of rewards), elasticity plays a role in SLAs as well. Elastic management of properties such as cost and workload are presented in [Sch13b], where elasticity is used to define restrictions for performance metrics defined in SLA guarantee terms. 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 [MMDC⁺07], where the authors present a specific proposal for extension of WS-Ag to support temporality.

Elastic Social Compute Units: Provisioning and Management

3.1 SCU Preliminaries

3.1.1 ICU and SCU Definitions

Definition 1 *Individual Compute Units (ICUs)* represent people who provide their capabilities online, to be utilized as services or activities for executing on-demand human-computation tasks.

Hence, in this dissertation we use the term ICU to refer to human-based resources and/or services, but in some cases we use the terms resources, services, experts and workers where the context of the discussion requires it for clarity.

Definition 2 *Social Compute Units (SCUs)* are elastic collective and collaborative units, performing human-computation tasks with a certain goal, the core resources of which are people with a certain expertise who provide their capabilities as services or activities (ICUs).

SCUs can be created ad-hoc or on request from a customer who defines requirements and sets constraints. Because our work on SCUs is guided by the end goal of providing this construct to be utilized in hybridity-aware collective adaptive systems, which means systems where people and software collaborate together, our assumption is that an SCU can be requested by a customer as well as by a software agent. SCUs have their own lifecycle, with the following states: Request, Create, Assimilate, Virtualize, Deploy and Dissolve [DB11], as Figure 3.1 illustrates. A customer requests a SCU to be formed, the

provisioning platform then creates the SCU by running ranking, selection and formation mechanisms. The assimilation phase has to do with the assignment of each ICU within the SCU to a certain domain of the problem to be solved, which means that tasks of a certain type will be assigned to ICUs with appropriate skills to those task types. The virtualization step is setting up the collaboration environment for the SCUs and the deployment step is actually the runtime execution and result gathering. In the dissolution phase the SCU is dissolved after member ICUs are compensated and rewarded appropriately for their performance and results. SCUs have a Cloud-like behavior, which means that ICUs can be added or excluded from them at runtime on demand or when preset thresholds regarding non-functional parameters are reached (or violated). Moreover, in SCU duration and performance depends on the goal, customer constraints as well as events generated during SCU execution. SCUs have their own compute power and this depends on the requirements of the customers and the effectiveness of the formation process in selecting the most appropriate ICUs according to requirements. In other words, the compute power of an SCU can be (to a certain degree) measured by metrics that appropriately reflect the performance and skill capabilities of ICUs as well as their cost for executing tasks. To sum up, Figure 3.2 illustrates the basic working principles for SCUs: formation, monitoring and runtime-adaptation.

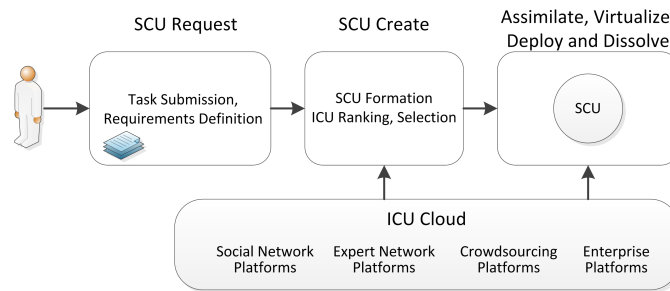


Figure 3.1: SCU lifecycle states

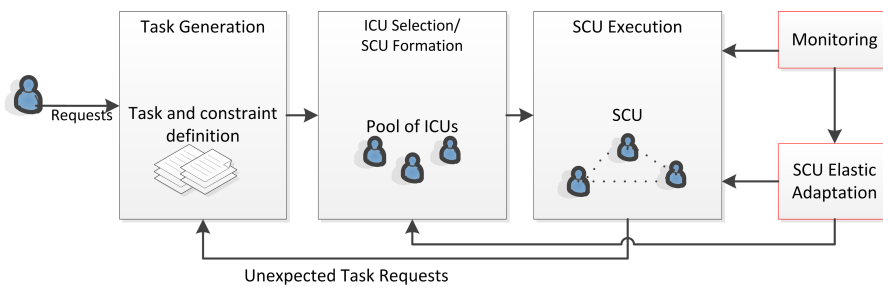


Figure 3.2: SCU basic-working concept

3.2 Motivation Scenario

Consider a scenario of a software development project. A new software consulting and development company is engaged in a health-care project and is assigned the task of developing of a health-care platform for a hospital. We assume the company is small and has a few employees with software development skills. However, delivering a product such as a health-care platform requires a diverse set of skills, some of which the company lacks. To address this issue, some of the work would need to be outsourced to other companies or new employees would be needed to be hired for areas for which the company lacks skills, such as health-care consultants. Outsourcing part of the project to other companies means that the budget needs to be shared to the services of the outsourced company which brings additional costs. In addition, hiring new employees would bring challenges related to keeping deadlines since hiring new employees is an activity that requires time to find an appropriate person. Last but not the least, even in the case when a company has acquired all the skills needed for the project unplanned problems related to the fixed number of resources allocated for the project often arise. This is due to the fact that task management is always difficult and often some employees become overloaded with work while others are idle at the same time. Badly planned work-balancing often causes delays in delivering milestone results and/or the end product. On the other hand, catching up deadlines with lack of appropriate skills affects the quality of the delivered product. To sum up, problems such as lack of skills, project delays, budget allocation issues and unsatisfactory products in these type of scenarios demonstrate the need of new approaches in managing collective work. All of these challenges come from the importance of performance and quality of results in paid expert collectives.

To generalize further the aforementioned challenges, we need to understand that traditionally even in outsourced work we have fixed collectives, in terms of number of employees as well as skill variety and optimization is done within the collectives, adaptations are done with task reassignments within the collective; decision-making is also in most cases centralized. Thus, we have vertical elasticity. However, current trends come up with new needs, today we have tailored and elastic applications to fit customer requirements on demand and cloud computing has become the paramount support mechanism for these applications. New needs are already reflected in socially-enhanced applications engaging people for a variety of tasks. Thus, we need dynamic adaptation at runtime, we need resource variability as well as decentralized decision-making to fit the collective's goals and runtime needs. Consequently, what we need today is horizontal elasticity.

Our approach of addressing the aforementioned challenges is to form a collective of experts, that we call Social Compute Unit, with the needed skills utilizing the company employees as well as experts available from human computation platforms. With the availability of online resource-pools from the so called human clouds [KCHO13], experts can be acquired and released on demand. Thus, the problem of time delays in finding part-time appropriate experts to employ for the project duration or companies to outsource the skills that are lacking can be addressed with automated online resource selection from

available human computation platforms. Hence, we assume that the outside experts can be recruited from human clouds on demand. Moreover, work balance can be achieved by automated task delegation mechanisms where tasks are re-assigned to appropriate and available experts.

These scenario assumptions and the more general challenges that we discussed, demonstrate the issues in existing work in human computation today that need addressing:

1. an investigation of provisioning mechanisms for collective work in human computation, and
2. the development of platforms that will support not only resource discovery and task assignment but also the management of runtime task execution.

The related but more concrete challenges that we focus on in this chapter however are the following:

- Which are the most relevant metrics to monitor ICU activity and task-execution in an automated way, in order to enable self-adaptive ICU and SCU management?
- Given an initial formed SCU and a set of monitored performance metrics, what are the set of actions and mechanisms that can enable SCU elastic capabilities, in situations when performance is degraded and reaches a threshold value for a customer set constraint?
- What are the design requirements of an SCU provisioning platform, having in mind the mechanisms for SCU provisioning and management?

3.3 On the Elasticity of Social Compute Units

We discussed that there is a need for management mechanisms to support elasticity by scaling in size and computing capabilities of SCUs in an elastic way. Authors in [DT12],[TDB12] identify the underlying challenge in provisioning SCU elasticity to be the lack of techniques that enable proactive provisioning of human capabilities in a uniform way in large scale. To address this and the aforementioned issues, in this chapter, we investigate and provide runtime mechanisms with the elasticity notion in mind, so that platforms would be able to provide *elastic capabilities of human-based compute units/SCUs*, that can be managed flexibly in terms of the number of resources, as well as their parameters such as cost, quality and performance time. Hence, our key contributions in this chapter are:

- defining metrics to be used in ICU monitoring and SCU adaptation mechanisms
- conceptualizing and modeling the SCU execution phase and states,
- defining SCU-elasticity properties, and
- designing an SCU provisioning platform model with elastic capabilities.

3.3.1 Definition and Principles of Elasticity

Elastic SCUs have elastic capabilities that can be triggered at runtime to tailor their performance to best fit client requirements at runtime. With human based resources being unpredictable and dynamic, their skills, price, interest and availability can change with time and within a specific context. However as stated in [DB11] the concept of SCU does not have a notion of elasticity in itself, thus an SCU provisioning platform which creates, deploys and supports the execution of SCUs needs to include mechanisms for scaling it up or down as needed, and as aforementioned, with this scale an SCUs performance parameters vary as well. These mechanisms should ensure that at each time point these parameters are within desired levels and comply with customer constraints. For our purposes, we conceptually define the elasticity of SCUs as follows:

Definition 3 *The Elasticity of Social Compute Units is the ability of SCUs to adapt at runtime in an automatic or semi-automatic manner, by scaling in size and/or reorganizing and rescheduling, such that the variations in the overall performance indicators such as capability, availability, effort, productivity and cost, at each point in time are (near)optimal within the boundaries of the customer-set constraints.*

3.3.2 ICU/SCU Formal Notation

We denote a cloud of ICUs (e.g., from online platforms and/or enterprise internal pool) as the set $R = \{r_1, r_2, r_3 \dots r_n\}$, and the set of ICUs that are members of a particular SCU as $S = \{s_1, s_2, s_3 \dots s_n\}$, where $S \subset R$. Let the set of tasks to be executed from a specific SCU be $T = \{t_1, t_2, t_3 \dots t_n\}$. For each task $t_i \in T$, we denote the set of matching, appropriate and possible ICUs that can perform the task t_i as $P = \{p_1, p_2, p_3 \dots p_n\}$, where $P \subset R$. Depending on constraints the following can be valid in different situations: $S \subset P$ (when P also contains reserve ICUs that are not included in the SCU), $P \subset S$ or $P = S$. To enable elasticity, ICUs from S can be released and new ones can be added from P to S, therefore, S might change at runtime. We use these notations in other chapters throughout this dissertation as well. We denote the collection of SCUs, in which a specific ICU, s_i has been a member over a specific time period, with $\mathfrak{U}_{s_i}^\tau = \{S_{1,s_i}^\tau, S_{2,s_i}^\tau, \dots, S_{n,s_i}^\tau\}$.

ICU-related Metrics	Description
$n_{approved}(s_i)$	Total number of successfully executed/approved tasks for an ICU
$\tau(s_i, t_x)$	Processing time for task x executed by an ICU
$c(s_i, t_x)$	Cost for task x when executed by an ICU
$c(s_i^{nw}, t_x)$	Cost for task x when reassigned to a new ICU

Table 3.1: Notation and description of basic ICU metrics and parameters

3.3.3 Metrics: Notation and Definitions

Let us look at some basic metrics for SCUs. We denote the number of all completed tasks of an SCU as: $CT(scu_i) = \sum_{i=1}^{|S|} n_{completed}(s_i)$. However, the result of all completed tasks does not always mean that all of these tasks are approved at the end when the SCU is dissolved. Thus, we also define SCU approved tasks as $AT(scu_i) = \sum_{i=1}^{|S|} n_{approved}(s_i)$. Then we have the *success rate* of an SCU defined as $ST(scu_i) = AT(scu_i)/CT(scu_i)$. Some tasks can be delegated at runtime, and their completion at another ICU different from the initially assigned one is, needless to say, also included in the completed tasks, but it is worth noting that we consider delegated tasks because they play a crucial role in adaptation mechanisms.

Project effort and *productivity* have been listed as performance measures for software projects [Kas08]. Modified versions of these metrics can be reused for SCUs on software and other goals. We define the *SCU effort* as the average time spent by each ICU on each assigned task:

$$Effort(scu_i) = \sum_{s=1}^{|S|} \sum_{x=1}^m \tau(s_i, t_x) \quad (3.1)$$

Related to this metrics we define the *productivity of an SCU* as the ratio of approved tasks to the given effort by the SCU for all assigned and completed tasks, as follows

$$Productivity(scu_i) = ST(scu_i)/Effort(scu_i) \quad (3.2)$$

This means that the productivity of the SCU shows the effective time for which the number of successful tasks with accepted results are executed.

The *cost* of an SCU is an aggregate sum of the cost of each of the ICUs for each type of tasks that they have executed as each type of tasks has different type of skill and skill-level requirements, $Cost(scu_i) = \sum_{i=1}^{|S|} \sum_{x=1}^m c(s_i, t_x)$. Table 3.2 gives a clear overview of the described metrics, while Table 3.1 gives the notation of some metrics that we use for calculating some of the metrics in Table 3.2.

3.3.4 SCU Execution Model

SCU States

As we are defining the SCU as a computational concept, we need to define its execution states as well. Thus, an SCU in execution mode, at a specific time point τ , can be in one of the following action-states, $SCU_{state}(\tau) = \{running, suspending, resuming, expanding, reducing, substituting, stopped\}$. These states are listed in Table 3.3. The mentioned states are basic/atomic ones and a combination of them makes a complex SCU execution

SCU Metrics	Definition
SCU Total Completed Tasks	$CT(scu_i) = \sum_{i=1}^{ S } n_{completed}(s_i)$
SCU Approved Tasks	$AT(scu_i) = \sum_{i=1}^{ S } n_{approved}(s_i)$
SCU Success Rate	$ST(scu_i) = AT(scu_i)/CT(scu_i)$
SCU Effort	$Effort(scu_i) = \sum_{s=1}^{ S } \sum_{x=1}^m \tau(s_i, t_x)$
SCU Productivity	$Productivity(scu_i) = ST(scu_i)/Effort(scu_i)$ =
SCU Reputation	$Reputation(scu_i) = \sum_{i=1}^{ S } w_{expertise} * reputation(s_i)$ *
SCU Cost	$Cost(scu_i) = \sum_{i=1}^{ S } \sum_{x=1}^m c(s_i, t_x)$

Table 3.2: SCU metrics to be used in adaptation mechanisms

state. For example, an SCU might be running but due to an adaptation action, at the same time multiple ICUs (a cluster of ICUs) within an SCU might be suspended, while a new ICU is being added in expanding state. In this case because *running*, *suspending* and *expanding* are all execution states of an SCU, then $running \wedge suspending \wedge expanding$ is also an SCU state. However, some states are mutually exclusive if they refer to the whole SCU and cannot be aggregated, i.e., an SCU cannot be in $running \wedge stopping$ state. If one of the atomic states refers to (a change in) individual or a cluster of ICUs, an SCU can be in $running \wedge extending$ state or for example an SCU can be in a $running \wedge reducing$ state. Thus, the aggregate states are valid in the context of the scope that a state-changing action takes place. Table 3.3 also shows the scope for which the state-changing actions are valid, in terms of the whole SCU, a cluster of ICUs, or ICUs only. The importance of the state of an SCU as a whole is tightly coupled with ICU states and is crucial when applying elastic strategies in two ways: 1) the state of the SCU can be a trigger for elastic operations on the SCU, and 2) it can be a desired result after applying these operations.

SCU in Execution

Table 3.3 shows ways of adaptation triggering: platform based, customer based and ICU based. To clarify, a platform that supports an SCU should have the mechanisms to support *all* of its execution states elastically. Thus all state-changing actions can be

Trigger action	State	Scope	Triggering Role		
			Platform	Customer	ICU
Run	Running	SCU	✓	✓	
Suspend	Suspending	SCU/ICUcluster/ICU	✓	✓	✓
Activate	Resuming	SCU/ICUcluster/ICU	✓	✓	✓
Add	Expanding	ICUcluster/ICU	✓	✓	✓
Exclude	Reducing	ICUcluster/ICU	✓	✓	✓
Stop/Exclude/Add	Substituting	ICUcluster/ICU	✓	✓	✓
Stop	Stopping	SCU	✓	✓	

Table 3.3: Fundamental state alternatives of the SCU Execution phase

triggered in an automated way as shown in Table 3.3. Referring to our motivational scenario, in rare cases the customer could suspend the whole SCU of software development until he has consulted and decided for crucial changes. There are other triggering state-changing actions that the customer can also make (shown with light gray check signs). Table 3.3 also shows which state-changing actions can be most affected by communication and ICU feedback, which we illustrate in Section 4. We show an example for a software

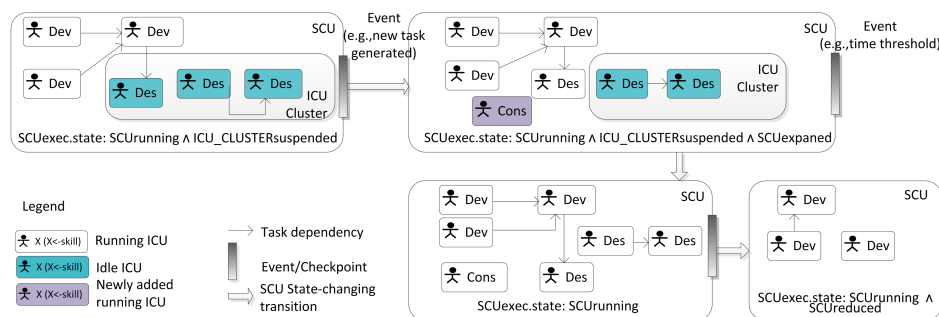


Figure 3.3: An illustrative example of an SCU in execution: expanding and reducing states

developing SCU in execution mode in Fig. 7.3. At a specific time point ICUs with *developer* skills are in running state while designers are suspended. Next, due to an event when expert information is needed (e.g., health-care information in our scenario), the SCU is expanded by including ICU with specific expertise and consultancy skills while a designer-ICU is resumed. At another time point each ICU is running, while before dissolving, the SCU is reduced as ICUs with designer and consultancy skills have finished their tasks. Adaptation actions on an SCU can change its execution model not only in terms of the state but also in terms of its execution structure. These changes are interdependent with task structure changes and ICU state changes.

Scheduling methods	Description
abstract AddICU()	adds an ICU to the SCU
abstract void SuspendICU(SCU scu)	brings an ICU to idle state, still included in the SCU
abstract void ExcludeICU(SCU scu)	excludes an ICU form the SCU
abstract void ResumeICU(SCU scu)	restart an ICU and its associated tasks
abstract void ReserveICU(Task t)	reserves an alternative ICU for an already assigned task
abstract void SubstituteICU()	substitutes an ICU with a reserved one
public List<ICU> getAllICUinSCU(SCU scu)	returns ICUs within the SCU
public List<ICU> getSuspendedICUs(SCU scu)	returns suspended ICUs within an SCU
public List<ICU> getIdleICUs(SCU scu)	returns idle ICUs in an SCU
public List<ICU> getReservedICUs(Task t)	maintains an ordered list of top appropriate ICUs for a certain task (ICUs might be in/out of the specific SCU)

Table 3.4: Example API, abstract methods for ICU manipulation

Elasticity APIs

To be able to test mechanisms for SCU elasticity capabilities, which include ICUs having the aforementioned (and other domain-dependent) properties, we designed a proof of concept prototype, where we modeled ICUs, SCUs, their description and management classes. We designed classes which we categorized in ICU-description Interfaces for manipulating ICU profiles, ICU-scheduling Interfaces for ICU management and elastic operations, and communication operations. Table 3.4 describes some specific methods that we develop to be utilized in strategies providing SCU elastic capabilities.

3.3.5 Elastic SCU Provisioning Platform

Figure 3.4 shows a model of our concept of an elastic SCU provisioning platform, that utilizing our SCU execution model, including metrics, is able to support elastic SCU management. Thus, the platform supports the following behavior: a customer/SCU consumer submits a project/request with multiple tasks to it. When submitting tasks and request for SCU formation, the client specifies functional and non-functional ICU requirements such as: skill, reputation and cost. In addition he specifies overall SCU constraints, such as trust, total budget and deadline. The platform integrates an SCU formation component with ICU selection/ranking algorithms. The SCU creation/formation component's output is an initial SCU created by selecting ICUs from human cloud providers. This SCU is "fed" to a *controller*-a component that hosts monitoring and adaptation algorithms utilizing APIs for elasticity control, which provide SCU runtime

management. The challenge of this component, is to monitor and adapt the SCU in accordance to customer set constraints, such that the SCU gives the maximum performance and quality within the preset boundaries for time related, cost and quality related indicators. Different scheduling and ICU management algorithms can be plugged into the platform, which would support the SCU during its lifecycle.

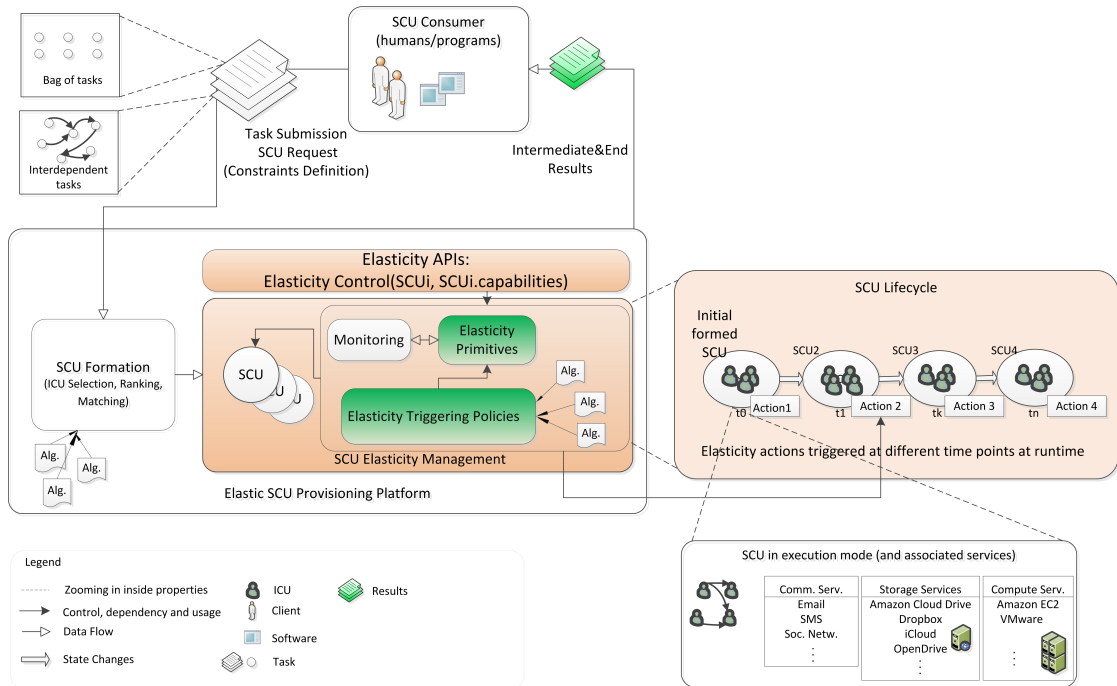


Figure 3.4: Conceptual platform model supporting elastic SCUs

3.4 SCU Runtime Management: Elastic Adaptation Mechanisms

In this section we show the benefit of having an explicit execution model for SCU. We present an implementable adaptation mechanism to illustrate the usefulness of our framework in simplifying the complexity of the development of runtime mechanisms to support elastic adaptation of SCUs. Typically, an elasticity strategy for an SCU would depend from the domain of an SCU execution. In other words, much of the adaptation mechanisms depend on the type of tasks that are going to be executed by the SCU and the task-interdependence. However, as part of our framework we provide a few basic algorithms that can be easily extended to tailor them to specific domains. In the following, we provide an elastic adaptation algorithm to demonstrate how an *ICU Feedback-based elastic SCU management strategy* can be implemented.

3.4.1 Programming an Elasticity Strategy: A semi-automatic adaptation strategy with human-in-the-loop decision making

As ICUs within an SCU are inherently dynamic and unpredictable due to human nature, we cannot always fully rely on the system-based availability information concerning an ICU and fully automated task assignment and scheduling might not always be the most suitable approach, especially when there is a possibility of unexpected generation of tasks at runtime. Hence, we propose an SCU adaptation strategy that uses system requests and corresponding ICU acknowledgments for their *willingness* to work on specific tasks. More specifically, the acknowledgments are sent in response to system requests for availability guarantees. These availability guarantees are requested in two cases:

1. for specific tasks that would need to be executed in the future during SCU execution, which means that willingness requests can be sent for a task before it is assigned, and multiple ICUs that send acknowledgments as a reply to willingness requests stating that they are willing to execute particular tasks can be used as reserve resources, and
2. at run-time for immediate execution of tasks that need reassignment, where ICUs who send willingness ACKs are ranked so that tasks can be reassigned to the most appropriate ICU at run-time.

Our example of elastic SCU mechanism is a semi-automatic task scheduling strategy where part of the coordination and decision-making for task re-assignment is delegated to ICUs. With this approach a task is being re-assigned to a more available ICU, on an ICUs own approval and when certain conditions apply (e.g, when a threshold is reached). Thus, the task reassignment decisions are partly based on feedback from ICUs and in this way the elastic SCU management is not completely automated but involves “human in the loop” decentralized coordination. With this example, we show how new SCU metrics can be derived and how APIs for elastic capabilities can be used. By utilizing programming interfaces for obtaining SCU metrics at runtime, we can calculate the willingness of an ICU. We define *ICU Willingness* as the ratio of the number of acknowledgments sent by ICUs to the number of willingness requests sent to the ICUs, as in the following:

$$ICUWillingness = \frac{SentAcks}{ReceivedReq}. \quad (3.3)$$

Related to the willingness metric we define and derive another metric to help us further quantify the task execution performance of an ICU, namely the Confidence-Score for the ICU Willingness metric, that we naturally call *Willingness Confidence-Score*. We derive it from the basic indicators, *ICU Willingness*, and the ICUs *rate of success in executing the reassigned/delegated tasks*. More specifically the ICU Willingness Confidence value or score, is computed from the number of acknowledgments that an ICU has sent to the scheduler in response to its *requests for willingness*, and the number of successfully completed tasks from the ones that are delegated to it as responses to those

acknowledgments. In other words, Willingness Confidence-Score of an ICU which we define as a product of Willingness and the rate of success in executing the number of tasks that are delegated to the ICU as a result of its own feedback for willingness to execute those tasks, as in the following:

$$WillingnessConf = ICUWillingness \times \frac{DelegatedTasksExecuted}{TotalDelegatedTasks} \text{ [RTD14]}. \quad (3.4)$$

Thus, the Willingness Confidence-Score is a measure of the *delegation reliability* of an ICU for tasks that have been delegated to it, because it shows how true to its own statements an ICU is by accounting how many tasks it has executed of those it claimed that it is willing and going to execute.

To detail our adaptation strategy that utilizes willingness and willingness confidence-score we assume that each incoming task is assigned to the ICU at the top of a ranked list that is returned by a ranking algorithm, and references to the first x most appropriate ICUs from the ranked list are stored as reserves/alternatives for each task. The algorithm can be summarized with the following steps:

1. When a preset threshold, related to a task which is already assigned to the most appropriate ICU matching the requirements is reached, e.g., the tasks waiting-time in an ICUs task-queue, the scheduler sends a willingness request for executing that particular task to the next top x number of ICUs that it has references to (reserves from the initial ranked list). These appropriate ICUs at the same time need to be idle, or their task queues need to be smaller than that of the ICU to which the task was initially assigned and at which it reached a threshold. With this request for willingness, a scheduler notifies the reserve experts that there is a task that they can work on. This request is a resource availability-check. In other words, it is a request for a resource's *willingness* to work on a specific task as a form of a worker-side commitment and a form of a *guarantee* that the task will be executed by him/her.
2. Each ICU that receives this request and is ready and wishes to work on the task, sends the scheduler a willingness acknowledgment(Ack), which at the same time is a positive feedback to this request.
3. The scheduling component reassigns the task on threshold to the alternative resource that has sent a willingness acknowledgment and that is idle or has the smallest task queue. Priority is given to idle or less loaded ICUs that are already members of the SCU. Thus, the task is assigned by going in descending order through the ranked list of reserve ICUs that have sent Acks and assigning the task to the first found ICU that is idle or has a smaller task-queue than a preset task-queue value.

This type of scheduling combines the freedom of choosing tasks that workers have in crowdsourcing environments, with policy based assignment of tasks. It is these ICU-side guarantees combined with task queue analysis, that can avoid problems such as *delegation sinks*. The steps of our strategy are given in the pseudo-code in Algorithm 3.1, which

Algorithm 3.1: SCU Adaptation: Task-delegations with ICU-side assurance

```

Data: scuTasks for SCU
Data: customer constraints on NFP
1 forall tasks in T do
2   rank matching ICUs and return the first 10 appropriate ;
3   P ← getReservedICUList(Task t) /* store reserve ICUs */
4   assign task t to top ranked ICUs r;
5   if r is not an element in SCU then
6     | SCU ← addICU() /* add ICU r to SCU x and update its profile */
7   end
8   if task.taskQueueTime == task.timeThreshold then
9     if r == idle then
10      | SCU ← removeICU() /* reduction: remove ICU r from SCU */
11    end
12    forall ICU in P do
13      | getICUState(ICU ICUid) ;
14      | if ICU_ STATE==idle AND icuReserve.tQueue()
15        | <r.tQueueSize()/2 then
16          | willingnessReqMessage() ;
17        end
18      end
19      forall ICU in P in ascending order of icuResource.taskQueue do
20        if icuReserve.sentAck == true then
21          | substituteICU() /* delegate task to another ICU */
22          | ;
23          | // check if the ICU already belong to the SCU or not, if
24            | not include it in
25          | if !SCU.contains(icuReserve) then
26            | | SCU ← addICU()
27            | end
28          | ;
29          | Break ;
30        end
31      end

```

also shows how the concept of elasticity in social computing departs from the idea that a customer knows in advance which and how many experts will contribute to the project and what the final cost will be. However, the customer budget is kept within its limits as the cost may vary within these limits, just as the size and structure of the assembled SCU may vary with time until the final result is returned. When a delegation is executed, the new cost calculation includes the price of the new ICU. Thus, the cost of an SCU is adapted with each delegation as follows:

$$Cost_{adapt}(scu_i) = Cost_{previous}(scu_i) - \sum_{i=1}^m \sum_{x=1}^j c(s_i, t_x) + \sum_{i=1}^m \sum_{x=1}^j c(s_i^{nw}, t_x) \quad (3.5)$$

, where $Cost_{adapt}(scu_i) \leq Allowed\ Budget$.

3.4.2 Experiment: Executing an Elasticity Strategy

Utilizing an Analytic Hierarchy Process (AHP) model for ranking of ICUs

For the implementation of the described adaptation strategy, we need to form an SCU first. For SCU formation in this case we used an Analytic Hierarchy Process (AHP) model, and more specifically a modified version of the methodology presented from authors in [CFMT09]. In a nutshell an AHP mechanism is used for decision-making scenarios based on multiple metrics, complex or simple. We use it to rank our ICUs for each task according to customer constraints and thus form an SCU with appropriate ICUs. Our approach to it and our implementation considers the following steps:

- Get the non-functional parameters for which a customer sets a constraint, as an input (to simplify we require a set of three metrics with which we rank ICUs). Thus, the input is a list of three metrics and required values for them.
- Next we request customer input for the importance of the three metrics, as follows: 1 means equal importance, 2 means moderate importance and 3 strong importance. Then we create a comparison matrix for the three metrics according to the importance of each of them in relation to each other, which is different from the method of authors in [CFMT09], where they compare the importance of sub-characteristics of one metric, but for our needs we modify the model and compare metrics without sub-characteristics such as average and maximum values of a metric. Thus, we create the matrix by following the formula $k(i, j) = \frac{1}{m(j, i)}$, $\forall i, j$, where i and j are metrics and not sub-characteristics of a metric, so here we differ from the model in [CFMT09]; $k(i, i) = 1, \forall i$.
- In the next step the matrix is normalized and a weight for each metric is calculated with the following formula: $w(i) = \frac{\sum_{j=1}^n m'(i, j)}{n}$, $\forall i$, where $m'(i, j)$ is the normalized matrix.
- Finally, w is multiplied with the results of a utility function in relation to the ratio between the requested value for the metric from the customer and the offered value

from an ICU. The sum of this multiplication for all metrics is the ranking score of the ICU. The formulas to describe our AHP method are all from [CFMT09]. We modified the version of the authors model to fit our purposes and utilize this AHP-based algorithm to rank ICUs in the experiment with the implementation of the adaptation strategy that we describe in this section. Just for illustration, Appendix A, shows a code snippet from the method of getting a satisfaction score for ICUs based on three metrics, and ranking a list of ICUs based on that score, and one short code snippet from our delegation mechanisms based on a time threshold.

Of course, there are other methods that could be used for ranking ICUs based on multiple criteria, such as Logical Scoring Preference, and different genetic algorithms (-we present a genetic algorithm based SCU formation strategy in Chapter 5). We used the AHP method to help us in the formation of SCUs, as our elastic adaptations take as input a ranked list of ICUs when delegating tasks.

Implementation of Algorithmn 3.1 and results

We implemented our previously described elastic adaptation algorithm using methods described in the API section with a Java based simulation. We created tasks with different skill requirements and modeled ICUs with a single skill for simplicity, using a few different types of skills (e.g., design, development, database-management, tester). We also assigned different costs to each of the ICUs, and to each of the Tasks based on skill-type. New tasks were generated in multiple steps, after every bag of tasks executed. The results of our experiments show that SCU productivity raises with the number of ICUs, while it declines if the effort is high for a low number of tasks. The productivity declines in case a small number of tasks are executed by a high number of ICUs, which means they are first assigned to other ICUs and then delegated to others. Sub-figures a) and b) of Figure 3.5 show productivity and effort in relation to the number of ICUs, number of tasks, and number of delegated tasks as a result of the SCU adaptation based on our algorithm, after each checkpoint in 10 time checkpoints, in each checkpoint we assigned a new bag of tasks to appropriate ICUs.

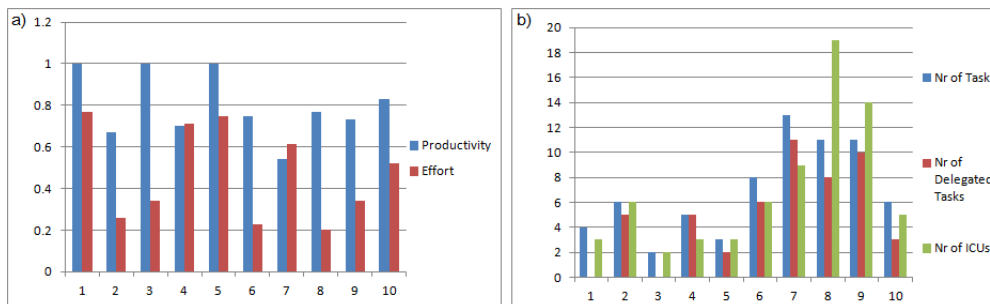


Figure 3.5: SCU Productivity and Effort in relation to Task and ICU number

3.5 Related Work

Resource Management and Adaptation. We discuss here specific works related to resource management and load balancing that are closely related to our elastic adaptation strategy. Work on a retainer model for crowdsourcing environments and examples of its application are presented in [BKMB12],[BBMK11]. The model is designed for recruiting guaranteed workers by paying them a small additional amount, and in this way keeping them in reserve and ready for handling real-time tasks. By being paid they are asked to be in ready state when a task from a specific client arrives. This work has also improved existing retainer models on minimizing task waiting times by using predictive recruitment. With this approach, by using the queuing theory M/M/c/c model the authors find the probability when a task will arrive and pre-recruit a worker from the retainer pool for it. The similarity of our ICU-feedback based task-assignment strategy to the aforementioned work is in that our scheduler keeps references to the top x number of resources that are previously ranked as most suitable for a specific task. Hence, these resources are the reserve resources in our approach. However, the difference in our approach is that no prior payment is made for reservation of these resources, rather the scheduler sends them a notification asking for feedback for their willingness to execute a task. In addition, the request for willingness messages are sent for tasks that are already assigned to another resource and which need to be delegated (e.g., for which a time-based threshold is reached). Our strategy is not concerned with initial task assignment and it is not intended for crowdsourcing tasks, although ICUs may be invoked from a crowdsourcing platform. For illustration, and an additional justification of our willingness metric, the work presented in [SKS12] is worth mentioning, where authors have investigated factors for workers' task-choice in human computation/crowdsourcing markets. Some of the mentioned ones are: abilities fit, payment fit, time fit-related to task deadline and task familiarization, as well as work intention, where willingness is mentioned as a person's overall impression about the task and his/her decision on whether to work on it. In our work, we try to define willingness, and in such a way that it can be calculable so that it provided us with a one type of a reliability metric (based on delegated tasks), indicating the behavior of a person in terms of words vs. action.

There is a considerable amount of work conducted on adaptation and more interestingly on self-adaptation strategies. For example, the authors of [PJS⁺10], who we have mentioned in Chapter 2, along the presented architecture that includes a self-adaptation framework for service-oriented collaboration systems, have presented their approach on identifying worker misbehavior patterns (e.g., as a result of uncontrolled task delegations) and provided a solution of reassigning tasks to other alternative resources by taking into account their task-queue size. Our feedback-based approach is closely related to their delegation mechanism in that it also takes into account task-queue sizes but differs from their approach in that tasks are not delegated if resources with small task-queue sizes are not willing to accept tasks. In our approach, a scheduler/controller manages the task reassignment with consent from workers which are considered as alternatives (i.e., are most trustworthy and have small task-queue size). The work presented in [HC08]

describes a delegation model and related algorithms that concern trust updates. The authors mention adoption as a process where an entity accepts an object from another one. Our willingness-based adaptation algorithm is related to the adoption concept, as tasks are delegated only with workers consent.

Elasticity. The notion of elasticity is treated in several domains and contexts and has especially gained importance with the advance of cloud computing. In [DT12] authors discuss the reasons, challenges and their approach toward virtualizing humans and software under the same service-based model that will enable elastic computing in terms of scaling both software and human resources. The concept of elasticity in cloud computing, is being extended to concepts like application [ZKJG11] and process [DGST11] elasticity, e.g., in [DGST11], the authors identify *resource elasticity*, *cost elasticity* and *quality elasticity* as being crucial in modeling processes in service oriented computing. Mechanisms and a middleware to support scaling services in and out from applications utilizing SaaS is presented in [KHCK13].

Trust in Social Computing: Metrics, Model and Algorithms

4.1 Background and Motivation

4.1.1 Trust for Social collectives

In the previous chapter we introduced a number of metrics with which it is possible to monitor ICU performance and at least to some degree quantify human performance in elastic social computing constructs such as SCUs. However, a crucial parameter is missing to make the picture more complete, namely *trust*. Trust has been extensively researched in the area of social networks, e.g., in [ZF11] and [Gol05] as well as in crowdsourcing, e.g., [AIB⁺]. Moreover, existing works that treat human task execution within service oriented environments also include trust, such as [SSD10b]. In short, trust has been identified as an important indicator of the appropriateness of people for sharing content in a certain topic, exchanging information, and collaborating in various domains that require online task-execution.

For social networks and crowdsourcing platforms it is already identified that trust is important in deciding with whom to establish connections and with whom to interact [Gol05], [ZF11], [AIB⁺]. We stress that this importance is even higher in SCUs because they entail complex structures and organized work. In the previous chapter we argued that ICUs should be managed *elastically* for effective SCU performance and customer cost savings. This implies that ICUs can be added and removed from the SCU at runtime based on different strategies, so that the SCU capabilities and performance are optimized at any time and for any changes in customer requirements. Hence, as much as trust is needed at the stage of the formation or selection of SCUs, it also plays a crucial role during their lifecycle for runtime adaptation, i.e., for *managing* SCU execution and thus *controlling* the level of non-functional parameters and quality of the returned results.

4.1.2 Motivation Scenario and Challenges

Consider an enterprise that is hired as a contractor to do infrastructure maintenance of buildings in multiple smart cities. Clients may vary from governmental institutions, to private businesses, to individuals. All clients have different types of maintenance requirements for different type of infrastructure. Usually maintenance today operates such that infrastructure is monitored and when a malfunction is detected experts are engaged to fix it. All this is done through predetermined processes and procedures. However, problems arise in these setups as maintenance is based on momentary detection of malfunctions. In addition, the expert technicians sometimes can not tell whether the component problem arose because it was malfunctioning or the predefined maintenance procedure for increasing components effectiveness was itself faulty. These problems can be avoided if they can be predicted. Thus, a maintenance process is much more effective if it is predictive. Consider for example the maintenance of multiple chillers. Chillers have very high cost if they need to be replaced, their component replacement costs are also very high and most of the time electricity costs are high. Thus, predicting a malfunction early can help tune a chiller's parameters so that it does not suffer damage, it can help preserve components, as well as high operational costs. Three types of SCUs can be utilized in this scenario:

- (a) an SCU with *data science* experts, who will collect monitoring data from facilities of different clients, they will filter this data, structure it and provide it as Data-as-a-Service (DaaS). The data to be monitored, gathered, and provided in a way that can be utilized by proactive processes, may be metrics values and properties of the chiller and its components, such as: the temperature of the water going in and out, the compressor state, the evaporator state, the water flow, the state of the condenser, the on/off status of the chiller but also environmental data;
- (b) an SCU with experts in *data analytics*, who will implement predictive maintenance procedures using the DaaS that the aforementioned SCU will provide;
- (c) an SCU with technical expertise who will react according to the predictive procedures and do maintenance on-site before any malfunctions happen.

Selecting not only the appropriate members according to specific requirements but also the best among the available ones is important for forming an effective SCU. From the system perspective, the aforementioned SCUs should have a common collaboration platform, which will at the same time provision and manage the task-execution and the SCU members at runtime. In cases of incidents regarding a malfunction of a certain component of a chiller, new expertise might be needed that the members of the already formed SCU lack. In these cases, the SCU should be extended by including new members with the required expertise and the common collaboration and SCU management platform will have the mechanisms to support these type of elastic adaptations. Figure 4.1 illustrates the described scenario, whereas Figure 4.2 extends the description of the platform from the previous chapter with trust considerations.

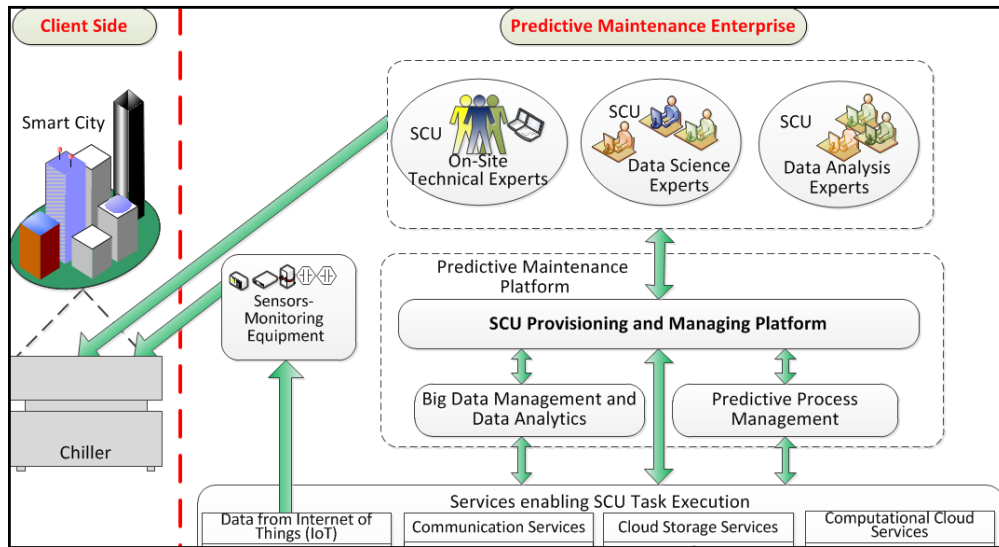


Figure 4.1: SCU Working Environment in a Predictive Maintenance Scenario

The envisioned SCU provisioning platform keeps a record of registered ICU profiles, including their expertise, and logs their information regarding the SCUs in which they have been included. Hence, it maintains SCU profiles as well. SCU data such as the domain and skills utilized, the number of ICUs, the cost, success rate etc, need to be logged not only to extract data regarding its members when they are assessed for inclusion in new SCUs, but also for strategies where an SCU is not *formed* from scratch for a new project, rather if SCUs exist for the same problem domain, SCUs can also be ranked and one or more of them *selected* for a new project. The ICU profiles may be registered and hosted on the SCU provisioning platform but they can also be references to other resources from other online resource pools such as crowdsourcing platforms, expert networks and social networks. The platform runs SCU formation algorithms to create new SCUs from trusted ICUs, or it can run a ranking algorithm for whole SCUs to chose the most trusted one, depending on the submitted client request. The selected SCU, or the newly formed one, is monitored at runtime and may be elastically adapted in response to different events, and based on different elastic strategies to best fit customer requirements. The adaptations include activities such as task reassignments, delegations, addition of new ICUs and removal of existing ICUs. Thus, the capabilities and expertise of ICUs can be adapted, and consequently, the performance of the SCU can be changed at runtime. The SCU collaboration can be supported by communication services, cloud services, and also data from Internet of Things devices.

Observations and Challenges

From the described environment we derive the fundamental trust related observations in SCU provisioning, generalized across domains:

- Because an SCU is based on expertise and task execution, a trust (and reputation)

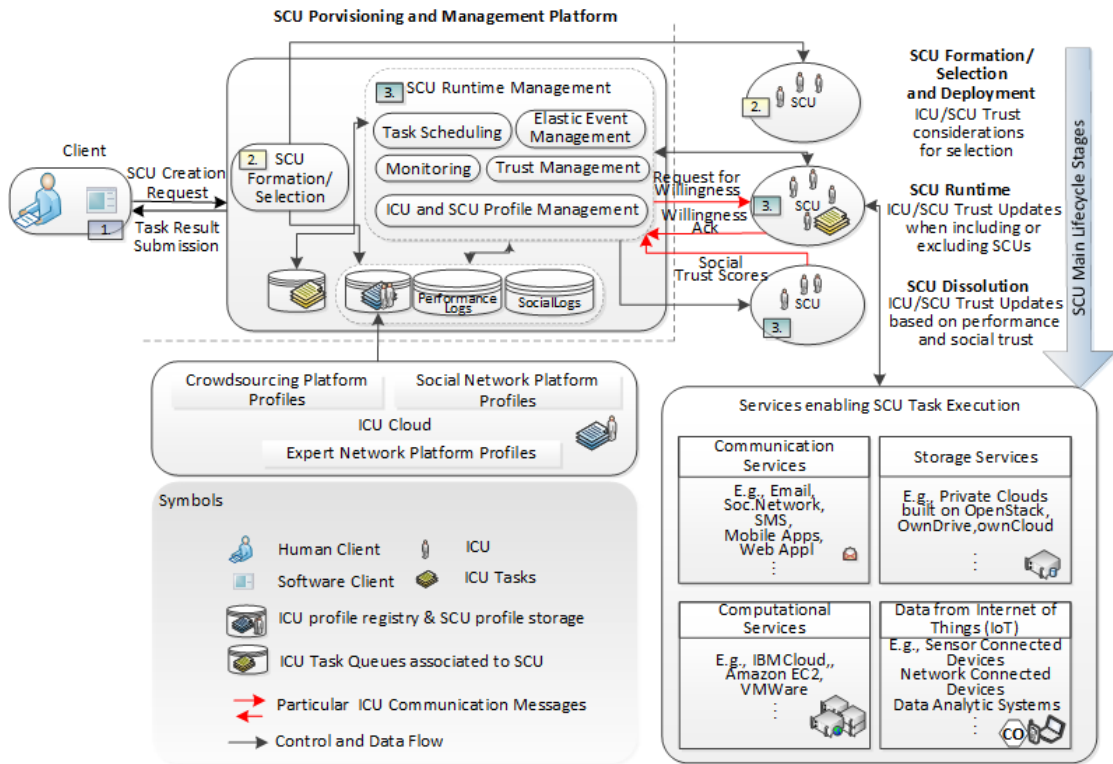


Figure 4.2: Platform/SCU Management with Trust

score of an ICU to be included in the SCU should be calculated both from the social trust scores of collaborators out of interaction satisfaction, and from its performance within the required expertise.

- SCUs are elastic in term of number of ICUs, topology, price and NFP. Consequently, the SCU trust is an elastic property as well.
- SCU is a socio-technical formation that has ICUs (people who offer their capabilities as services) as its core resources, thus its nature is unpredictable and trust is crucial when regulating their behavior.

These observations bring us to the following corresponding salient research challenges:

- What are the fundamental metrics to be included in a socio-technical trust model that will include both the social and performance contexts of ICU?
- How is trust updated and how it fluctuates within the dynamics of an elastic SCUs lifecycle?
- How can trust be included in elasticity-based management strategies for effective SCUs?
- What can be a feasible trust-based incentive strategy for ICUs?

In the following sections we present a Socio-Technical Trust (STT) model for ICUs and a Socio-Technical Trust model for SCUs. In addition we present elastic adaptation

mechanisms which consider trust, as well as incentive mechanisms for ICUs based on trust. Through these models and mechanisms we address the aforementioned challenges.

4.2 A Socio-Technical Trust Model for Social Compute Units

4.2.1 Modeling Trusted Individual Compute Units

To come to an SCU trust model, understandably we first need to define ICU trust as ICUs are the core resources of an SCU. ICU trust can be defined from two perspectives:

1. that from an SCU of which it is a member, and
2. from the global perspective of its performance and social impression, derived from each of its task executing engagements within all SCUs of which it has been a member during a certain time-line.

The trust defined from the perspective described under 2), implies a global trust score for an ICU, and aside from calling it the global STT we also call it reputation of the ICU to differentiate it from its local trust score within a specific SCU. Therefore, we present here two types of trust models for ICUs, local and global STT.

4.2.2 Metrics: Notation and Definitions

Based on the work from Chapter 3 on ICU performance as well as on related work, we have identified key performance indicators for ICUs and SCUs that can be used in ICU/SCU elastic management. We discuss them in this section, even also some that are introduced in Chapter 3 as they are part of the core of our socio-technical trust model, and their elaboration has the purpose of easing the flow of our model discussion.

Effort- the effort of an ICU is the average time spent by an ICU for executing a task, thus it indicates timeliness. The SCU effort is an aggregate of the effort of its members-ICUs, as in Equation (3.1).

Productivity- we define ICU productivity as the number of successfully executed tasks over a time unit. Thus, productivity is the ratio of successfully executed tasks that are submitted by an ICU as a result, and the total number of tasks executed per time unit. The SCU productivity is an aggregate of the productivity of the constituting ICUs, or it can be calculated as the ratio of approved tasks to the given effort by the SCU for all assigned and completed tasks, as in Equation (3.2).

Consistency- the performance of ICUs over time is important in some cases, and even-though we haven't used consistency in our model and experiments we need to mention it as an important metric that could be used as one of the atomic metrics to calculate reliability. In some cases, the performance of ICUs gets better with time as ICU skills get better, so this metric would be relevant in some cases (e.g., number of

pages translated, number of objects tagged) and not in others; consequently, we chose not to generalize it. To define it for our purposes we use the standard deviation over the productivity values of a specific ICU in different SCUs within a specific time-interval or in every SCU it has taken part over time.

Reliability- we derive ICU reliability from multiple atomic metrics. Namely, in Chapter 3 we presented two novel metrics, Willingness and Willingness Confidence-Score. Reliability in our trust model is partly indicated by the Willingness Confidence Score, because this value shows how true to its own statements an ICU is by accounting the number of tasks it has executed of those it claimed that it is going to execute when it was sent requests for task executions. To define a comprehensive reliability metric for an ICU we combine the delegation reliability with the total successfully executed tasks of an ICU, including tasks that are initially assigned to it. To calculate it, we use a weighted sum of the delegation and success rate. If we want to include consistency in calculating reliability then we would add it in the weighted aggregated sum with the values in negative sign (as we need a small standard deviation of productivity over time if used).

Quality of Results/Client Satisfaction Score- the quality of tasks results (QoR) depends on multiple factors, such as the domain of the SCU goal, the task types, the SCUs structure [18]. In this work we define it as the client satisfaction from the task results. Thus, in our model this metric is calculated as a vote from the client to the SCU, and denoted as Client Satisfaction Score. However, QoR can be set up to be automatically measured in some cases when expected task results are specified, or defined differently for different domains. Related to this metrics we define the *productivity of an SCU* This means that the productivity of the SCU shows the effective time over which the number of successful tasks with accepted results are executed.

System-based Satisfaction Score-The System-based satisfaction score STTR, is a value that is assigned to all ICUs at the end of the execution (SCU dissolution) based on their collective performance. This metric can be used as an incentive mechanisms to encourage people to have better collective performance, because a higher value of a collective's trust signifies that an ICU will be given a higher STTR. This metric is up to the developer to define, but we postulate that to be used as an incentive mechanism it should be a value equal for all members of the SCU. In our model we define it as $STTR = STT(scu_i) \div |S|$, where S is the set of members of scu_i . Of course with our formula, STTR will be very small, but it is enough to have an effect nonetheless.

4.2.3 Socio-Technical Trust (STT) Model

In the following subsections we discuss how we use the mentioned metrics in trust models for ICUs and SCUs. We define an ICUs software-based trust, called performance trust as the product of its productivity and reliability, which are calculated from automated monitoring.

$$PT(s_i) = Productivity(s_i) * Reliability(s_i) \quad (4.1)$$

To come to a social trust score of an ICU in the context of an SCU we define the *Membership Collaboration Trust Score* of an ICU within a specific SCU. For this metric we propose the strategy that each ICU within the same SCU votes for all other ICUs of the same SCU before the SCU is dissolved. This vote is given for the interaction satisfaction with the ICUs with which the voting ICU has interacted within the same SCU or, in the case if it has not interacted with, the vote is cast based on the perception of what the ICU to whom it casts a vote has contributed for the SCU. Thus, we assume here that every member has either interacted with all others at least once or if not then at least it knows what all members have contributed. Consequently, the Membership Collaboration Trust Score of an ICU is calculated from the votes that each member of the SCU gives to another member of the SCU before the SCU is dissolved as in Equation (4.2). This metric is a weighted aggregation of scores, such that the votes of ICUs with higher reputation are given higher weight. In this work, in all equations the sum of the weights is 1, and every metric value is in the $(0,..1]$ range.

$$MCTS(s_i) = \sum_{k=1, k \neq i}^{|S|} w(s_k) * mc(s_k, s_i) / |S| - 1 \quad (4.2)$$

From the global perspective, the *Global Membership Collaboration Trust Score* of an ICU in regard with all SCUs that it has been a member of, is an aggregate of its MCTS scores from every SCU of which it has been a part of, taking into account the number of invocations of each SCU, because an SCU can be invoked multiple times for the same or different clients. In addition, because human behavior is highly unpredictable and changes with time, the inclusion of a time restriction is important. Thus, a time period limit for the number of SCU invocations, has to be assigned in calculations, with the purpose of accounting for the freshness of trusted relations. Equation (4.3) shows the definition.

$$MCTS_{gl}(s_i, \tau) = \frac{1}{|\mathcal{U}^\tau|} \sum_{j=1}^{|\mathcal{U}^\tau|} \frac{1}{m} \sum_{l=1}^m MCTS(s_i)_{l,j}, \quad (4.3)$$

where m is the number of invocations of a specific SCU, of which an ICU has been a member, and \mathcal{U}^τ is the total number of different SCUs that the ICU has been a member of, within a specific time τ .

The local Socio-Technical Trust of an ICU (in the context of a specific SCU invocation) $STT(s_i)$ is a weighted sum of its performance trust and the Membership Collaboration Trust Score from its co-members in the SCU, given in Equation (4.4).

$$STT(s_i) = w_{pt} * PT(s_i) + w_{mcts} * MCTS(s_i) \quad (4.4)$$

On the other hand, we name the global Socio-Technical Trust Score of an ICU, *Reputation*(s_i). This is in line with the concept that trust in an ICU is individual, whereas reputation is a global metric that includes the trust scores from all the actors in the system who have interacted with the ICU. We calculate the reputation of an ICU as

a sum of its STT and a Socio-Technical Trust score that we assign to it based on the STT scores of the SCUs to which the ICU have belonged, considering all invocations of the ICU within different SCUs in a specific period of time.

$$Reputation(r_i, \tau) = \frac{1}{|\mathcal{I}^\tau|} \sum_{j=1}^{|\mathcal{I}^\tau|} \frac{1}{m} \sum_{l=1}^m (w_{stt} STT(s_i)_{l,j} + w_{str} STTR(s_i)_{l,j}). \quad (4.5)$$

Context

To account for the possibility of multiple capabilities of people, it is important to note here that our metrics for calculating the Socio-Technical Trust for ICU and SCUs are calculated in the context of a particular skill or expertise of an ICU, for which an ICU is invoked in the SCU; e.g., a data scientist can conduct tasks related to structuring and filtering data to provide DaaS but he/she can also be involved in another SCU for analyzing data and creating operations plans. Consequently, in fact we have separate roles for the same ICU, so the same ICU is treated as separate in the context of its expertise within different roles (icu1 in Figure 4.4 after SCU adaptation during time τ). Tasks are assigned based on a match of the skill needed to execute the task and the skill that an ICU possesses. We define the set of skills of an ICU with $SK(s_i) = \{sk_1, sk_2, \dots, sk_n\}$, and the set of tasks assigned to an ICU with $T(s_i) = \{t_1, t_2, \dots, t_n\}$, where $\forall t \in T, \exists sk \in SK$. The set denoting the skill-task/s pairs is $\mathfrak{ST} = \{ST_1, ST_2, \dots, ST_n\}$, with its elements defined as $ST = \{(sk, \{T_{sub}\}) \mid T_{sub} \subseteq T \wedge T_{sub} \neq \emptyset \wedge sk \in SK\}$. The context C , in which we calculate a metric M for an ICU corresponds to the skill type with which the ICU is invoked in an SCU, so we have $C(M, s_i) = sk \Leftrightarrow sk \in ST$.

4.2.4 Modeling Trusted Social Compute Units

We consider two cases where trust is important for SCUs: a) when an SCU is newly-formed and composed from ICUs with appropriate expertise for the SCU tasks, and b) when the client may chose an already existing SCU. For SCU formation algorithms, we propose an aggregated trust score that is based on the reputation score of ICUs over which the selection algorithm is executed. As every skill does not have the same importance within an SCU, weights are assigned to ICUs with different type of skills and for each skill-type. For example in our scenario, data scientist members may have higher weight than ICUs with other roles. In different domains, the importance of expertise may vary drastically depending on the SCUs goal. The Socio-Technical Trust for an SCU that is newly formed is calculated as a weighted aggregate score of the reputation of the ICUs of which it will be formed, because the reputation contains the performance as well as the social trust of ICUs. Thus, this metric can be used in formation algorithms to decide about the most trusted SCU from the possible compositons.

$$STT^f(scu_i) = \sum_{i=1}^{|S|} (w_{expertise}(s_i) * Reputation(s_i)) / \sum_{i=1}^{|S|} w_{expertise}(s_i) \quad (4.6)$$

Table 4.1: Notations

Notation	Description
s_i	ICU, a member of an SCU
scu_i/S	SCU/an SCU as a set of ICUs
r_i	ICU not associated to any specific SCU, a global profile of an ICU in the pool of ICUs
m	Number of invocations of a particular SCU
$mc(s_k, s_i)$	Trust vote from ICU k to ICU i based on their collaboration experience within the same SCU
$MCTS(s_i), MCTS_{gl}$	Local, respectively, global Membership Collaboration Trust Score of an ICU
$STTR$	Socio-Technical Trust score given to ICUs based on the STT of the SCUs in which it has been a member
$PT(s_i)$	Performance/Technical trust of an ICU in SCU
$ST(scu_i)$	Social Trust of a specific SCU
$STT^f(scu_i)$	Socio-Technical Trust metric of an SCU at the time of formation
$STT^e(scu_i)$	Socio-Technical Trust metric of an SCU that has been invoked before
$STT(scu_i)$	Socio-Technical Trust metric of a specific SCU scu_i

In the case when a customer or a software client wants to chose an SCU that has been previously invoked, our model takes into account overall SCU metrics. SCU specific metrics are aggregates of the ICUs that have taken part in it and it is also a function of performance-based trust (Equation 4.7), as well as social trust (Equation 4.8). PT or software-based trust of an SCU, which is the *technical* part of trust in our model, is an aggregate of the performance trust of its member ICUs, considering the total number of SCU invocations m , over a period of time.

$$PT(scu_i) = \frac{1}{m} \sum_{k=1}^m \sum_{i=1}^{|S|} PT(s_i)_k / S \quad (4.7)$$

$$ST(scu_i) = \frac{1}{m} \sum_{k=1}^m (w_{mcts} * MCTS(scu_i)_k + w_{css} * CSS(scu_i)_k). \quad (4.8)$$

The social trust of an SCU, ST, is calculated as a weighted average of MCTS of its members, and the customer satisfaction score of the client (CSS) regarding SCU's

performance and quality of results, as presented in Equation (4.8). Here too, CSS is a subjective trust vote from the client to the SCU. Equation (4.9) defines the Socio-Technical Trust of an SCU that can be used in SCU selection algorithms, as it considers historical data for SCUs, by including both performance trust (Eq.4.8) and social trust (Eq.4.9) over a number of invocations of the SCU, m .

$$STT^e(scu_i) = \frac{1}{2}(PT(scu_i) + ST(scu_i)) \quad (4.9)$$

Figure 4.3 shows the metrics that we have discussed and consequently gives an overview of our Socio-Technical Trust Model of an SCU. The lines with arrowheads are in the direction from more simple metrics, to metrics which are comprised of those simple ones, and show how we derive the STT for SCUs from both the social scores and automatically calculated scores based on SCUs performance monitoring. As QoR metric is domain-dependent we have not included it in our definitions but the model shows that it is easy to add this and any other metric to it.

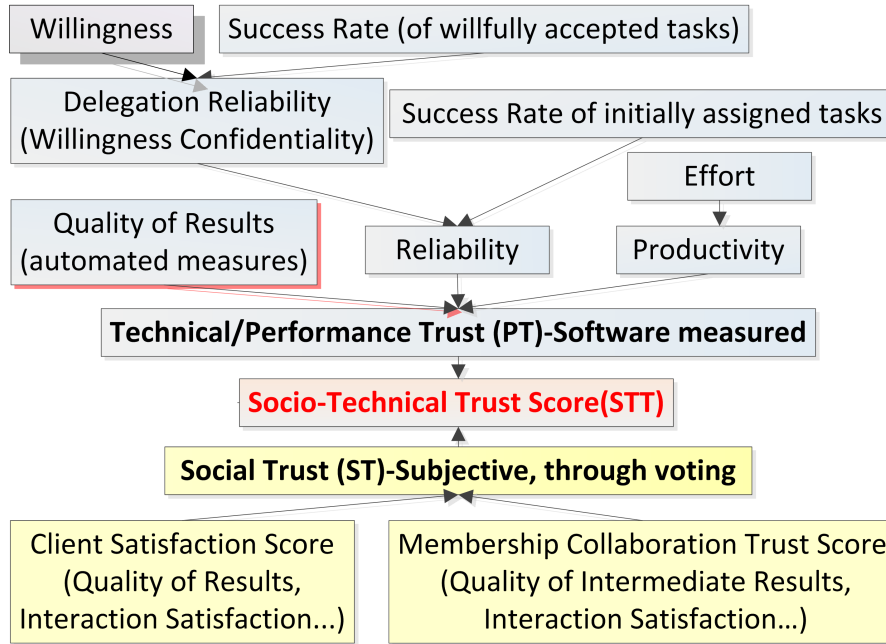


Figure 4.3: SCU Socio-Technical Trust Model

Definition 4 *Trust in a collective in the context of a specific goal is the expectation that at any point in time within its lifecycle, its structure and member-capabilities are appropriate for executing the tasks and will perform within the preset or negotiated constraints.*

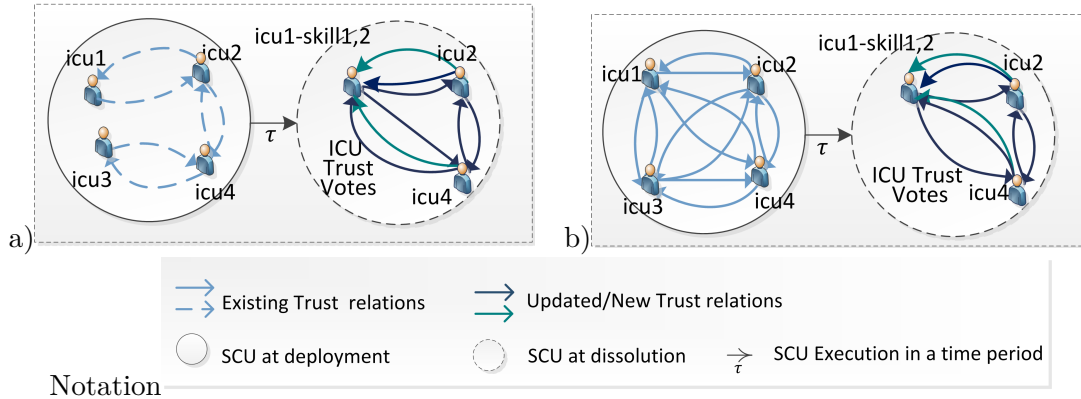


Figure 4.4: SCU trust metrics and relations updates

4.3 Elastic Adaptation Strategies with Trust: Algorithms and Experiments

The objective of an SCU provisioning platform is not only to provision an appropriate SCU for a specific client, but also to maximize performance, which means keeping deadlines, submitting high quality results and maintaining the SCU operation cost within the allowed budget. In our scenario, e.g., we need to ensure that data is always available for the second SCU (Section 4.1.2/B/b), and that a person is always available for immediate response for the third SCU (4.1.2/B/c). This requires for ICU availability at all times for all SCUs. For this, for efficient SCU performance with Algorithm 4.1 we propose an approach for elastic SCU adaptation based on delegations and considering ICU trust (that includes availability and performance metrics). The algorithm monitors each task and in case of a time-threshold at one ICU, it delegates the task to another available ICU. Lines 2-7 check for the reputation score and current execution state of the ICU from which the task needs to be withdrawn and delegated to another. If its trust score is less than 0.5 and the ICU does not have tasks in execution it is removed from the SCU. Lines 9-13 check ICU reputation and cost for all ICUs in the pool of available ICUs, regardless if they are members of the SCU or not. ICUs that have a reputation higher or equal to 0.5 and that fit the allowed cost are ranked in ascending order of their task queues. A request to work on the task that needs to be delegated is then sent to the ranked list of ICUs. The task is finally reassigned to the ICU highest on the ranked list that has sent an acknowledgment for willingness to execute the task. If the ICU is not a member of the SCU, it is added. The SCU trust is updated at runtime during the adaptation of the SCU.

4.3.1 Experiments

We evaluated our trust model via simulations. We designed ICU Profiles with static properties (unaltered values) and dynamic properties (updated at runtime). The static properties are skill type and cost per task, whereas the dynamic properties are: Pro-

Algorithm 4.1: A Cost-Effective Algorithm for Elastic Adaptation of SCUs based on ICU Reputation

Data: icu member of SCU, icu member of P
Data: task assigned in SCU

```
1 forall task in T do
2   if task.inIcuQueueDuration ==
3     task.thresholdDuration then
4     icuCurrent = task.getICU() ;
5     if icuCurrent.icuReputation ≤ 0.5  &&
6       icuCurrent.taskExecuting = 0 then
7       | SCU ← removeICU();
8     end
9     forall icu in P do
10      if icu.STT ≥ 0.5&&
11        (currentCost + icu.Cost) ≤ Budget then
12        | rankingList ← min(icu.taskQueue) ;
13      end
14      sendWillingnessReq();
15    end
16    /* Assign the task on threshold to the highest ranked ICU that
17       acknowledges a Willingness request */
18    forall icu in rankingList do
19      if icu.Ack() == true then
20        | icuCurrent = icu;
21        | if icu is not a member of SCU then
22        | | SCU ← addICU(icuCurrent);
23        | end
24        | Break;
25      end
26    end
27  end
28 end
```

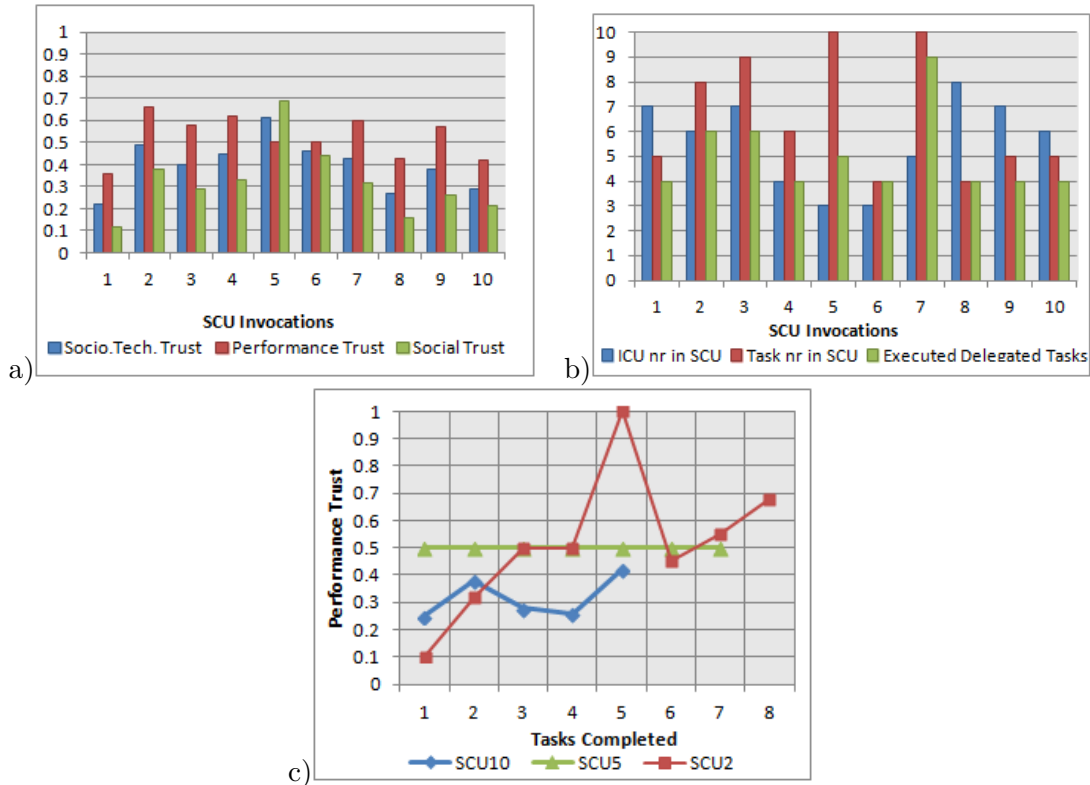


Figure 4.5: SCU metric updates without considering trust for delegations

ductivity, MBCT, ICUReputation, STT and all the atomic metrics discussed in this work, all of which are calculated according to our model. Nevertheless, we note here that we calculate social trust only with membership collaboration trust scores not including the client satisfaction. Tasks are designed with states: ASSIGNED, INEXECUTION, SUCCESS, FAILURE (if in FAILURE state a task is delegated). Tasks are individually executed and they are not interdependent.

Base Algorithm We implemented an algorithm which we take as a base for comparison and which adapts the SCU without considering trust. We assigned tasks randomly and delegated randomly to ICUs. Graph a) in Figure 4.5 shows the STT score, MCTS and Productivity for multiple invocations of an SCU. We calculated trust by assigning different weights to performance and social trust (0.4 for PT and 0.6 for ST). Figure 4.5(c) shows performance trust updates of particular SCU invocations at runtime. If we take the 10th invocation as an example, we can see that we have four delegated tasks out of five in total so the decline in the performance trust (productivity) in Figure 4.5(c) comes at the point when delegations occur. If we look at Figure 4.5(a) and Figure 4.5(b) for SCU7, we notice that the performance trust is higher than the social trust, because here we have a high number of delegations and exclusions of ICUs. This means, that a small number of ICUs executed high number of tasks but the social trust of the SCU is

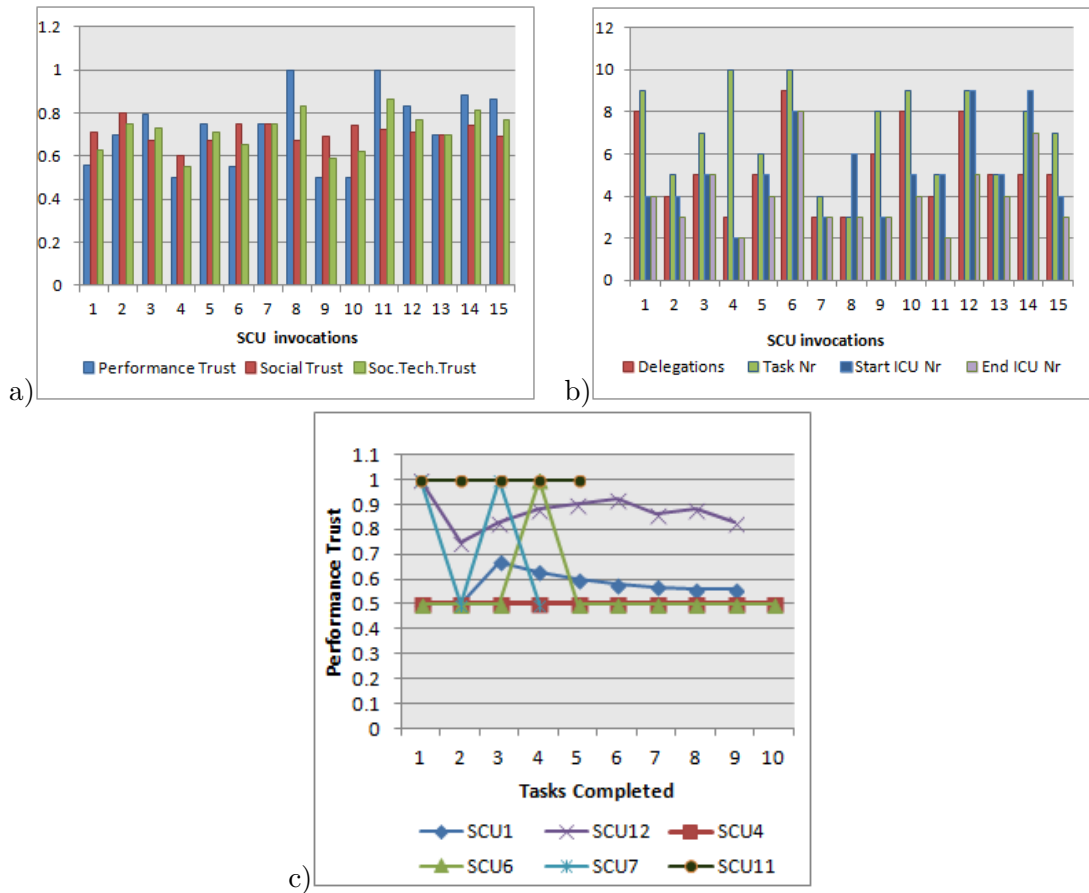


Figure 4.6: SCU metric updates considering the STT trust model low because many ICUs failed to execute tasks.

Adaptation Algorithm considering the STT score In order to evaluate our trust model’s behavior we implemented the aforementioned Algorithm 4.1 with the following settings: we set all the ICUs’ reputation in the SCU to be higher than 0.5 and thus we assigned the delegated tasks only to ICUs that have MBCT higher than 0.5. We set the weight of both the social and performance trust of the SCU to be the same, i.e. 0.5. We set ICUs that had previously failed at all assigned tasks to be excluded from the SCU after they delegate a task, some ICUs can be excluded from SCUs after a delegation even if they have previously completed tasks(their PT is then included in the final trust calculations of the SCU). Analyzing the results of this experiment we see SCU adaptations due to delegations. In some SCU invocations tasks are delegated to SCU members, in some they are delegated to ICUs that were not members and so there were inclusions of new ICUs and exclusions of ICUs. Figure 4.6 shows the end results of each SCU invocation (out of 15), while the graph in Figure 4.6(c) shows performance trust updates of some selected SCUs at runtime. Examining the case of the SCU11 from Figure4.6(a) we noticed that its PT was 1 while ST was lower and the STT was high as well. From Figure 4.6(b) we noticed that the number of ICUs is lower at SCU dissolution

than at start, which means that some ICUs are excluded and thus their productivity is not included in the PT calculation because the excluded ICUs had no successfully completed tasks. Hence, the PT is constant at 1.0. Examining ICU12 we saw that at runtime the number of ICUs also changed with delegations. Some tasks were delegated to ICUs within the SCU, whereas some new ICUs were included for executing a task that needed to be delegated. An ICU with a failed task was not excluded from the SCU and its performance was included in calculating the SCUs PT. Hence the lower points in PT Figure 4.6(c) for SCU12. In all cases, those ICUs that had previously performed well but had failed in a task or two were still included in the STT calculation so the variations of the trust score to lower values is also due to this fact and not only due to the number of failed tasks. For excluded ICUs the MBCT is low, in our implementation this value is excluded from the ST calculation at runtime. Hence the social trust values are relatively high in Figure 4.6(a). Comparing the two algorithms it is clear that the

```

<ICU icuId="6">
  <ICUSkill_>developer</ICUSkill_>
  <ICUCostPerTask_>35.0</ICUCostPerTask_>
  <ICUAvailability_>0.6</ICUAvailability_>
  <ICUsatisfactionScore_>0.63</ICUsatisfactionScore_>
  <ICUMetrics metricsId="6">
    <totalAssignedTasks>14</totalAssignedTasks>
    <totalDelFromTasks>2</totalDelFromTasks>
    <totalDelToTasks>0</totalDelToTasks>
    <successfulTasks>14</successfulTasks>
    <successRate_>0.8571</successRate_>
    <delToSuccessRate_>0.0</delToSuccessRate_>
    <nonDelToSuccRate_>0.8571</nonDelToSuccRate_>
    <reliability>0.8</reliability>
    <MCTS>0.3667</MCTS>
    <technicalTrust_>0.9626</technicalTrust_>
    <socioTechnicalTrust_>0.6050</socioTechnicalTrust_>
  </ICUMetrics>
</ICU>

```

Figure 4.7: Illustrative xml file with ICU metrics

algorithm implemented with our STT model keeps a high-value trust of an SCU during runtime as well as with time brings a steady performance of an SCU as compared to a strategy that adapts the SCU without considering trust (both social and technical).

As much as experimenting with real data sets would give more accurate results it is very difficult to conduct the experiments due to lack of available data sets that fit the SCU construct. We designed the ICU profiles the best we could, to fit the purpose of our algorithms. The experiment also reflects the advantage of using our model in adaptation strategies for SCUs, because based on our model the SCU can be tuned in terms of structure and non-functional parameters to keep a certain level of desired performance.

Lastly, for illustration, Figure 4.7 shows one ICU, and some of its metrics data in an xml format generated from one run of an SCU with an assignment of one batch-of-

tasks. Appendix B shows selected results from running Algorithm 4.1, where results are regarding an SCU in separate time points together with information about members, member metric values as well as collective metrics.

4.4 Incentive Mechanisms with Trust

Social Capital as a collective gain is seen as one of the main drivers for people to cooperate [WF05] and perform well in collaborations. We were motivated by this concept to model $Reputation(s_i)$ but most importantly to use it as an incentive method to control the behavior of SCU members for performing efficiently and avoid misbehavior. In calculating the reputation of an ICU we include the Socio-Technical Trust scores of each SCU that it has been a member of. Equation (4.5) and Algorithm 4.2 describe the reputation update that is our incentive mechanism. With this mechanism, ICUs become aware that the total trust score of the SCU in which they are engaged, will affect their reputation and so this may influence their motivation to work and the SCU provisioning platform can avoid or lessen misbehavior within the SCU. This incentive mechanism is enforced with designing the ICU/SCU supporting platform in a way that gives ICUs the knowledge that their performance data and the Membership Collaboration Trust Score are included in the SCU STT score, and in turn the SCU STT score is used to update their reputation.

Lines 2-7 in Algorithm 4.2 check if all ICUs have finished their tasks and ask them to give a membership collaboration trust score to every other ICU in the SCU. Lines 13-16 calculate ICU and SCU STT scores, according to the presented model.

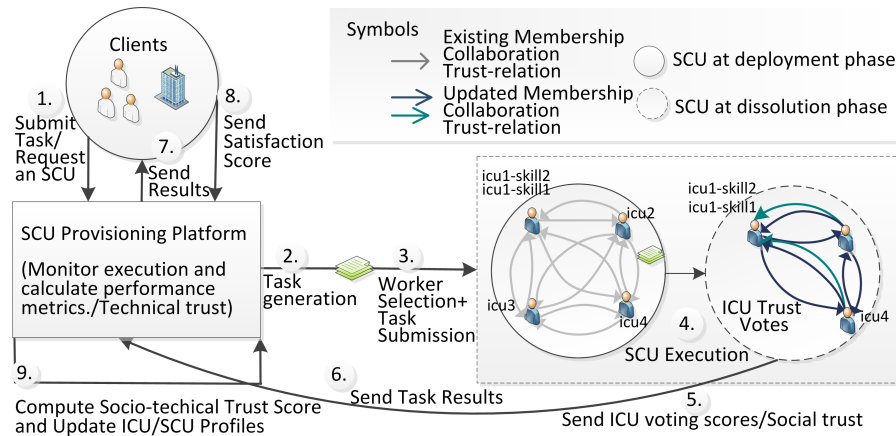


Figure 4.8: SCU operation and ICU trust updates

In [SSD10b] the authors argue that people have low incentives to manually assign ratings. To overcome this challenge, our strategy enforces Membership Collaboration Trust score voting as a condition for an ICU so that its work can be accepted and a payment can be made before the SCU is dissolved. See Figure 4.8. Thus, lines 17-21

update ICUs' Reputation, pay the ICUs *after they have voted*, and the SCU dissolves. On the other hand, the incentive mechanism obviously brings the challenge of keeping ICUs from misbehaving while voting, because the Socio-Technical Trust score of the SCU in our model is higher the higher the trust score between SCU members is, thus ICUs can be tempted to give a higher score to collaborators ($mc(s_k, s_i)$) to increase the SCU trust so that their reputation is also increased. This might lead to unfairness, because ICUs can vote high for others only for their own gain.

To overcome the issue on unfair ratings, based on the voting ICU's current reputation, we assign it an allowed trust range for voting (line 5 of Algorithm 4.2). This range can vary according to the range values that a user will impose on the model. For example, if the voting ICU's (s_k) reputation is lower than 0.5, we chose the allowed voting ranges for the voted ICU's (s_i) as follows:

$$mc(sk, si) = \begin{cases} [0.5, 1] & \text{if } Reputation(si) \geq 0.5; \\ (0, 0.5] & \text{if } Reputation(si) < 0.5. \end{cases} \quad (4.10)$$

Hence, the person who votes a collaborator can assign votes to that collaborator only from an allowed range of votes, and this is defined depending on his/her own reputation.

4.5 Related Work

4.5.1 Trust in Social Computing

Different context-based platforms such as those enabling pure social networking and those enabling networking in terms of expertise are an important part for provisioning human capabilities. Thus, there is a solid amount of work that investigates trust for human computation from the pure crowdsourcing perspective, modeling individual worker trust, as well as work that mix crowdsourcing with context-based networking, such as those in expert networks. Golbeck in [Gol06], describes the TidalTrust algorithm for computing trust in online social networks, which is based on the breadth-first search. Urbano models Social Computational Trust in [Urb13]. Authors in [GW11] discuss trust in network of people and computers, focusing on behavioral trust and computational trust, where behavioral trust is characterized by peoples' trust in others (defined by multiple dimensions such as beliefs and preferences), while computational trust is one that can be computationally measured, as for example reputation score. Skopik et. al, in [SSD10b] present a framework to analyze trust in mixed service oriented networks, metrics and rules for inferring trust. They solve the issue of overloaded workers with an approach on interaction balancing through delegations to trusted actors. We believe that our approach, further strengthens trust models that consider delegation mechanisms as our willingness confidence metric can be used to strengthen trust metrics when deciding about task delegations.

Most trust models concerning social computing omit the aspect of people collaborating for executing complex tasks and their semi-automatic elastic management. Rather, most of

Algorithm 4.2: Membership-Collaboration Trust Update Algorithm as an Incentive Mechanism

```
Data: icu worker in SCU
1 forall icu in SCU do
2   if icu.getStatus() == FINISHED_TASKS then
3     icuCurrent = icu.geticuId()
4     forall icu in SCU && icuId != icuCurrent do
5       getTrustRange(icu)
6       votes ← mc(icuCurrent, icu)
7       icuCurrent.UpdateMCTS(votes)
8     end
9   end
10  icuMCTSList ← getMCTS(icuCurrent)
11  icuPTList ← getPT(icuCurrent)
12 end
    // calculate SCU Socio-Technical Trust
13 scuMCTS = calculateSCUMCTS()
14 scuCSS = getClientSatisfaction()
15 scuPT = calculateSCUPT()
16 scuSTT = calculateSCUSTT()
17 sttr = calculateSTTR(scuSTT)
18 forall icu in SCU do
    // calculate the reputation of ICUs by adding the SST score derived
    // from the current SCU, to the existing global Socio-Technical
    // Trust score
19 icu.Reputation = UpdateReputation(icu, sttr)
20 icu.Payment = getICUPayment()
21 SCU ← SCU \ ICU /* dissolve SCU */
22 end
```

the existing work focuses on initial collaborator/partner selection and more often analyze trust on the individual, which include for example the exchange of eCommerce goods in electronic marketplaces [ZMM00], user trust in mobile networks [LZZ10], or simple non-complex task executions such as those similar to HITs in Amazon Mechanical Turk. Some work have modeled trust in group collaborations, such as [CAK13]. The difference of our work is that it concerns complex socio-technical systems, where collaborations are automatically managed with input from a human-in-the-loop.

Authors in [LCCM09] present a trust model for a specific collaboration scenario that includes document creation and development. They introduce a trust score that is computed based on the amount of words added and deleted to a word document, including different roles for people working on the document creation, writers, reviewers and one validator. This approach is focused on automated trust computation but it does

not take into account the social aspect of trust where collaborators can decide whether to read or not changes in the document based on historic behavior of their collaborators and subjective impressions for example. Moreover, the utilization of trust scores during collaboration will help to raise its effectiveness. However, the authors do mention that these are important open questions for their future work. Our model, includes social metrics that are computed from input by humans (clients, collaborators) in addition to metrics that can be automatically monitored. A survey of trust in multiple areas such as computer science, economy, and sociology is presented by Caton et al. in [CDG⁺12]. As we also showed a way to include trust scores for incentivizing workers by adding a score to their overall individual trust-score based on the overall trust-score of the collective after the work is finished, the work in [Kea12] is worth mentioning, as in the description of a variety of social computing experiments the authors mention one where people were incentivized to bring value at the collective level, by being paid for their activities that would result in a good collective result rather than good individual results.

4.5.2 Agent-based Trust

Multidimensional trust inference for selecting a partner for collaboration is described in [KG08]. The mechanism of trust and reputation is based on multiple sources, such as from direct interactions, and trusted recommendations from direct and indirect communication that they name as witness reputation. [ARH00] also considers indirect trust, where one agent is trusted to give recommendations about others within a specific context. A survey of various trust and reputation models and their possible use in various agent-based distributed system models is presented in [KG10].

Examples of research reports on utilizing fuzzy approaches in trust investigations are [RMM03], [Gri06].

Team Formation

One of the existing challenges that still requires attention in human computation is forming the most efficient team/s for a particular need. In this chapter we investigate this problem. Existing works have focused on team formation strategies, mostly based on the appropriateness of skills and team connectivity based on existing interaction analysis between possible team participants. They provide solutions to team-formation in terms of minimizing coordination cost, team size, and workload. Authors in [DSSD11] for example consider that the frequency of previous interactions are one of the indicators for efficient teams. As far as we know, existing work has focused on efficiency metrics, but only on one type of connections between team members, coordination cost between team members, general interaction frequency or both, while we chose to look at interaction in more details, specifically focusing on *interaction types*. We consider two types of interaction networks, communication interactions which include only natural language communication between people, and coordination interaction, which include task-related interactions, such as a delegation of one task from one person to another. To generalize, our hypothesis is that if we are to assemble efficient collectives, a group of individuals who have the required skills for the specific goal can not be formed without considering multiple underlying (possible) relation types between them. Our contributions in this chapter are:

- a novel approach to team formation considering *interaction types*,
- a team-formation algorithm by ranking experts based on trust and weighted interaction values,
- a team formation algorithm based on data of previously formed teams that have already completed projects.

5.1 Team Formation based on trust and multiple interaction types

5.1.1 Problem Statement

The specific team-formation problems that we investigate in this chapter can be defined as in the following definitions.

Problem 1 *Given a network of experts with information about their previous performance and interactions, and a given project-requirement, find a collective/team of experts from the network who can execute the project effectively.*

Problem 2 *Given a set of existing teams with information about their previous performance and interactions, and a given project-requirement, find and select a collective/team of experts who can execute the project effectively.*

5.1.2 Model

To approach the stated problems we consider two types of information crucial for our team-formation model and strategies:

1. **interaction links**, and
2. **data regarding expert performance**, ie., votings and recommendations, acceptance or rejection of tasks assigned from the system etc.

Regarding the interaction links, we are distinguishing two types of them in the context of a specific collaboration:

1. **communication links**, which include messages in natural language; and
2. **coordination links**, which include task-related interactions such as delegation of tasks (control flow with human in the loop).

Regarding the performance-data of experts, we consider the STT trust metric for experts and teams over a specific period of time presented in Chapter 4.

Specifying our expert environment model now, we denote a pool of experts as $P = \{p_1, p_2, p_3 \dots p_n\}$, a team of people (a collective) as $C \subseteq P$ and a set of initial tasks for a specific team C as $T = \{t_1, t_2, t_3 \dots p_n\}$. Every expert in the pool P has her/his own profile properties. The important ones for us in this work are $s(p)$ denoting the skill type of p , $stt(p)$, denoting the trust score of p , and $c(p)$ denoting the labor cost of expert p for skill $s(p)$. As aforementioned, we consider two interaction network types: a) a communication network, in which the interaction intensity between two people is denoted by weight $w_m(p_i, p_j)$, and b) a coordination network, in which the intensity

of interactions in terms of coordination is denoted by weight $w_c(p_i, p_j)$. The shortest path between two people in the network is denoted by $d(p_i, p_j)$. The edges within a team are denoted with the set E , and the total number of edges is denoted with $|E|$. A communication edge between two team-members is denoted with e_m , while a coordination one with e_c . The total number of edges between p_i and p_j is denoted with $e_{m,c}$. Thus, $w_m(p_i, p_j) = \frac{\sum e_m}{e_{m,c}}$, and $w_c(p_i, p_j) = \frac{\sum e_c}{e_{m,c}}$. A pair of two team members within a team between who exists at least one interaction link is denoted with $(p_i, p_j)_e$. We denote the collection of teams, in which a specific person has been a member (over a specific time period), with $\mathfrak{U}^\tau = \{C_1^\tau, C_2^\tau \dots C_n^\tau\}$.

The requirements for team-members (for each particular task) are denoted with $I(s, r, c)$ individually, where s is the skill-type, r is the reputation of a member of a team in the context of a particular skill, and c is labor cost. The initial network consists of multiple experts who have previously worked together in teams and consequently created different types of interaction links. From this network we distinguish two types of interactions, one based on *communication* and another based on *coordination* interactions.

Because we consider trust scores of experts, we hypothesize that given a high STT, a high number of interactions between experts in the context of message exchanges and/or task-related coordination is an indicator that those experts can work well with each other. Thus, in our model we aim to form teams with a high value for the weighted values of these type of interactions. We denote the total weight of communication interactions within a specific team, with $W_m(C)$, whereas the total weight of interactions in terms of task-coordination as $W_c(C)$.

As aforementioned, in our strategies we use the previously introduced STT metric, which we use in our human-based resource model in this chapter. Thus, each node in our model is defined with a trust score. The team trust score is a normalized aggregated trust score from the member expert trust values. The threshold value for the team STT is denoted with δ . Now we have mentioned all the indicators over which we want to execute a team-formation algorithm (notations are given in Table 5.1). Consequently, we need to form a team that:

- includes experts with matching skill requirements
- satisfies $STT \geq \delta$
- minimizes the team diameter, $D(C)$
- maximizes the weight of communication interactions between team-members, $W_m(C)$
- maximizes the weight of coordination interactions between team-members, $W_c(C)$.

The team-weight of communication interactions is defined as:

$$W_m(C) = \frac{1}{\sum_{(p_i, p_j)_e} \sum_{i \neq j} w_m(p_i, p_j)} \quad \forall p_i, p_j \in C \quad (5.1)$$

The team-weight of coordination interactions is defined as:

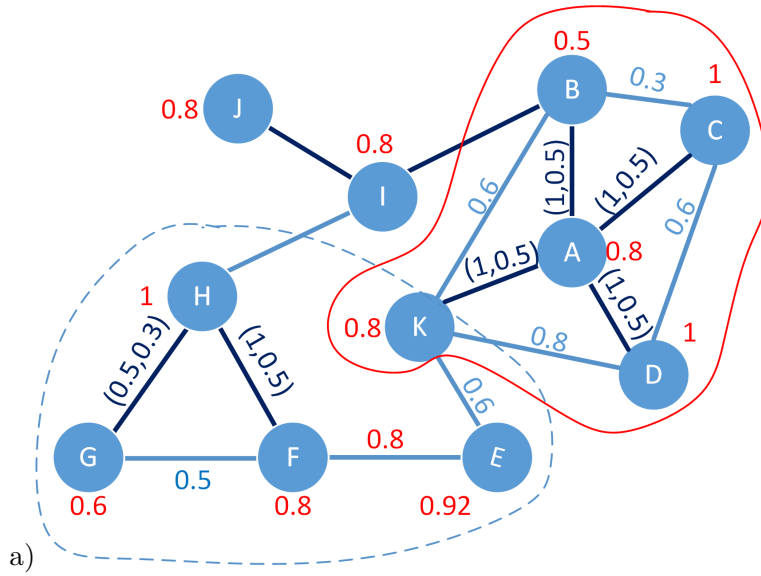
$$W_c(C) = \frac{1}{\sum_{(p_i, p_j)_e} 1} \sum_{i \neq j} w_c(p_i, p_j) \quad \forall p_i, p_j \in C \quad (5.2)$$

We approach the problem of team formation in two ways: a) by using an Analytic Hierarchy Process (AHP) method in ranking the importance of the above constraints for experts individually, and using a non-evolutionary Pareto-based approach to rank formed teams and select the most appropriate one, and b) by using a Genetic Algorithm approach by assuming we have team-based historical information and creating new efficient teams out of existing teams and their data. We compare both approaches in the experiment section. The team requirements and consequently, indicators of efficiency, in our model are: $C(s)$, STT , $D(C)$, $W_m(C)$, and $W_c(C)$.

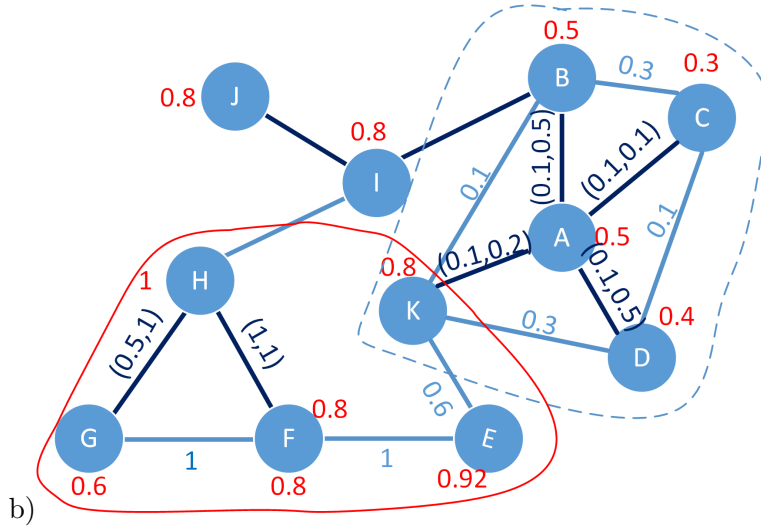
Table 5.1: Notations

Notation	Description
P	Pool of people/experts
C_i	Team $C_i \subset P$
e_m	a communication interaction-type link between two experts
e_c	a coordination interaction-type link between two experts
e_m, c	total communication and coordination interaction links between two experts
$(p_i, p_j)_e$	a pair of team members with at least one interaction link
$w_m(p_i, p_j)$	communication link strength between expert p_i and expert p_j
$w_c(p_i, p_j)$	coordination link strength between expert p_i and expert p_j
STT	Socio-technical trust score of a team
$W_m(C)$	Normalized weight of communication interactions (message-exchange) between team-members
$W_c(C)$	Normalized weight of coordination interactions between team-members
$D(C)$	Team diameter

Let us look at an example to further motivate our problem and justify the reason behind our argument that interaction types provide valuable information regarding team collaborations and efficiency. Figure 5.1 shows two same sections of an expert network with two teams on each of them. The lighter links represent communication interactions, and the ones in dark blue represent two types of interactions existing between team members, communication and coordination, and edge values are given respectively as well for illustration purposes. In sub-figure a) the team formed by A,B,C,D,K is more favorable than the one formed by K,E,F,G,H. Both teams have the same number of nodes and close values for the total team socio-technical trust scores (if we have an average value of all node STT-values), which are presented by the values in red. However, in the same sub-figure the favorable team is better connected and has more connections of both



a) Favorable and non-favorable teams in the case when a better connected team has favorable interaction weights. The bordered area with a full line represents a favorable team, whereas the bordered area with a dashed line represents a non-favorable one.



b) Favorable and non-favorable teams in the case when a better connected team has less favorable interaction weights. The bordered area with a full line represents a favorable team, whereas the bordered area with a dashed line represents a non-favorable one.

Figure 5.1: The lighter lines represent interactions only in terms of communication, the darker lines represent the presence of two types of interactions: communication and coordination. Edge values represent interaction weights, a single value represents communication weight, while sets of two values represent weights of communication and coordination, respectively. Values in red represent STT scores of experts.

interaction types between team members. Thus, this case is fairly intuitive for our case. However, looking at sub-figure b), we see that the more favorable team is the one that includes K,E,F,G,H. The STT score does not have a high discrepancy, but the values for the edge weights for both interaction types are much higher than the ones for the team with nodes A,B,C,D,K. This means that a ranking algorithm or a team-formation algorithm should take into consideration the case when a team that is "less connected" has more communication and coordination interactions between connected members, because this might be an indicator of effective collaborations and higher trust between team members. Hence, it is these type of specific cases that bring us to consider the type of interaction links, and the distance between team members/diameter in a multi-objective team-formation strategy that we present in Algorithm 5.1.

5.1.3 Expert role connected to different types of interaction links-Discussion

One reason of separating communication interactions and coordination interactions is because these two types of interactions separately can inform about the member roles. More specifically, one application of the analysis of these two interaction types is the differentiation between a leader of the team or unavailability of a team member. For example, if a member of a team has a high trust score and it has more communication interactions in terms of message-exchanges in natural language and more coordination interaction, then these three metrics can indicate that the member might be the leader/coordinator in the team, as it communicated but also manages the control flow, e.g., delegated tasks to other members of the team. If on the other hand, a member of the team does not have a high number of communication messages and has high coordination-based interactions then it can mean that the member has been unavailable because it has not communicated and it had rejected tasks or delegated tasks to co-members. However, these are only assumptions, and the investigation of the problem of role-identification based on interaction types is an open question.

5.2 Programming team formation

SCUs can be formed from scratch by selecting appropriate ICUs, or they can be engaged in task-execution/project through a selection process based on historical data of existing SCUs. We have come up with two algorithms for both cases and in the following we describe their implementation and the experiments we conducted. With Algorithm 5.1 we present a strategy to form teams out of a ranked list of available experts, while Algorithm 5.2 presents a genetic algorithm method to form teams out of an existing initial set of teams.

Algorithm 5.1 ranks all experts by *reputation* separating them by skill. Because the reputation metric is a composite one, we use an AHP based algorithm to rank experts as AHP provides a hierarchical weighting method for parameters that are comprised of several others. Next, a team is formed in such a way that based on skill types an

Algorithm 5.1: Team-formation algorithm utilizing AHP for ranking experts and non-evolutionary Pareto based team selection

```

Data: Graph  $G(P,I)$ ,  $T$ ,  $I(s,r,c)$ ,  $C_r$ 
1  $U = \emptyset$  forall task in T with  $I(s_k, r, c)$  do
2   rankWithAHP(P) /* store ranked experts in separate lists by skill type */
   listP( $s_k$ )  $\leftarrow$  ranked experts with skill  $s_k$  /* store ranked experts in
   separate lists by skill type */
3   forall listP( $s_k$ ) do
4     create x number of teams by choosing experts from the same rank, from
     each list with ranked experts based on skill, in descending order ;
5      $C_i \leftarrow listP(s_k).getFirstAListElement()$   $U \leftarrow C_i$ 
6   end
7   /* Rank all teams within U compared according to objectives with weights:
    $minD(C) \in CO, maxW_m(C) \in CO, maxW_c(C) \in CO$  */ Collections.sort(U,
   ParetoComparator( $C_i, C_j, o$ ));
8 end
9 return U

```

available expert with the highest reputation score for each skill type is included within the team. After forming multiple such teams in descending order from the reputation-based ranked lists the algorithm compares teams based on communication and coordination interaction weights, and team diameter, and ranks the teams maximizing communication and coordination interactions, and minimizing the team diameter. Algorithm 5.1 shown in this paper is a high-level pseudo-code showing the basic steps for clarity, as algorithms and methods such as the *AHP ranking* of experts and the *ParetoComparator* are implemented in different classes; thus details are not shown for clarity of the overall algorithm.

Algorithm 5.2 is a genetic algorithm, which takes as input a number of appropriate teams for the tasks. Instead of ranking existing teams we generate new configurations from them to get more effective teams but also to avoid being forced to use only solutions from a set of "local optimum" ones, if other more effective solutions are possible. In our algorithm, a population is represented by a set of teams, each team having the same number of team-members. The set of genes considered for each team are the values for $W_m(C)$, $W_c(C)$ and $D(C)$. For the fitness function, we set the following requirements: $STT \geq 0.5$, $((W_m(C) + W_c(C))/2) \geq 0.5$, and $D(C) \leq 0.5$. The algorithm gives a sorted list of teams based on a comparison function of our communication, coordination, and team diameter objectives.

Algorithm 5.2: Team-formation Genetic Algorithm

```
Data: PC
1 population ← PC.size(), newPopulation = ∅ ;
2 forall teams in PC do
3   | team ← setGenes(Wm(C), Wc(C), D(C)) population.add(team)
4 end
5 while generationCount ≤ maxSteps do
6   | forall team in PC do
7     | calculateFitness();
8   | end
9   | selectParentsByRouletteWheel();
10  | newChildrenCrossover();
11  | if mutateVar ≤ mutatePercent then
12    | newChildrenMutation();
13  | end
14  | calculate fitness for children;
15  | calculateFitness();
16  | newPopulation ← newPopulation.add(child) population ← newPopulation
    | sort(population)
17 end
```

5.3 Experiments

5.3.1 Evaluation with synthetic data

We implemented Algorithm 5.1 for team formation and selection, combining AHP for team formation and Pareto-based efficiency for selecting the most appropriate team, based on pre-set requirements. Thus, we used an a priory decision making approach, considering a scenario where teams are not formed ad-hoc but with a customer request. We used the AHP method for ranking people based on skill-types and reputation ¹. We modeled and generated a pool of 400 human profiles with skills and different metric values, such as values for cost per task, and a socio-technical trust score. Each person has a single skill. We modeled and generated tasks, where each task was associated with a single skill. Every person was modeled to have a connection with a (random) number of other experts from the pool of resources indicating a previous collaboration. Every connection/edge had two weighted values, one indicating a communication interaction and the other indicating a coordination interaction.

Algorithm 5.1 has two blocks, the first one forms and ranks teams based on AHP analysis of two requirements for teams: cost per task and global socio-technical trust score (reputation) of team members (lines 1-9); the second block (lines 10-15) ranks teams based on Pareto analysis of three pre-set objectives: higher $W_m(C)$ and $W_c(C)$, and lower $D(C)$. We set the requirement for the STT of the team as $STT \geq 0.5$ in the

¹We based our model on the framework presented by authors in [CFMT09].

team formation with AHP. We generated 10 initial tasks, with 6 tasks having the same skill and 4 tasks having different skills, so as to simulate for example realistic teams such as development, where development skills are represented more often, while testing, and design skills for example are represented with fewer people. Thus, the size of each team is the same and does not influence the formation, ranking and selection process. The teams were formed such that we ranked and matched people to the requirements for each incoming task, by skill type, reputation and cost per task. For one run of the algorithm 30 teams were created with 30 rounds of 10 task assignments. We ran a Pareto comparison and ranking method on the 30 teams and ranked them based on the three interaction-based objectives, namely the normalized, communication weights, coordination weights, as well as the diameter weight of the team. Table 5.2 gives the results of the 10 most appropriate teams returned by a run of Algorithm 5.1.

Analyzing the results we could observe that for example the most appropriate team with our algorithm does not necessarily have the highest trust score but it has a fairly high trust score and enough high scores for the communication and coordination values and low enough value for the diameter value. Looking at Table 5.2, we notice that team with Id 21 has better scores when considering the three objectives, although it might have worse scores when considering single objectives one by one, compared to individual objectives of various other teams. Let us check two other interesting examples, the teams with Id 6 and 22 for example. Team 22 is better with regard to $W_m(C)$ and $W_c(C)$ when considered together, but on the other hand is worse regarding $D(C)$ than the team with Id 6, that is why it is ranked much lower. Thus, selecting the highest ranked team returned from Algorithm 5.1 seems a valid option, considering every pre-set requirement. Perhaps it is not the best solution considering single requirements, as for example team with Id 22 might be better in terms of trust and cost (not shown in the results for the sake of clarity) but not better than team with Id 21 in terms of the overall requirements considered in combination.

Table 5.2: Ranked teams and values of three objectives

Team ID	Trust	$W_m(C)$	$W_c(C)$	$D(C)$
21	0.8	1	0.6	0.4
6	0.64	0.8	0.6	0.3
23	0.75	0.6	0.6	0.1
19	0.52	0.5	0.2	0.2
8	0.65	0.5	0.4	0.3
27	0.6	0.3	1	0.5
22	0.82	0.5	0.8	0.8
3	0.65	0.4	0.2	0.6
17	0.62	0.2	0.6	0.8
7	0.56	0.1	0.5	0.5

Algorithm 5.2 takes as an input 10 team configurations ranked according to higher

communication and coordination weights and lower team diameter. Table 5.3 shows the teams at the beginning of the algorithm, and the values for our three objectives, while Table 5.4 shows the last returned teams as they were changed/generated in each run for new generation of teams with the algorithm. The results show that up to the point of the pre-set number for new team generations, the communication weight got to the value of 1 for all generated teams, while the communication weight value did so for only a few teams, this is due to our configurations and the pre-set values for team members regarding these values.

From the perspective of the comparison of both algorithms, we can conclude that in the case we want to form new teams from existing ones, which have been invoked with similar project requirements and with the same number of team members, a genetic algorithm approach that considers both high trusted teams, high communication and coordination interaction weights, and low team distance measure returns better teams than an algorithm such as the one presented in Algorithm 5.1, which forms teams from a pool of experts not considering previous individual membership in teams. However, Algorithm 5.1 returns efficient enough teams when considering individuals from a large pool instead of individuals from already existing teams. Needless to say, Algorithm 5.2 can be run on existing team-network structures in the case that such logs exist, and only on cases where team-structures include all the skill types required.

Table 5.3: Ranked teams and values of three objectives as input for Algorithm 5.2

Team ID	$W_m(C)$	$W_c(C)$	$D(C)$
12	1	0.4	0.1
13	0.8	0.8	0.5
2	1	0.5	0.5
28	0.8	0.8	0.6
20	0.2	0.5	0.2
27	0.6	0.4	0.8
14	0.1	0.4	0.2
17	0.1	0.2	0.8
25	0.3	0.2	0.1
6	0.3	0.2	0.2

5.3.2 Evaluation with real data

In addition to our synthetically generated data-set we investigated how Algorithm 5.1 behaved with a real data-set for comparison. We utilized a data-set created by authors in [PSO⁺16]², which provides real and anonymous data for the activities of software engineering student teams for a final project in a software engineering course. The

²We found the data-set from: <http://archive.ics.uci.edu/ml/datasets/Data+for+Software+Engineering+Teamwork+Assessment+in+Education+Setting>.

Table 5.4: Ranked teams and values of three objectives at a final run of team generations in Algorithm 5.2

Team ID	$W_m(C)$	$W_c(C)$	$D(C)$
4	1	1	0.2
2	1	1	0.2
8	1	1	0.4
14	1	0.8	0.1
5	1	0.6	0.1
24	1	0.5	0.4
27	1	0.5	0.1
11	1	0.5	0.3
3	1	0.2	0.4
1	1	0.1	0.1

data-set provides a variety of data regarding 74 teams working on projects with the same requirements, collected during several semesters at the San Francisco State University. Each team within the data-set is assigned two different grades, one for the development process, and another for the final software built. The grades assigned are A, and F, A representing good results or above expectations and F representing below expectation teams.

The difference between our generated data and the real data-set is that with the synthetic data we generated expert profiles and formed teams according to their rank in the context of their reputation score, and then ranked the teams optimizing three objectives, while the real data-set does not provide information regarding team member skills and competencies, rather it provides data on a team-level. Thus, we evaluate the ranking part of Algorithm 5.1 based on the three objectives only: communication score for the team, coordination score and distance between team members. Moreover, we did not form teams by ranking experts but ranked teams from the data-set by mapping appropriate team related data that fitted our model.

Mapping the data from the data-set, we selected the following team indicators that fitted most to our objectives: *meetingHours*, from the data-set gives the number of team meeting hours which we associated with *communication weight* value in our model to indicate the communication intensity of teams, and *teamMemberResponseCount* and *leadAdminHoursResponseCount*, which are self-reporting team-member and team-lead reports collected multiple times during the development process, which we associated with our *coordination weight* value in our model. The sum of *teamMemberResponseCount* and *leadAdminHoursResponseCount* for each team represents a teams coordination weight value.

The teams in the data-set are of two types, local (from the same university) and global (composed of members from multiple universities). Thus, mapping this data to our model, we assign $D(C) = 1$ to the local teams, and $D(C) = 0.5$ to global teams, with the assumption that members in the local teams know each other better than those in

the global teams, and thus we assign a hard-coded distance weight value to each team based on this assumption. Table 5.5 shows the returned list of the first ranked 13 teams. After examining the ranked teams we noticed that most of the teams had A score for both the development process and the product delivered, but some had A either for the development process or the product delivered. However, the global teams which were formed from various universities had less meetings than those that could meet online, but some of these global teams got grade A even with lower communication interactions. If we take the grade A as a trust indicator, the results show that communication and coordination interactions should be considered together with a trust score for team formation algorithms to be effective. Consequently, trust plays an important role in two contexts when considering communication, coordination and network distance. On one side it can be used as an additional indicator to clarify cases where the communication link number is low, because if the trust between two experts is high, low communication does not mean bad communication. On the other hand, trust can be an indicator of communication link type such that if we want to denote communication links with positive and negative signs denoting positive and negative communication between two experts then understandably, a trust score can be used in decision-making scenarios for the sign of communication links. We leave this problem for our future research.

Table 5.5: Ranked teams and values of three objectives from a real-world data-set

Team ID	$W_m(C)$	$W_c(C)$	$D(C)$	Type
2	176.57	49.0	0.5	local
4	145.57	49.0	0.5	local
1	102.92	48.0	0.5	local
5	219.0	46.0	0.5	local
0	94.95	41.0	0.5	local
3	78.36	43.0	0.5	local
6	56.32	41.0	0.5	local
8	138.86	38.0	0.5	local
12	139.86	34.0	1.0	global
9	54.29	31.0	0.5	local
7	52.86	23.00.1	1.0	global
10	47.71	35.0	1.0	global
11	52.43	27.0	1.0	global

5.4 Related Work

The authors in [DSSD11] have presented two heuristics based on genetic algorithms and simulated annealing for team-formations that consider skills and connectivity of teams members. They also present and discuss a recommendation model for adding new members to the team to fulfill skill requirements. Interactions are also considered in [AKZ13] where authors present multiple strategies for team-assembly focusing on

multiple aspects of cost, they present team-assembly strategies considering the cost of communication, a strategy to find a team based on the cost of team-members, as well as finding Pareto-optimal teams considering both the communication cost and the team-member cost. Lappas et al. in [LLT09] also present Pareto-optimal team formation algorithms with minimized communication cost. Anagnostopoulos et al. in [ABC⁺10] discuss forming teams considering the trade-offs between team-size and load with the help of a greedy task-assignment algorithm, while in [ABC⁺12] they provide team-formation algorithms that consider coordination cost and workload balancing. Examples of work that in general consider the underlying social network from which teams are formed are [LSL15], [CDS13]. Contractor in [Con13] also argues that the network structures play an import role in team effectiveness.

If we allow for elastic teams once the team is formed we can adapt the teams at run-time like some approaches in [RTD15], and avoid considering the trade-off between team-size and cost at the time of the formation of the team as is the concern of authors in [KAK16].

The authors in [Gd05] discuss team formation as well as network adaptations based on two different approaches, namely, structure-based and performance-based strategies. Authors in [KA11] present algorithms for team formation based on skill and coordination cost in two different situations, when a team does not have a leader and when a team has a leader responsible for the team coordination, and run experiments on the DBLP dataset, concluding that the algorithms can form small teams with a high number of common publications and high expertise. An AHP-based approach of a multi-objective optimization is presented in [MCR12].

[Par08] presents an overview of distributed intelligence where human teams are mentioned as entities in distributed intelligence systems. Another mentioned fact is that organizational and social paradigms are a starting point for designing agent systems that can cooperate and collaborate for a common complex objective. Hence, we postulate that learning the formation of efficient human-based collectives could be valuable for agent and mixed-system resource-assembly problems. Distributed intelligence including human computation in various application areas are discussed in [Hey13]. The authors present the term 'global brain' and elaborate on the benefits of distributed intelligence, such as for example disaster prevention and relief, research, innovation, traffic management and others. Considering the team formation problem in this work, the benefits of it can be seen in many areas where distributed expert teams need to be formed for a specific objective. The importance of social interactions in human computation is emphasized in [LP13], where the authors make parallels between neural networks and human computation ones.

The Application of Service Level Agreements for Social Collectives

Collective Adaptive Systems include heterogeneous type of resources and services that interoperate 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*. 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 (and offline), are mainly managed by standard (human-language) contracts. Nevertheless, client requirements and consequently people performance should be monitored in an automated way, so that task management can be efficient and customer needs can be fulfilled. Thus, SLAs for social computing is the challenge that we tackle in this paper. To approach it, we utilize the terms Social Compute Units (SCUs) for social collectives and Individual Compute Units (ICUs) for individual workers, in line with the previous chapters.

For SCUs, a Service Level Agreement (SLA) may play a crucial role in managing SCU execution and controlling the quality of the offered services and thus the returned results. However, as far as we know, Service Level Agreements in human computation [QB11] in general are not explicitly investigated. Although, there is some work that mentions SLAs in crowdsourcing, virtual teams and human-enhanced service oriented environments(e.g., [KPSD11]) that we discuss in the related work, but no specific Service Level Objectives (SLOs) investigation based on metrics relevant for human-based collectives. Having the possibility to describe and define the terms and conditions under which platforms will provision and manage their human-based services is crucial, because it is one of the conditions for enabling human work to actually and properly be utilized as a service.

In Chapters 3 and 4 we described a conceptual architecture of a platform for SCU

provisioning with elastic capabilities and mechanisms to implement elastic SCUs with adaptations such as changes in size and ICU type. We took into account different performance metrics for an example of an adaptation based on ICU feedback. In the elastic SCU provisioning mechanisms the QoS requirements of the SCU customers are taken into account. In this chapter we extend those chapters with possible human-centric SLAs for Social Compute Units, we discuss SLAs for Individual Compute Units and how SLAs fit in an SCU provisioning platform.

6.1 Motivation Scenario

In Chapter 4 we discussed a facility-management scenario with proactive incident management, to justify the need of collectives such as SCUs and raise trust-related issues. Here, we take another scenario to justify the same, and raise negotiation related issues.

6.1.1 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 (mostly for mobile ones). 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 who would translate the book for the required time and within budget. Let us assume that the book is submitted to be translated as a task to a Socio-technical CAS. There are two possibilities in this case: 1) The SCU provisioning platform uses software translation services as a first round for translating the material and then assigns appropriate people/ICUs to fix the software translation, and lastly another SCU to do the final editing; and 2) the platform assigns the translation task to a set of ICUs, forming a language-translation SCU from start, and in a second iteration assigns the translated material to a set of ICUs, (can be the same SCU or a new one) for reviewing and final editing. Thus in this scenario two SCUs will be formed: a) *Language-translation SCU*, and b) *Reviewing and Final-editing SCU*.

6.1.2 On the need for SLAs supporting human computation

The scenario in Chapter 4 and the language translation scenario are very different and require very different types of SCUs. The first one forms multiple SCUs, while the work is done online as well as on-site and offline. The second concerns SCUs which are formed, invoked, and executed online, which is to say that the task execution and service execution is done as with software services - the SCUs lifecycle is completely online and the artifacts are delivered online. However, from both of the scenarios it is clear that SCU customers have certain requirements when submitting a request for SCUs.

Usually, SaaS, PaaS and IaaS are managed with Service Level Agreements. However, for efficient management of Socially-enhanced CAS we need proactive management of

people as well. To provide efficient performance of SCUs and minimize cost for the customers an SCU platform should strictly manage the ICUs, and this can be done only if there is a negotiation between it and the customer defining clear terms and conditions in the form of contracts such as Service Level Agreements (SLAs). There is no research in SLAs for human-computation specifically, to go 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, as aforementioned this is a challenge that needs to be addressed for Socially-enhanced CAS to be able to efficiently manage human-computation in an automated or semi-automated manner. In particular, 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 this, we identify some of the core characteristics of human-based resources that would impact the different types of SLA specification and management as follows:

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

6.2 Computational-Environment Setting

In [RTD14] we proposed a conceptual framework for provisioning SCUs on demand and their elastic management. With SCU provisioning platforms there is a crucial difference opposing those of pure software services and, needless to say, it relies on the unpredictable nature of humans. Hence, the SLA negotiation and establishment is two-fold, it concerns the customer and the SCU platform and the platform and the human-based services selected as SCU members. To achieve this and to prevent SLA violations the platform should have reliable monitoring and elasticity mechanisms. As aforementioned; an SCU has its own lifecycle, and so does an SLA. In Figure 6.1 we give an abstract overview of a SCU provisioning platform (without much details and this is not the focus of this work). The figure shows that SLAs are established between a customer and the platform for a specific project, which requires a specific SCU, so there are SLAs for SCUs. Depending on these requirements, the platform can run a ranking algorithm to rank ICUs from the ICU Pool (connected to it or a pool that it has access to) according to the requirements within the SCU SLAs. The ranking can be done according to the weight of importance

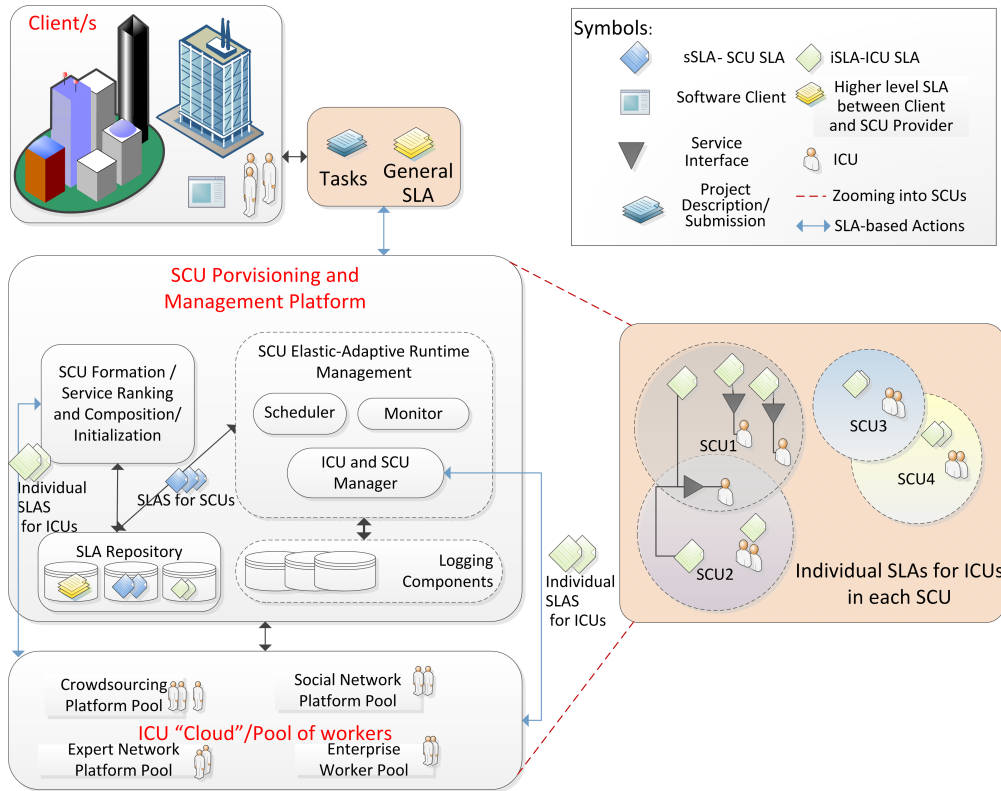


Figure 6.1: SLA-based ICU-SCU Provisioning and Management Platform

of different SLA parameters in the customer agreements. In the next step, a negotiating component will negotiate individual SLAs with ICUs in the ranked list (in a descending ranking order) and form an appropriate SCU. The negotiations should be automatic so that SLAs can be automatically changed at runtime if customers or workers decide to change and re-negotiate specific SLA parameters.

In previous chapters we have discussed how to enable elastic properties for SCUs. We showed the need of being able to adapt an SCU at runtime based on different elastic strategies which trigger events based on monitoring metrics. We discussed that an SCU can be elastically adapted when a threshold for some predefined customer-set constraint is reached, when customers change their requirements or decisions or based on workers' feedback about their willingness to work. In this work we bring SLAs to the elasticity model. From the aspect of SCU execution controlled by SLAs, the SCU may elastically be adapted, such that:

- compliance with run-time changes and customer or worker requirements is achieved,
- an SLA violation is prevented or managed.

The platform needs to enable the changing and adaptation of an SLA as the customer or worker requirements change. A customer or a worker 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 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 could be also conducted at run-time. 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 support elastic SLAs as well.

6.3 Modeling SLAs for SCUs

6.3.1 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 and establishment. Hence, the definition of Quality of service (QoS) metrics and the proper way of their monitoring is crucial for service providers as well as consumers. In Table 6.1 we list relevant metrics from our previous chapters 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, part of an SCU with s_i .

6.3.2 Penalties

Human service characteristics differ from those of software due to the human-centric dynamic and unpredictable behavior that is conditioned by many factors. Thus, penalties should be given an important role in defining SLAs for SCUs. Since we advocate for elastic SLAs and integrating SLAs to a collectives lifecycle, allowing SLA adaptation together with collectives' adaptations, we have two cases to consider regarding penalties: 1) when SLA parameters are changed at runtime, and 2) when there has been an SLA violation.

In Chapter 4 we defined the Socio-technical Trust score, calculated as: $STT(s_i) = w_{pt} * PT(s_i) + w_{mcts} * MCTS(s_i)$, 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:

$$STT(s_i) = w_{pt} * PT(s_i) + w_{mcts} * MCTS(s_i) + w_{mcts} * CPS(s_i), \quad (6.1)$$

where CPS is defined as:

$$CPS(s_i) = w_{pt} * CSS(s_i) + w_{mcts} * PSS(s_i), \quad (6.2)$$

where CSS is Client/Customer Satisfaction Score and PSS is Platform Satisfaction Score.

Properties	Description	Value/Range
Responsive-ness	Reciprocity	[0..1]
Consistency in behavior	Executed tasks as promised, e.g., via feedback messages	[0..1]
Socio-Technical Trust Score	Includes other metrics: social scores (subjective), and monitored performance metrics(objective), such as: productivity, effort, success rate, number 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 SCU members	%
Availability	An average of ICU availability (as percentage or time periods executing tasks)	hour
Budget	The max price that the customer is willing to pay	specific currency

Table 6.1: Notation and description of basic parameters for SCU SLAs

When an SLA change occurs at runtime, e.g., a customer requests a cost-change for a specific skill type, ICUs can be sent a notification request message for task acceptance for the new 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 6.1. The tasks for the changed cost are executed by other appropriate ICUs which are newly included in the SCU from the available pool of ICUs. ICUs should always be informed that they will get lower social scores which will influence their overall reputation score, so that this knowledge can sometimes serve as an incentive.

The same penalties can be applied in SLA violations. In these cases, in addition to lowering performance metrics and satisfaction scores, monetary penalties are unavoidable. These can sometimes be used as an incentive for keeping to the agreed contract parameters next time an ICU is assigned to an SCU. As an extension to our cost model from Chapter 3, the new cost of the SCU after adapting it because of a violation by ICUs will be:

$$Cost_{adapt}(scu_i) = Cost_{agreed}(scu_i) - \sum_{i=1}^m \sum_{x=1}^j c(s_i, t_x) + \sum_{i=1}^m \sum_{x=1}^j c(s_i^{nw}, t_x) - \sum_{i=1}^m c(s_i, vSLA_i), \quad (6.3)$$

where t_x denotes a task executed by an ICU s_i . In the case of a violation by the customer the new SCU cost will be:

$$Cost_{adapt}(scu_i) = Cost_{previous}(scu_i) + Cost_{customer}(c(scu_i), vSLA(c(scu_i), scu_i)). \quad (6.4)$$

6.3.3 Enforcing Privacy with SLAs

Currently, platforms in the industry that involve social-computing, as well as crowd-sourcing platforms all have a common flaw, lack of privacy. Current platforms all require private information to register on them, such as real names, addresses, date of birth, gender, education details. Some platforms also utilize other sensitive data such as current location information. In CAS, this problem is and will get even more serious if it is not approached with radical changes in the way we design and develop them. Sensitive information in socially-enhanced CAS can be collected from all the devices that people may use to execute tasks, to provide the end-artifact, as well as to require tasks. Some information, may be collected and stored for years without the knowledge and consent of the end-users. However, platforms usually publish their own Privacy Policies, about what information they collect and how they use them. These are not always fully transparent and understandable, and people rarely read them. SLAs can be a tool to enforce privacy in addition to privacy policies. Privacy clauses can be included in agreements. This can improve peoples knowledge if and what data will be collected during and after they engagement on a platform, whether as a worker or a client. Developers should always design privacy aware platforms of course and provide privacy-by-default. Some platforms allow users to delete their bank account information after they have transferred the needed money to use the platform as requestors of work¹. However, in situations where an application cannot work or a service cannot be provided without data (like medical applications) SLAs can help in getting consent. Some privacy conditions that can be included in SLAs, may be: *specification of server locations if the platform has servers in multiple locations, software to be used for task execution and communication (e.g., encrypted), 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, time period after a security breach to inform the users about it, privacy breach penalties for the platform.*

6.3.4 Examples

To date, there is still no standardization about exposing ICUs as services, but as we discuss in the introduction section and in related work, existing work has shown that it is possible that they are provided as web services. Thus, for establishing an SLA we turn to standards. Examples are the web service agreement specification (WS-Agreement) [ACD⁺07] and the Web Service Level Agreement (WSLA) language specification [LKD⁺03], which were proposed as a way to facilitate and standardize the establishment of service level

¹<https://microworkers.com/>

agreements between service users and service providers. Listing 6.1 provides an example SLA describing the *parties* involved in the agreement, in this case a customer and the SCU provisioning platform. Listing 6.2 provides an example of the *obligations* part of the same SLA presented in Listing 6.1, where we include some metrics discussed in this section. Listing 6.3 shows a similar part of an SLA between the SCU provisioning platform and an ICU, illustrating the hierarchical SLA establishment between ICU and the platform on one hand, and the platform and customers on the other.

Listing 6.1: Example WSLA - *Parties* part

```
1 <wsla:SLA
2 xmlns:wsla="http://www.ibm.com/wsla"
3 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4 name="SLA11">
5   <wsla:Parties>
6     <wsla:ServiceProvider name="Customer">
7     </wsla:ServiceProvider>
8     <wsla:ServiceConsumer name="Platform">
9     </wsla:ServiceConsumer>
10  </wsla:Parties>
11  <wsla:ServiceDefinition name="SCUProvisioning">
12  <wsla:Operation xsi:type="wsla:
WSDLSOAPOperationDescriptionType" name="ProvisionSCU">
12 <wsla:SLAParameter name="SCUAvailability" type="int" unit="Hours">
12 <wsla:Metric>CalculateAllICUAvailability</wsla:Metric>
12 </wsla:SLAParameter>
12 <wsla:SLAParameter name="SCUMaxBudget" type="double" unit="₹">
12 <wsla:Metric>CompareBudget</wsla:Metric>
12 </wsla:SLAParameter>
12 <wsla:SLAParameter name="SCUSocTechTrust" type="double" unit="₹">
12 <wsla:Metric>CalculateTrustScore</wsla:Metric>
12 </wsla:SLAParameter>
12 <!--_details about other parameters and metrics_-->
13 </wsla:Operation>
14 </wsla:ServiceDefinition>
15 <!--_definition of Obligations_-->
16 </wsla:SLA>
```

Listing 6.2: Example WSLA - *Obligations* part

```
1 <wsla:Obligations>
2 <wsla:ServiceLevelObjective name="sloMetrics" serviceObject="ExecTask">
3   <wsla:Obligated>Platform</wsla:Obligated>
4   <wsla:AND>
5     <wsla:OR>
6       <wsla:Expression>
7         <wsla:Predicate xsi:type="GreaterEqual">
8           <wsla:SLAParameter>SCUAvailability</wsla:SLAParameter>
9           <wsla:Value>12</wsla:Value>
10        </wsla:Predicate>
11      </wsla:Expression>
12      <wsla:Expression>
13        <wsla:Predicate xsi:type="LessEqual">
```

```

14 <wsa:SLAParameter>SCUMaxBudget</wsa:SLAParameter>
15 <wsa:Value>12750.00</wsa:Value>
16 </wsa:Predicate>
17 </wsa:Expression>
18 </wsa:OR>
19 <wsa:Expression>
20 <wsa:Predicate xsi:type="GreaterEqual">
21 <wsa:SLAParameter>SCUSocTechTrust</wsa:SLAParameter>
22 <wsa:Value>0.8</wsa:Value>
23 </wsa:Predicate>
24 </wsa:Expression>
25 </wsa:AND>
26 <!--_further details_-->
28 </wsa:Obligations>

```

Listing 6.3: Example WSLA - *Parties* part

```

1 <wsa:SLA
1 xmlns:wsa="http://www.ibm.com/wsla"
1 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1 name="SLA12">
1 <wsa:Parties>
1 <wsa:ServiceProvider name="ICU">
1 </wsa:ServiceProvider>
1 <wsa:ServiceConsumer name="Platform">
1 </wsa:ServiceConsumer>
1 </wsa:Parties>
1 <wsa:ServiceDefinition name="ICU_1">
1 <wsa:Operation xsi:type="wsa:
WSDLSOAPOperationDescriptionType" name="ProvideSkill">
12 <wsa:SLAParameter name="SkillType" type="Skill" unit="String">
12 <wsa:Metric>CheckSkillType</wsa:Metric>
12 </wsa:SLAParameter>
12 <wsa:SLAParameter name="CostPerSkillType" type="double" unit="₹">
12 <wsa:Metric>CompareICUCost</wsa:Metric>
12 </wsa:SLAParameter>
12 <wsa:SLAParameter name="SCUSocTechTrust" type="double" unit="₹">
12 <wsa:Metric>CalculateTrustScore</wsa:Metric>
12 </wsa:SLAParameter>
12 <!--_details about other parameters and metrics_-->
13 </wsa:Operation>
14 </wsa:ServiceDefinition>
15 <!--_definition of Obligations_-->
16 </wsa:SLA>

```

6.4 SLAs and Elasticity

6.4.1 Programming SLA Parameter Changes at Runtime

In Algorithm 6.1 we show an example of adapting an SCU at runtime when a specific SLA Parameter is changed by the customer at runtime, namely cost. We suppose that

we have a ranked pool of resources by ranking requirements set from the customer and that an SCU is formed from that ranked list, and describe an adaptation algorithm of an already formed SCU. When the budget for the SCU is lowered, this strategy calculates new costs for each skill type and sends a notification request message for approval or rejection of the changes to each ICU, (lines 1-5 of Algorithm 6.1). Figure 6.2 shows the process in Business Process Model and Notation (BPMN) notation. The Platform Satisfaction score and reciprocity, metrics which we have described earlier, are lowered for those ICUs that have rejected the payment changes, and those ICUs are excluded from the collective (lines 7-13 of Algorithm 6.1). The tasks for specific skills for which the cost is changed is then delegated to ICUs from the pool of available ICUs outside of the SCU, and those ICUs that accept the agreement terms are then included in the SCU.

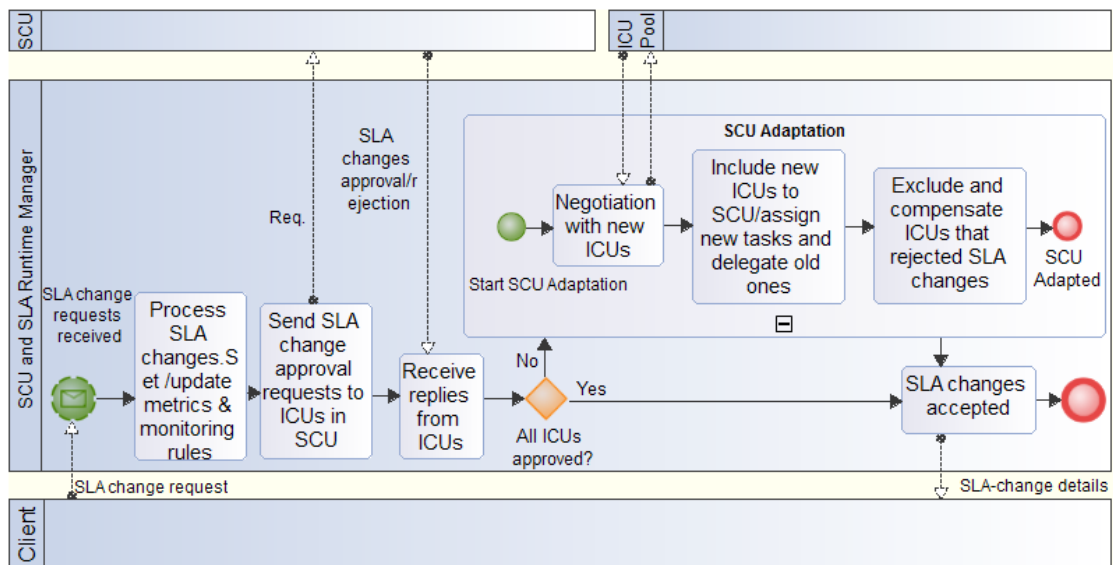


Figure 6.2: Elastic SCU adaptation with SLA cost changes

6.4.2 Implementation of a Proof of Concept prototype and Experiments

We ran a Java-simulation experiment to evaluate runtime adaptations of SCUs when requirements of parameters are changed at runtime. We designed and implemented ICUs with skill (each individual ICU having a single skill, for better overview of the results) and cost per task as properties. In addition, we designed the metrics (that we have previously defined in [RTD15] and this work). There are both atomic metrics which are designed by monitoring one parameter only, and non-atomic metrics, which are comprised of multiple atomic ones. The following are only some of the metrics: availability, total assigned tasks, delegated tasks, success-rate, productivity, reliability, social trust, performance

Algorithm 6.1: Elastic Adaptation of SCUs based on SLA changes

```

Data: icu member of SCU, icu member of ICU Pool
1 if scu.currentBudget == (scu.currentBudget - amount) then
2   forall icu in SCU do
3     calculateNewFees(taskList)
4     sendFeeChangeNotification()
5     icuApprovalList ← getApproval(icu)
6   end
   // Adapt the SCU if not all ICUs approved the agreement changes
7   forall icu in icuApprovalList do
8     while icuCurrent.Approval() == false do
9       icuNeededSkillsList ← icuCurrent.getSkill()
10      icuCurrent.icuMetrics.PSS = getPSS(icuCurrent) -  $\partial$ 
11      updateAgreementReciprocity(icuCurrent)
12      icuCurrent.updateMetrics()
13      SCU ← removeICU(icuCurrent)
14    end
15  end
  /* Add the next ICU from the ranked list that has the skills from the
  removed ICU that approves the agreement */ forall icuNeededSkillsList do
17    forall icu in rankingList do
18      if icu.Approve() == true then
19        | icuCurrent = icu SCU ← addICU(icuCurrent)
20      end
21      Break
22    end
23  end
24 end

```

trust, socio-technical trust. We designed tasks with skill and cost requirements, and SCUs with the same metrics as for ICUs. 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; and 4) a process execution engine, as shown in Figure 6.3.

SCU Adaptation with SLA parameter changes at runtime. We implemented a variation of Algorithm 6.1 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 and delegated already assigned tasks for which the fee was lowered to other ICUs with matching skill and (lower) cost requirements. In addition, tasks were also delegated at points when they were assigned but were not executed on a time-threshold. In both

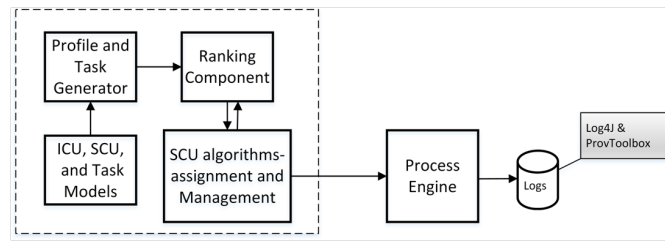


Figure 6.3: Proof-of-concept prototype

cases tasks were delegated to ICUs already within the SCU or new ICUs were invoked from an ICU pool if no matches were found. Let us examine the results, which are shown in Figure 6.4. Fig. 6.4 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, 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, which score here we randomized giving higher scores to ICUs that were more productive and lower scores to those that delegated their tasks. In a) we can see that the trust scores are generally rising, but some low points exist. The cost change requirement for SCU1 and SCU2 are 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 last adaptation, 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)), so SCU performance is high, some ICUs are excluded from the SCU and more tasks are executed with lower nr of people. The case with SCU2 is very similar. 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), 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 the those changes were accepted by ICUs already in the SCU, in other words ICUs with appropriate skills accepted the lowering of the cost per task. SCU6 on the other hand has a clearly low score of STT at point 6 in sub-figure a). Sub-figure b) shows that the number of 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, and we gave as an input a very low cost 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 adds to the lower value of the STT. In summary, this kind of flexibility at runtime is not possible without having a platform design that provided the possibility to change SLAs at runtime but most importantly to include ICUs in negotiations when SLAs change so that collective adaptations at run-time are easier, efficient and agreeable for all parties.

For illustration, a code snippet of adding a new ICU that had accepted the new cost, by acknowledging a willingness to work with the new cost, is shown in Figure 6.6. The

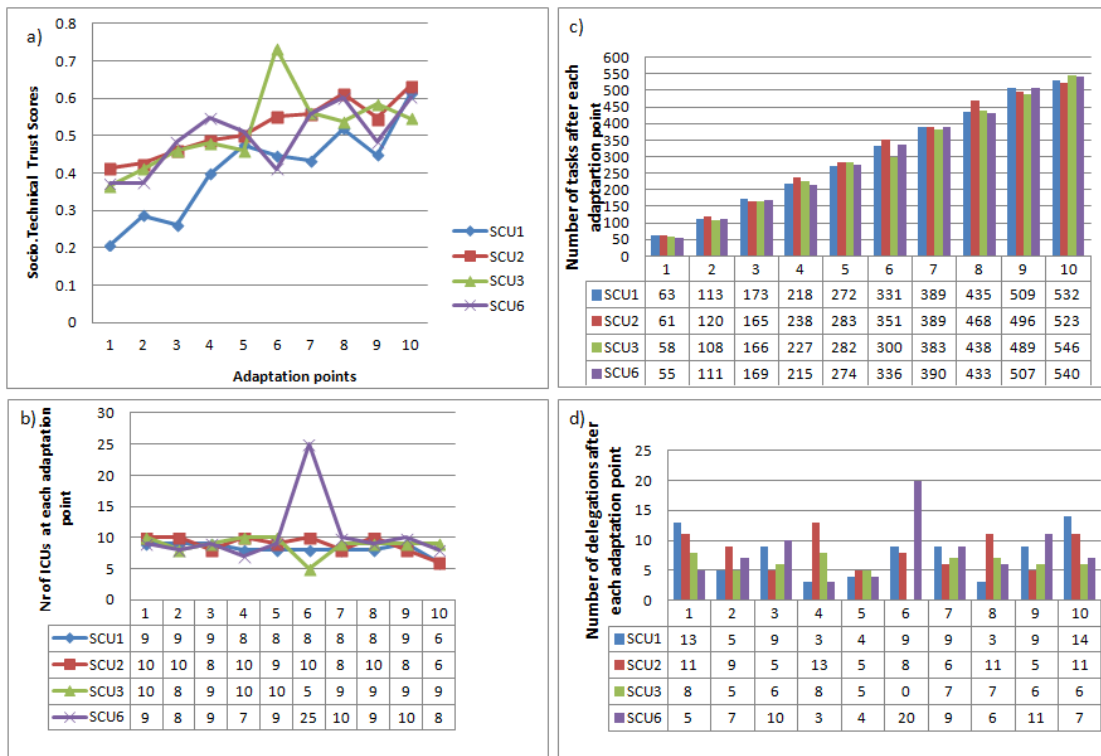


Figure 6.4: Elastic SCU adaptation with SLA cost changes

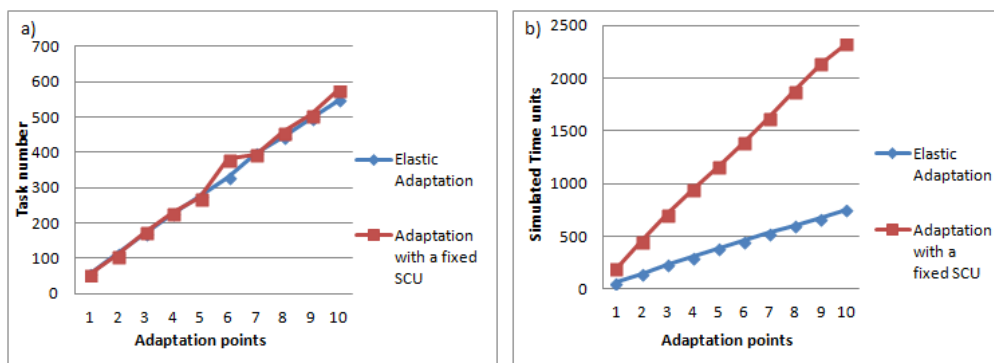


Figure 6.5: Time Comparison of elastic and fixed SCU adaptation algorithms

code snippet is just an illustration of different possible cases for task-delegations, it differs from our experiment in that it checks all available ICUs from the ranked list that have a lower cost for a specific skill and are willing to work for that cost, and that do not belong to the current collective, the task is then delegated to the selected ICU and the selected ICU added to the collective.

```
//Delegate tasks for which parameter-change requests are made, e.g., cost.
public static void negotiateSLAPar(ICUList icus) throws IOException {
    // The two methods take parameter changes as input.
    setICUCostperSkillReq();
    setParameterChangesReq();
    for (int k = 0; k < taskList.size(); k++) {
        tempTask = taskList.get(k);
        currentICU = tempTask.getTaskOwner();
        //Run through all ICUs in the ranked list of ICUs.
        outerloop:
        for (int b = 0; b < icus.size(); b++) {
            icutemp = icus.get(b);
            //I skip here the ICU to which an ICU is assigned
            if ((tempTask.getTaskOwner().getICUId() != icutemp.getICUId())
                && (!currentSCU.contains(icutemp))) {
                //I
                innerloop:
                for (Map.Entry<String, Double> entry : skillcost.entrySet()) {
                    /*Find the first ICU with the required skill-type and the newly-changed cost.
                    Delegate the task for which the cost is changed if ICU is found,
                    if not leave it with the old ICU.*/
                    if ((tempTask.getSkillReq().equals(icutemp.getICUSkill()))
                        && (entry.getKey().equals(icutemp.getICUSkill()))
                        && (entry.getValue() > icutemp.getICUCostPerTask())
                        && (entry.getValue() < currentICU.getICUCostPerTask())
                        && (entry.getValue() != 0)
                        && (icutemp.willingAck())) {
                        /*Metrics updates: currentICU is the old ICU,
                        icutemp is the ICU to which the task is delegated...*/
                        icutemp.updateICUTaskQueue(tempTask);
                        icutemp.ICUMetrics_.updateICUTaskQueue(tempTask);
                        currentICU.ICUMetrics_.removeTask(tempTask);
                        //Add the new ICU to the SCU.
                        currentSCU.addICU(icutemp); //add(icu);
                    }
                }
            }
        }
    }
}
```

Figure 6.6: Code snippet from our experiments for delegating a task with a new cost to a new ICU

Comparison of elastic and fixed SCU adaptation. We ran another experiment comparing the variation of Algorithm 6.1 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 6.1 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 changes. We compared the two in terms of time efficiency. Figure 6.5 shows the results. At each adaptation point a new bag of tasks is assigned to SCUS, Figure 6.5 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 significantly, as shown in sub-figure b). We designed ICUs to randomly have an "accept" field as "true" for the newly calculated cost and thus the tasks were delegated to ICUs that match the skills and accepted the tasks.

6.5 RelatedWork

There is very little work on SLAs in human computation environments. Initial work which concerns crowdsourcing environments in particular is presented in [KPSD11]. 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 [PSSD11] by Psaiet al. The authors discuss scheduling mechanisms for crowdsourcing environments, which will meet the requirements of contracts, between multiple entities/roles. Schall in [Sch13a] discusses protocols for human computation and supports our claim that platforms for provisioning social computation need to work with SLAs, for better task management, so that the requester's requirements are met, just as in software services. A description of requirements and considerations for platforms that support Quality of Service management with SLAs in human computation is introduced in [KZA08]. Authors in [CGR⁺13] have presented a priority-based assignment of human resources to business process activities, based on ranking of human resources. Enabling requirement-changes at runtime would give more flexibility to these type of approaches.

As we mentioned that Social Compute Units should be provisioned elastically so that the performance and time of humans on task execution is utilized optimally for specific fees (or other types of rewards), elasticity plays a role in SLAs as well. Elastic management of properties such as cost and workload are presented in [Sch13b], 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 [MMDC⁺07], where the authors present a specific proposal for extension of the WS-Agreement Specification to support temporality.

Müller et. al in [MGMD⁺14] 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 in the

work for Guarantee Terms with compensation functions). They also provide examples of real world SLAs, analyzing them and explaining how their compensation model would fit in those existing specific examples of SLAs. Interesting for our work is the fact 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 [MH11] also discuss a contract model and a negotiation process in a social computing scenario considering the nature of social computing where rewards and penalties are not only monetary.

Provenance in Human Computation

In social computing environments and collaborative ones in particular, there are several information types that are of significant importance, such as information about who worked on tasks, who created and influenced different produced artifacts, which events occurred during run-time and what was their effect and how were they approached, and similar. This information may be utilized for several reasons, such as for calculating performance metrics, which in turn influence reputation scores and consequently task assignments; for accountability reasons for workers in case of erroneous or problematic results; for having a clear overview of which artifacts were created by which organization in complex inter-organizational collaborations. This type of information can be stored as *provenance* data. While appreciable work exists on metrics, monitoring strategies, coordination and management mechanisms for social computing concepts, little work exists on the utilization of provenance in social computing. However, we believe that provenance data can help social computing systems in several contexts, such as: a) extracting different behavior patterns for workers, based on which management mechanisms can be improved; b) easier visualization of events for business stakeholders; c) sharing of a common provenance model by multiple social computing environments, which would give a better overview of the capabilities of workers, and most importantly it will allow interoperability, so that workers from several different provisioning platforms can be invoked to work in a common collective. The main contribution of this chapter is to investigate how provenance data can be used in social computing, the benefits it can offer and trade-off implications, taking the concept of SCUs as our case study.

7.1 Motivation

7.1.1 Provenance data in social-computing management-mechanisms

We describe next, the most important cases where performance and interaction data is crucial for social computing systems, both from the perspective of an individual and from the perspective of collectives.

Individual Task-assignment and Formation of collectives/Social Compute Units

Whether it is for individual task execution or for task assignment in collectives, workers are ranked based on several pre-set metrics regarding their performance, with which the appropriateness for a task is assessed. Worker selection algorithms are also run during runtime, when a new worker is needed due to different events, such as unexpected task generations with new skill requirements, or insufficient capacity of a collective to handle tasks. Data about worker performance through time is crucial for designing effective ranking and selection mechanisms.

Adaptation mechanisms for Social Compute Units

Defining, modeling and measuring workers' performance is also important for novel systems that enable elastic adaptation of Social Compute Units. Algorithms that calculate ICU performance can be used with adaptation mechanisms that include task delegations based on events. A task can be delegated at runtime, to a worker who had not been included in the collective from start. Consequently, with elastic-adaptations of collectives, a worker can be added to a collective as well as excluded from a collective at run-time [RTD14]. After each adaptation, and at the end of each SCU execution, worker metrics are updated.

Misbehavior prevention and False negatives in Misbehavior detection

Worker misbehavior, such as assigning a lower vote to another worker on purpose, tricking the system by sometimes uploading non-complete or unsatisfactory results and still getting rewarded and other times uploading satisfactory results are common in human computation in general. With available provenance data of workers over a period of time, misbehavior detection mechanisms can be developed, by extracting different misbehavior patterns and models from that data. Moreover, mistakes can be treated as a misbehavior sometimes, and if there is a way to visualize which worker did what during an SCU execution, mistakes can be found more easily, false negatives identified and existing misbehavior patterns corrected.

Incentive mechanisms

Social computing systems employ monetary or non-monetary based incentive mechanisms. Some incentive mechanisms are based on updating *trust* and *reputation* metrics for workers, which are defined considering performance-based and interaction-based historical data of workers, whereas some incentive mechanisms are purely monetary. Incentive models utilize worker interests as well as behavior over time beginning from the starting point of worker engagement in a system, and in this way different incentive plans for each worker can be built.

Compensations

Closely related to incentives, compensation mechanisms are also based on monetary and/or non-monetary rewards or sanctions, both for workers and customers. One of the most common compensation types is updating worker and customer reputation. Once again, these mechanisms are based on appropriate metrics with which worker behavior is monitored over time, as well as metrics for customer behavior, e.g., their payment method and behaviors, and/or sincerity when specifying their satisfaction from the received results. Regardless of the motivation of people, whether they want to work voluntarily to benefit a cause, or to get paid, workers need to have a clear overview of their progress and of the way in which their work was rewarded or sanctioned over time, so that they can make better decisions as how to act in their future engagements in social computing systems.

7.1.2 Challenges

All of the aforementioned strategies, require data regarding workers, and most of it depends on data that is updated based on workers' history of interactions and performance. Thus, provenance can be a good fit to represent historical data for workers. We hypothesize that with provenance-data, the aforementioned mechanisms would serve social computing provisioning-systems in making them more efficient by focusing on providing qualitative work to get high-quality results, in shorter time and on a lower budget, than expected from cases when provenance data is not utilized. Several research questions arise from our hypothesis:

- Can provenance help us in tracing the path of a particular task, which means who worked on a particular task, who was assigned with the task and successfully executed it, and who was assigned a task and delegated it and why?
- Can provenance data help us in conceiving, modeling and defining novel metrics that could be used in worker/ICU ranking and selection algorithms for collective formations as well as for elastic-adaptation mechanisms? Consequently, can we enhance existing trust and reputation models, incentive mechanisms and rewarding with new metrics derived from provenance data?
- Can we predict which ICUs collaborate well with which ICUs based on provenance-data of their performance and interactions within the same SCU?

- How can a worker trace his/her progress across several SCUs and extract relevant information that would benefit his/her future behavior in being more productive, and contribute more efficiently in future engagements?
- Can we trace the events that brought to SCU adaptations and the parameters of the SCU adaptation due to those events, so that in future executions of SCUs with similar customer requirements and constraints, adaptations could be more efficient due to action plans that can be devised from provenance data that trace SCU executions?

7.2 SCU Environment and Provenance

7.2.1 SCU Environment

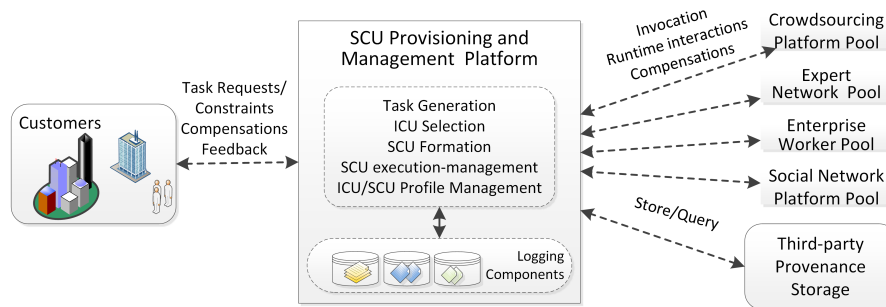


Figure 7.1: SCU environment model.

Figure 7.1 shows a model of an SCU provisioning platform. We do not show specific components as this is not the focus of this work, but some core management mechanisms of an SCU platform are listed. SCUs have their own life-cycle, so customers that can be companies, different organizations, private individuals etc submit their task request to the SCU provisioning platform, the platform then generates tasks based on the requirements and constraints given, e.g., required skills, cost, deadline. An ICU selection module, which is a task-to-ICU matching module returns the most appropriate ICU for each task. A collective formation module is a composition module, and an orchestrator. ICUs and SCUs have their own profile, which includes ICU specific metrics and which is monitored and updated at run-time and at the end of the SCU execution. Customers can give feedback about the final results, and ICUs are rewarded or sanctioned at the end of the SCU execution. Depending on the domain, an SCU provisioning platform may have its proprietary pool of resources, but human-based resources/services can also be invoked from multiple external sources. We have implemented a proof-of-concept prototype with ranking, monitoring, collective-adaptation algorithms and profile updates, (presented also in [RTD15]), and here we present our updated prototype which includes mapping of our collective-adaptation algorithm result-logs to PROV terms and adding provenance-data to our prototype, including visualization of ICU, SCU, and task related provenance data.

From the perspective of a specific SCU environment, a provenance approach to logging can give an overview of historical data useful for the mechanisms supporting social collective provisioning and management. On the other hand, if we assume that there can be multiple social computing milieus and that a third-party can provide provenance storage for multiple SCU provisioning platforms, workers can be treated equally regardless of the fact on which platform they have been registered, and from which SCU environment they are invoked. This brings to interoperability of multiple SCU environments, which will allow for a common large-scale pool of human-based resources bringing flexibility to the types of social-computing applications.

7.2.2 Modeling Provenance for SCUs

Provenance data is meta-data, and can be used to prove the ownership of data, the source of data, and how it changes over time. Provenance has mostly been used to trace data flows and to keep records of data in scientific experiments so that they can be repeated, e.g. in workflows. Today, provenance is gaining more usage in various areas as complex systems have complex data flows, which can be managed with information extracted from provenance data.

A provenance data model named PROV-DM [MM13] has been standardized by W3C on which we base our discussion. Other models and extensions exist as well, e.g., [MLJ⁺10]. The PROV data model provides several types and relations, the fundamental of types being: an Agent, an Entity, and an Activity. An Agent is a type which as a result of an engagement in an Activity it creates, or has an effect on an Entity. Some of the core relationships between the aforementioned types are, an association relationship, `wasAssociatedWith`, which describes a directed relationship from an Activity to an Agent; `wasAttributedTo` describing a relationship for cases when a particular Entity is attributed to an Agent, which means that an Agent has had an effect on an Entity; a `generatedBy` relationship describing cases when an Entity is generated, created, or updated by a specific Activity, and others.

Mapping the SCU construct and the actors in our system to PROV terms, we have ICUs, SCUs, Customers, and the Task Manager and ICU Manager represented by Agents. ICU and SCU profiles in which we store data for each ICU and the SCU respectively, are represented by Entities. In addition, both task related requirements, as well as the actual system-generated tasks identified with an id are also mapped to Entities. Activities in our model are the following actions: task assignment, task delegation, successful task execution, ICU profile update after each execution, and SCU adaptation based on events. Figure 7.2 shows the graphical model of our SCU environment, the implementation of which we discuss in the following section. A customer makes a request for a specific project, and based on those requests, task specifications are generated by a task creation activity associated with the ICU Manager. Activities related to tasks are collapsed in our model, as all task-related activities, are associated with an ICU, an SCU, a software agent. This is because of the reason that to extract valuable information we need to have information about which ICU executed which task and to which SCU/team that

ICU belonged to. The software agent is equipped with multiple task-to-ICU matching algorithms, as well as scheduling algorithms, thus the task activities are associated to the software agent through a plan, shown with the PROV relation `hadPlan` in Figure 7.2.

Figure 7.3 shows a more specific example. To specify, in an SCU provisioning environment we assume that people register their profiles just as in crowdsourcing environments. Thus, both ICUs, Alice and Bob, as agents would have engaged in activities to create their profiles. After a certain point Alice realizes that she can not execute the task assigned to her by a software agent on time because she is busy working on other tasks. Thus, she decides to delegate this task to Bob. In this case we have an Activity, Assignment for $Task_1$ in the figure, which is associated with both Alice and a software agent that assigned the task initially. Because the decision to delegate the task is made by Alice, we have another Activity that we labeled Delegation and Re-assignment for $Task_1$, and which is associated with Alice (because the task is delegated from her), Bob (because the task is delegated to him) and the software agent. The delegation activity is associated with the software agent because it keeps track of tasks, ICU profiles, monitors task executions, and includes scheduling mechanisms. We do not describe in details the software architecture as this is not the goal of this work. Figure 7.3 shows entities that show ICU profile characteristics after a certain time point when events occurs. As SCUs are constructs that can be managed by software or humans we show in the figure two cases. Alice in the figure has updated her profile at some point in time during the SCU execution, e.g., she has added an additional skill, while Bob's profile is updated by a software agent, e.g., his trust score is raised because he accepted and executed a delegated task. Figure 7.2 shows the graphical model of our SCU environment, the implementation of which we discuss in the following section. Figure 7.4 provides a code snippet of mapping SCU-related types to PROV terms, from results of our experiments stored in a log-file, which we describe in the next section.

7.3 Experiments

7.3.1 Setup

We have implemented a Java-based prototype that is a simulated social-computing environment. We have modeled an ICU with static properties such as Id and skill type, and dynamic properties, some of which are descriptive e.g. cost per task, and others that are mathematically defined and calculable. Thus, we designed metrics that we have identified as relevant for ICUs, and that serve as key performance indicators for them as dynamic properties. The most important metrics to mention here are: *effort*, *productivity*, *willingness*, *reliability*, and *performance trust*, which is a weighted aggregate metric of the aforementioned ones; *social trust*, which we define as a weighted aggregate of votes from people that the worker has collaborated with, and *socio-technical trust* which is a weighted average mean of the aforementioned performance-based trust and social-trust metrics. For the definitions of the aforementioned metrics, we refer you to the previous chapters. We modeled SCUs as lists of ICUs, which in turn have their own metrics. A

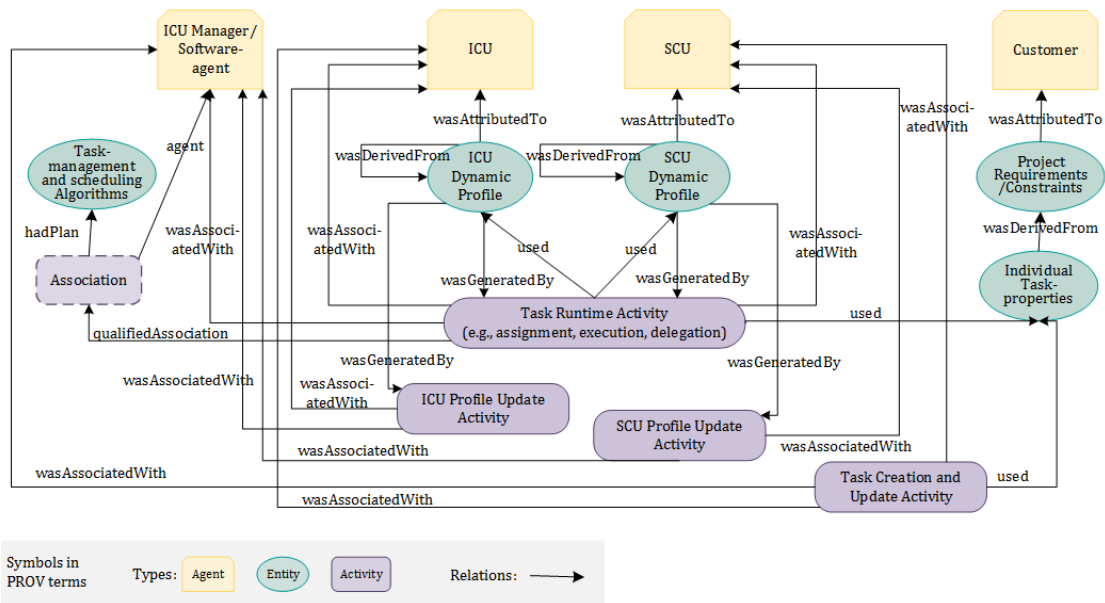


Figure 7.2: A graphical provenance-model based on the PROV-O specification, focusing on ICU task executions and profile updates.

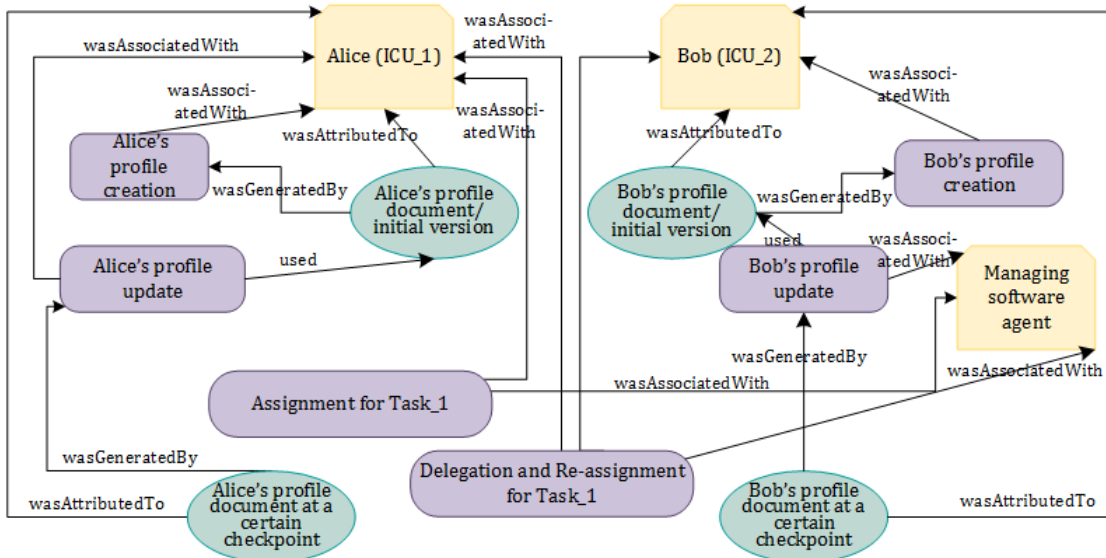


Figure 7.3: A specific example focused on ICU task assignments and profile updates.

```

Entity task = provFactory.newEntity(qn("T_"+taskID));

//Agents
Agent icuDelegatedTo = provFactory.newAgent(qn("ICU_"+icuID));
Agent icuFirstAssignedTo = provFactory.newAgent(qn("ICU_"+icufID));

//Relations
Activity assignment = provFactory.newActivity(qn(assignedID));
Activity delegation = provFactory.newActivity(qn(delegationID));
WasAssociatedWith wawAssignment = provFactory.newWasAssociatedWith(qn("waw"+assignedID+"_checkpoint"+checkpoint),
    assignment.getId(), icuFirstAssignedTo.getId());
WasAssociatedWith wawDelegation = provFactory.newWasAssociatedWith(qn("waw"+delegationID+"_checkpoint"+checkpoint),
    delegation.getId(), icuDelegatedTo.getId());
WasAttributedTo watAssigned=provFactory.newWasAttributedTo(qn("watA"+checkpoint), task.getId(),
    icuFirstAssignedTo.getId());
WasAttributedTo watDelegated=provFactory.newWasAttributedTo(qn("wat"+checkpoint), task.getId(),
    icuDelegatedTo.getId());
Used usedAssignment = provFactory.newUsed(assignment.getId(), task.getId());
Used usedDelegation = provFactory.newUsed(delegation.getId(), task.getId());

```

Figure 7.4: Code snippet from defining Entities, Activities and Relationships.

task is designed with specific properties, such as the skill type required to execute it, the cost for executing it, and the deadline to execute it.

7.3.2 Dataset

To test what type of provenance data can we store and how the visualization of this data will help us interpret results in a social computing environment in an efficient way, we implemented a specific task-execution and worker/ICU management mechanism. We generated 200 ICUs with different skill types and costs per task, as well as random initial values for the aforementioned metrics, all of them in the $(0,1]$ interval. For a faster algorithm run, instead of ranking ICUs for each task assignment, we ran a ranking algorithm first, to rank all ICUs, regardless of their skill type, based on the weighted values of three different metrics, given as an input: social trust, reputation and reliability. For ranking we implemented an algorithm based on the Analytical Hierarchy Process model (the description of which is out of the scope of this paper).

Next, we ran a task assignment and scheduling algorithm as in the following: we generated a bag of tasks with 40-50 tasks and assigned them in a FIFO order to ICUs from the ranked list, this time matching the skills of workers with those of the required ones for each task. The ICUs selected in the initial assignment form an SCU. We configured the algorithm and ran it three times to get different log files each time, with 10, 50 and 400 bags of tasks, each time sequentially assigning 40-50 tasks to ICUs, after each execution of a bag-of-tasks.

Our scheduling algorithm was designed such that it allowed for elastic adaptation of the initial SCU, such that tasks that reached a threshold in the queue of an ICU were delegated to other ICUs. Delegated tasks were assigned either to ICUs already within the SCU or to an ICU from the pool of ICUs from the ranked list, depending on a willingness value of ICUs for executing a task and availability. The willingness value was set randomly to 0 (not willing) or 1 (willing). ICUs that did not have any task assigned in a run, were excluded from the SCU, and new ones to whom task were assigned were

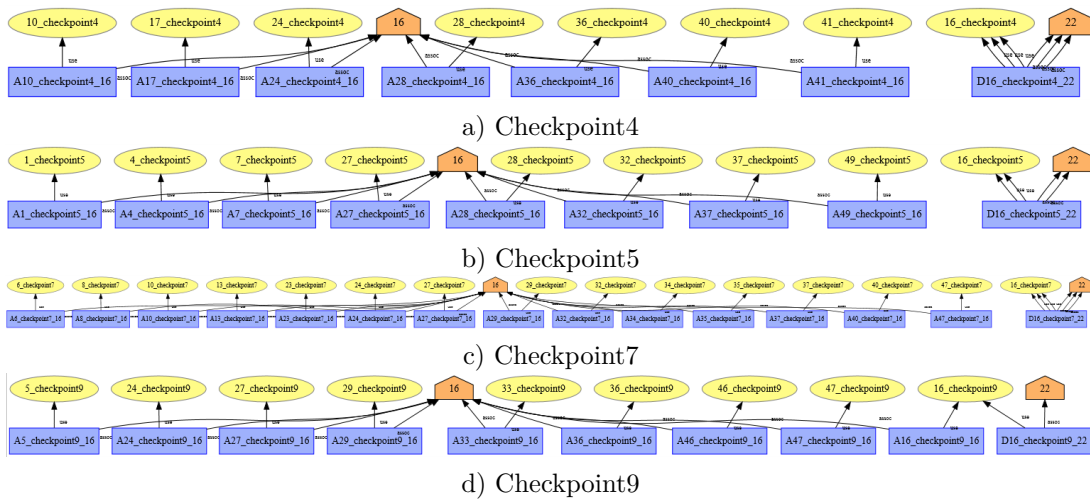


Figure 7.5: Tasks for ICU 16 and 22 at four selected checkpoints during one SCU execution.

added to the SCU. Hence the number of ICUs in each run fluctuated, and with this we updated the metrics, indicators of their performance and interactions, after each bag of tasks that was assigned and executed by the collective/SCU. For clarity, we denote the update of the properties of SCU members after each bag-of-task assignment, as a checkpoint. Thus, a checkpoint points to one bag-of-tasks execution. We generated a log file of the SCU execution, using the Apache log4j logging utility, storing metric values for every SCU member for each checkpoint.

We generated three log files of the SCU executions, using the Apache log4j logging utility, storing metric values for every SCU member for each checkpoint, for the three runs, thus we have a log file with 10 checkpoints, a log file with 50 and another with 400 checkpoints.

7.3.3 Experiment types and Results

Provenance Visualization

Utilizing the log4j log file, we mapped ICUs, SCUs, Tasks, and events during the SCU execution (e.g., task assignment, delegation, ICU profile updates and SCU adaptation) to provenance notation. We conducted the mapping using ProvToolbox [Mor], which allows for creating PROV documents in Java. Thus, with ProvToolbox, we generated XML files with provenance tags, as well as provenance graphs (svg files) that reflect provenance types and relations from the mapping (as in Figure 7.2). Figure 7.10 illustrates an xml output for one particular checkpoint regarding activities where an entity is an SCUs profile storing SCU related metrics, while SCUs only with Id's are mapped as agents.

We visualized provenance data in three contexts: a) *the ICU context* showing which ICUs were included within a collective at specific time-points, as in the example shown in

7. PROVENANCE IN HUMAN COMPUTATION

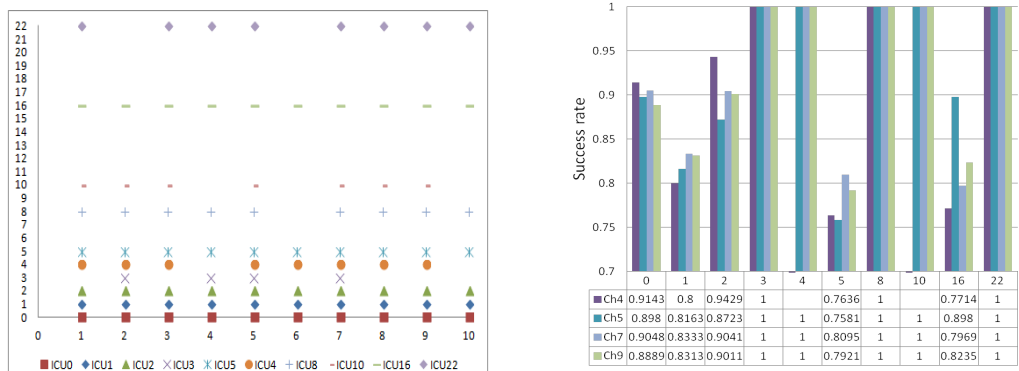


Figure 7.6: Provenance details after a run of an SCU adaptation-algorithm

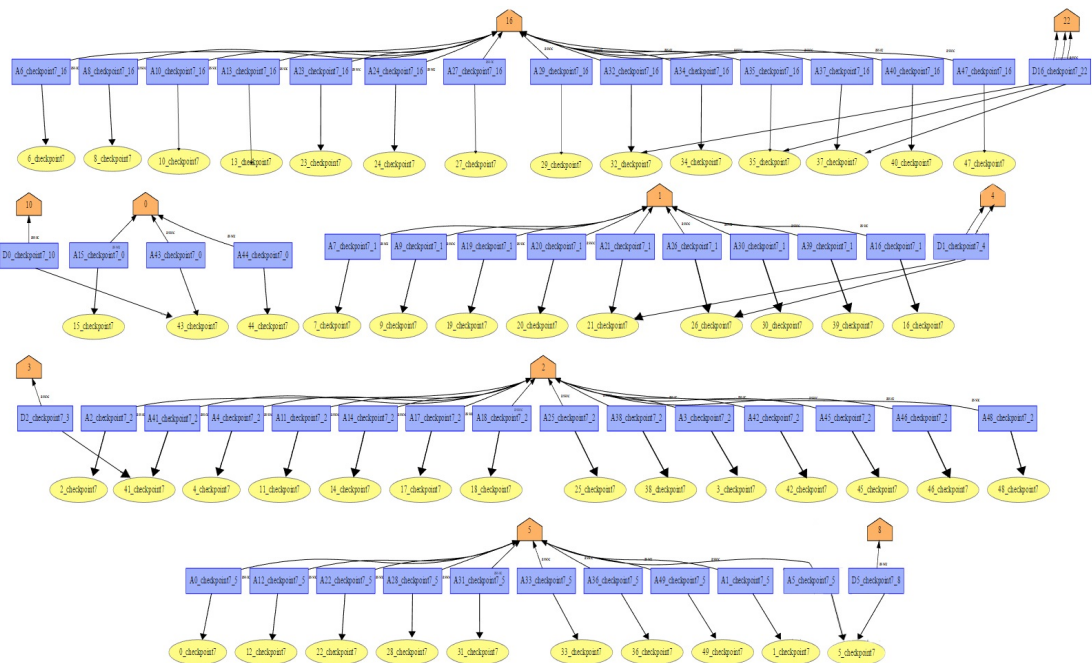
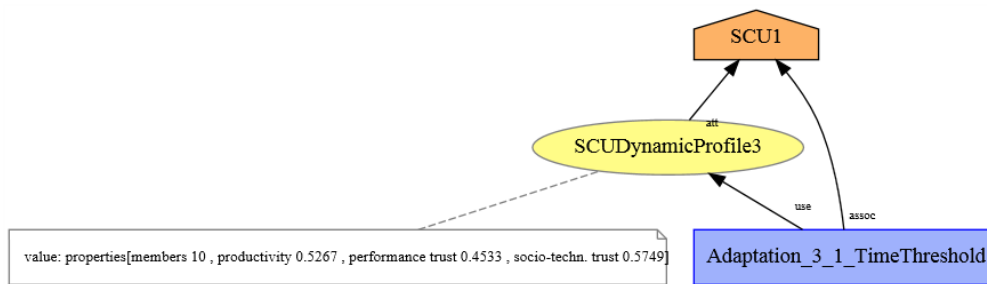
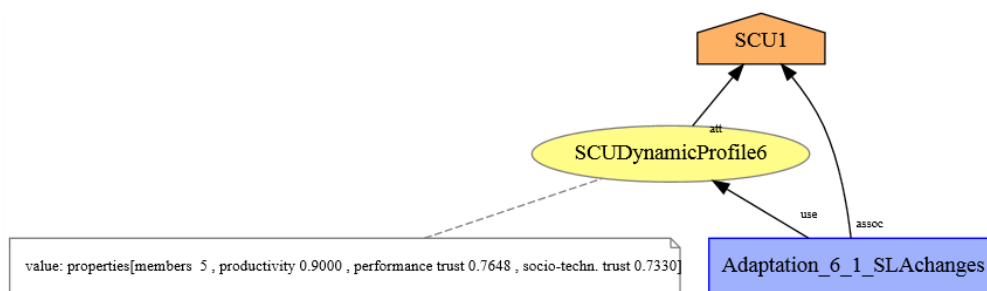


Figure 7.7: ICUs and tasks during the execution of one bag-of-tasks. The graph result of checkpoint 7.

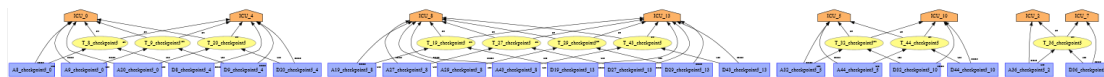


a) SCU provenance data at checkpoint 3

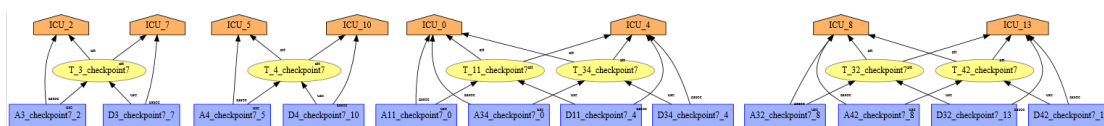


b) SCU provenance data at checkpoint 6

Figure 7.8: Provenance graphs for an SCU after execution of bag-of-tasks at two different checkpoints during run-time



a) Task (delegation) provenance data at checkpoint 6



b) Task (delegation) provenance data at checkpoint 8

Figure 7.9: Provenance graphs for an SCU after execution of bag-of-tasks at two different checkpoints during run-time

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<prov:document xmlns:prov="http://www.w3.org/ns/prov#" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SCUmanagement="http://www.infosys.tuwien.ac.at/scu/">
  <prov:entity prov:id="SCUmanagement:SCUDynamicProfile6">
    <prov:value xsi:type="SCUmanagement:SCUDynamicProfile6">properties[members 5 , productivity 0.9000 ,
      performance trust 0.7648 , socio-techn. trust 0.7330]</prov:value>
  </prov:entity>
  <prov:agent prov:id="SCUmanagement:SCU1"/>
  <prov:activity prov:id="SCUmanagement:Adaptation_6_1_SLAchanges"/>
  <prov:wasAssociatedWith prov:id="SCUmanagement:_6">
    <prov:activity prov:ref="SCUmanagement:Adaptation_6_1_SLAchanges"/>
    <prov:agent prov:ref="SCUmanagement:SCU1"/>
  </prov:wasAssociatedWith>
  <prov:used>
    <prov:activity prov:ref="SCUmanagement:Adaptation_6_1_SLAchanges"/>
    <prov:entity prov:ref="SCUmanagement:SCUDynamicProfile6"/>
  </prov:used>
  <prov:wasAttributedTo prov:id="SCUmanagement:_6">
    <prov:entity prov:ref="SCUmanagement:SCUDynamicProfile6"/>
    <prov:agent prov:ref="SCUmanagement:SCU1"/>
  </prov:wasAttributedTo>
</prov:document>

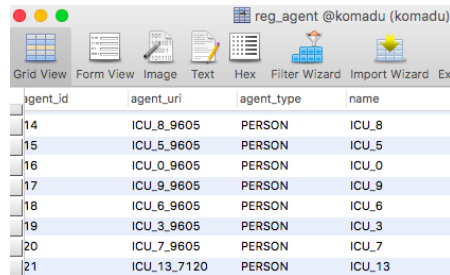
```

Figure 7.10: A sample xml file generated with ProvToolbox.

Figures 7.5, and 7.7; b) ‘the SCU context’ shown in Figure 7.8, which gives data about the SCU, such as number of members of an SCU within a given checkpoint, and the values of different performance and social metrics for the collective at each adaptation point, as well as the events that brought to SCU adaptation, e.g., task deadline thresholds or changes of an SLA requirement; c) the *Task context*, showing information regarding delegation activities, and information about which ICUs the task was assigned to initially, and to which ICU it was delegated to, which is shown in Figure 7.9. Looking at Figure 7.9, we can see that with provenance we can get information about which workers were included within a collective at a certain time point, and which tasks were assigned and delegated to which workers.

Looking at Figure 7.8, we can see that provenance can be utilized to show the values for different metrics regarding a collective at a certain point during runtime, as well as show events that happened at a specific point in time. For example, Figure 7.8 a) shows provenance data for an SCU after it was adapted due to time-thresholds for specific tasks, while Figure 7.8 b) shows provenance data regarding an SCU after it was adapted to changes of SLA values for specific parameters at runtime. The events are shown via the blue rectangles which denote *activities* in the PROV model, for example in Figure 7.8 a) the activity is denoted as *Adaptation_3_1_TimeThreshold*, which marks the event of the SCU1’s adaptation checkpoint 3 due to a time threshold, while in Figure 7.8 b) as *Adaptation_6_1_SLAchanges*, which denotes the event of SCU1’s adaptation at checkpoint 6 due to a change of an SLA parameter. These graphs are the concrete results of our adaptation algorithm where we simulate two scenarios during a single run-time of the same SCU, time-thresholds for tasks and changes of price to be payed for specific skills. Figure 7.9 gives provenance visualization from the context of tasks. More specifically we have chosen to track the delegated tasks at specific point in time, and see to which workers were the delegated tasks initially assigned, denoted with A in the Activity types, and to which workers were they delegated after they were not executed from the initially

assigned workers; delegations are denoted with D in the Activity types in the graphs. For example, in Figure 7.9 b, the Task with Id 4, was initially assigned to worker/ICU with Id 5 and then delegated to ICU with Id 10. Our experiments show that provenance can benefit social computing applications and systems, because it is a mechanism with which valuable information can be extracted to make management mechanisms effective. This information can help in constructing novel metrics. We inferred a few metrics from our experiments and we describe them in the next section.



agent_id	agent_uri	agent_type	name
14	ICU_8_9605	PERSON	ICU_8
15	ICU_5_9605	PERSON	ICU_5
16	ICU_0_9605	PERSON	ICU_0
17	ICU_9_9605	PERSON	ICU_9
18	ICU_6_9605	PERSON	ICU_6
19	ICU_3_9605	PERSON	ICU_3
20	ICU_7_9605	PERSON	ICU_7
21	ICU_13_7120	PERSON	ICU_13

Figure 7.11: PROV Agents in our experiments inserted to Komadu

Komadu experiments

We utilized Komadu [SZP15], an open-source Java-based standalone tool for capturing and visualizing provenance data, based on the W3C PROV standard, and developed by researchers at Indiana University¹. Komadu uses Apache Axis2 as a SOAP Web Services engine for implementing a provenance service. It was build after Karma, which is a provenance capturing tool based on the aforementioned Open Provenance Model (OPM). In addition to the fact that Komadu is based on PROV-O, while Karma on OPM, Komadu has addressed several limitations of Karma, such as upscaling provenance relationship types as well as enabling provenance data capturing from different sources and applications as provenance data in Komadu does not need to be connected to a single identifier like in Karma. Events in the form of notifications can be captured through Komadu’s inject API, and data can be queried through its query API [SZP15]. We deployed Komadu, with Tomcat 7 and MySQL 5.5 on a server of the t2.micro model² from Amazon.

The experiment we conducted with Komadu consists in evaluating processing time for provenance data-injection into Komadu. We performed a two-fold evaluation: a) measuring processing time of injecting data for two different checkpoints, which means two checkpoints with different number of provenance activities and, b) measuring the processing time for injecting data from different log sizes, one with 50 checkpoints and another with 2000. Figure 7.11 shows a screen-print from our experiments of how ICUs/People as Agents appear in a db when modeled and inserted with Komadu.

¹Komadu source code on github: <https://github.com/Data-to-Insight-Center/komadu>

²<https://aws.amazon.com/ec2/instance-types/>

The method that we used for our evaluation is the following: we mapped the data from our log files to PROV terms that match the way provenance data is inserted in a db with Komadu. Then, we programmed methods for injecting the mapped data which are read at runtime from the logs through the KomaduService to a db. Each injection is treated as an event, and our events are the following activities: task assignment, task delegation, and ICU profile metric updates.

For both log files we measured the processing time for each checkpoint, which includes data injected to Komadu regarding the aforementioned provenance activities: task assignments, delegations and ICU metric updates. As we have mentioned before, each processing of a batch-of-tasks might include a certain number of delegations from one ICU to another. In addition, because our scheduling algorithm adds or excludes ICUs at runtime, the starting number of ICUs at the batch assignment and the end number of ICUs at the end of a batch execution may be different. Consequently, the initial number of assigned tasks, the number of delegated tasks, and metric updates at the end of task-batch execution all contribute to the amount of data being injected into Komadu.

Examining the results of the processing time of data-injection about specific run of a task-batch, which means data regarding specific checkpoints, the results showed that the time is in the order of milliseconds. Table 7.1 shows the data-injection time from the log file with a run of 50 task-batches and 50 checkpoints, for a few checkpoints that we randomly chose to show. For the particular run, the results of which are shown in Table 7.1 we used 20 threads to define the KomaduServiceStub, however we tried with other thread numbers, and the results showed minuscule time differences between two threadpool sizes.

Next, we compared the total time to process 50 and 400 checkpoints. In particular, Table 7.2 shows a comparison of time considering the number of threads for 50 checkpoints, while Table 7.3 shows a comparison of processing time of data injections to Komadu with different threadpool sizes for a log file with 400 checkpoints. When considering the threadpool sizes we can see that the average processing times per checkpoint in both cases are similar. Moreover, comparing the total processing time for 50 and 400 checkpoints we can conclude that inserting even larger amount of data, 21780 activities as opposed to 2664, brings an acceptable processing overhead considering the amount of data. Hence, we conclude that provenance data and tools for supporting this type of data, like Komadu might be useful for utilization in social computing considering the variety and amount of data generated in social computing applications and scenarios.

7.4 Provenance-based Inferred Metrics for Social Computing

1. *Delegation-based Profile similarity.*

For the sake of the discussion, in this section we chose to analyze two ICUs from the the provenance logs of a specific run of our adaptation-algorithm, the ICU

Table 7.1: Komadu injection processing time of specific log-data

Checkpoint after a task batch is executed	checkpoint2	checkpoint12	checkpoint30	checkpoint49
Number of activities within a single task-batch execution	69	70	76	55
Processing time in ms	240	112	515	46

Table 7.2: Komadu injection data for 50 checkpoints, and a total of 2664 activities treated as events

Total Time	Average injection time per checkpoint	Threadpool size
6801ms	136.02ms	5
6412ms	128.24ms	20

Table 7.3: Komadu injection data for 400 checkpoints, and a total of 21780 activities treated as events

Total Time	Average injection time per checkpoint	Threadpool size
61432ms	153.58ms	5
59983ms	149.95ms	20

with Id 16, and the one with Id 22. In our experiment, every time tasks needed to be delegated from a particular ICU to another, the ICU to which the task was delegated was included in the SCU, this means that both ICUs, the one from whom the task was delegated and the one to whom it was delegated, do not provide the same skill type only, but also that they are close with their reputation scores; this is because we ranked ICUs based on their reputation scores initially, and tasks are delegated to the reserve ICUs in a FIFO order. Hence, provenance-data regarding delegations can be a good indicator for the fact that delegations can be used in metrics that define profile similarity between two ICUs. Let us examine our results in more details. Figure 7.5 presents the visualization of executed assigned tasks as well as delegated and executed tasks for ICUs with Id 16 and 22, in four separate checkpoints. Tasks were delegated from ICU 16 to ICU 22 during multiple time points of the SCU execution, as follows: two tasks in the first and the fifth checkpoint (Figure 7.5b)); three in the third checkpoint, the fourth (Figure 7.5a)), and the seventh (Figure 7.5c)); one task in the eighth, ninth (Figure 7.5d)) and tenth checkpoint; no delegated tasks in the second and sixth bag-of-tasks assignment. Due to the provenance visualization, this data can be easily inferred without queries,

which could be useful for business stakeholders. From the experiment results we can deduct that ICUs with Id 16 and Id 22 have the same skill type, because tasks from ICU Id 16 were only delegated to the ICU with Id 22. This also testifies for the consistency of ICU 22 in the sense that ICU 22 was only invoked in the SCU when some tasks from ICU 16 needed to be delegated. It means that ICU 22 was successful in executing the tasks delegated to it, and thus was invoked multiple times. Figure 7.6b shows the success rate of all ICUs at four checkpoints, which also proves our deduction from the provenance graphs, that ICU 22 executed all delegated tasks from ICU 16, as ICU 16 has a success rate that does not achieve a value of 1 (a value of 1 indicates that all assigned tasks were successfully executed), while ICU 22 has a success rate of 1 at each checkpoint.

Generalizing now, let us define $d(v,u)$ denoting the interaction intensity between v and u in the $[0,1]$ range, defined by the number of delegation relations between v and u across multiple checkpoints. If we denote the availability of ICUs with a_v , a_u and a_w with 0 if not available, and 1 if an ICU is available, and the reputation of all three ICUs with r_v, r_u, r_w , then we can consider the following relations to be valid at one single checkpoint: a) if a_u, a_w are both 1, and the value of $d(v,u)$ is within the $(0,1]$ range, while the value of $d(v,w)$ is 0, then we can safely assume that the reputation relation of v, u, w to be $r_v > r_u > r_w$, and b) if $a_u = 0$ and $a_w = 1$, and $d(v,u)$ has some value from the $(0,1]$ range, while the value of $d(v,w)$ is 1, then we can safely assume that the reputation score of u and w are close $r_u \approx r_w$, in addition of the validity of the $r_v > r_u > r_w$ relation, particularly when the worker pool is large. These conclusions are intuitive if one knows how ICUs are ranked, because w is ranked as the next appropriate worker after u . However, for analysts who do not know the details behind the mechanisms of ICU selection, it can be a valuable information. Moreover, this information is even more valuable when the worker pool with which an SCU environment works is comprised of workers registered on multiple platforms, and if analytics is conducted on Big-data.

Concretely, from our algorithm, we came to formulate a similarity metric based on reputation and skill. As aforementioned, for each task, we have a ranked list of reserve resources/workers with the appropriate skill, and they are ranked by their reputation values in a descending order. So, we define a similarity metrics based on reputation and skill-type, as in the following:

$$s_{vu} = \frac{rank_u}{m} * \frac{d_{vu}}{t_v}, \quad (7.1)$$

where d_{vu} denotes the number of delegated tasks from v to u , number of total assigned tasks to v , and m is the total number of workers in the reserve list. The rank value of the workers, starts with the value that represents the total number of workers in the list and continues in a descending manner. The similarity metric can have values in the range $(0,1]$, where a higher value means more similar profiles (similar reputation scores).

Hence, data-provenance regarding delegations can help in defining novel metrics for ICUs, and we demonstrated this by arguing that profile similarity with regards to

two parameters, ie. skill-type and reputation, can be defined by having a detailed overview of task-delegations.

2. *Active time in the social computing collective.*

The active time within a collective can be measured in two contexts: active time in the context of communication and active time in terms of performance, because high communication does not always indicate high-performance.

3. *Time-based reciprocity.*

Time-based reciprocity can be defined and calculated in two contexts as well, *interaction reciprocity* and *performance reciprocity*. We can define interaction reciprocity for a worker u , in respect to worker v as in the following:

$$R_{u,v}(\tau) = \frac{\sigma_{u,v}(\tau)}{\sigma_{v,u}(\tau)}, \quad (7.2)$$

where τ is a specific time period, $\sigma_{u,v}(\tau)$ represent number of replies from u to v , while $\sigma_{v,u}(\tau)$ represents number of communication requests from v to u . If we want to monitor reciprocity within discrete time intervals we assume that the frequency of updates within each τ is the same. For the performance reciprocity, the same definition applies, only the requests are platform automated requests for example, request for approval of an SLA parameter change, e.g., a runtime request to work on tasks with a different price, while the replies are acknowledgments from workers, e.g., accepting changes at runtime.

4. *Price per skill-type consistency.* Provenance data can track price changes for a particular skill, made by a worker as well as a customer. We can calculate the standard deviation of costs per skill type for workers, for every invocation in an SCU for a particular skill, over a defined period of time, and set a value of consistency which is a pre-set value in the plus or minus range from the most frequent cost value for an ICU for a specific skill. The variance can be defined as in the following,

$$C_{u,s}^2(\tau) = \frac{\sum(C(u,s)-\mu)^2}{m}, \quad (7.3)$$

where $(C(u, s))$ is the cost of worker u for tasks with skill s in the time period τ , μ is the number of invocations of worker u with skill s at the period τ , while m denotes the mean of cost per skill type within τ . Thus, the standard deviation is $\sqrt{C_{u,s}^2(\tau)}$. The price consistency of a worker for a particular skill then, can be calculated with the help of a pre-set range of values $[\alpha, \beta]$ α , and β delimit the variation from the standard deviation value. Hence, if the final value is within this range the price of worker u , for a skill type s , is consistent over time τ , otherwise the worker is not consistent regarding its own defined costs per skill type.

7.5 Privacy Implications: A Discussion

Current social computing systems in the industry all base their operations on sensitive personal data for verification of worker and customer profiles; with provenance this

sensitive data can be diversified and increased in volume, which we see as a serious issue in these systems and a challenge to be addressed. Thus, provenance, is a double-edge sword regarding privacy. It is by definition meta-data and meta-data can reveal considerable sensitive information. We see it important to list a few crucial aspects that we have identified regarding privacy issues that are induced from provenance-data in social computing. In the following, we argue privacy implications from multiple perspectives.

- *Participant context.* From the context of system participants there are two perspectives of privacy concerns: that of the workers and that of the customers. However, the needs for privacy for both type of participants interleave. From the workers perspective, they may not want information about customers, collaborators and projects to be publicly known. Customers on the other hand, whether they are individuals or organizations, in addition to sensitive personal information, may not want details about their projects and produced artifacts be publicly known. Provenance then is identified to be a major issue for privacy in the context of produced-data and not only of data connected to people profiles (e.g.,[DKR⁺11]). Thus, while provenance can help in identifying stakeholders of a project, the control flow of a project, who worked on which tasks, and other data about a collaboration, it can also be a point of risk. Some authors argue that provenance can help reduce privacy issues by providing data that help in authorization and access control mechanisms [CNB⁺12]. However, this approach tackles access issues and protection against access risks, but there is another aspect of privacy that we advocate for, which is: if a worker and/or a customer wishes not to be identified it is his basic human right not to be, and this should be reflected in system design as well. On the other hand, provenance can help to track where and how sensitive data has flowed, for example by a trusted third-party and thus, provenance can be used for privacy enforcement as well.
- *Behavioral data context.* As we have mentioned in Section 2 provenance can be used for interaction data as well as for performance data, which social computing systems use for different provisioning and management mechanisms. Consequently, while workers and customers may know the information that they are giving away while registering on platforms, they may not be aware about sensitive data that is generated and collected in an automated way, as these are usually not made transparent except in privacy policies. However, privacy policies are not as efficient as people rarely read them, and they usually register to platforms without the knowledge of how their sensitive data is collected and used.
- *Infrastructure context.* Large-scale crowdsourcing and inter-organizational collaborations, may use different infrastructures, for example Cloud systems, as well as a common decentralized provenance storage. In this case, sensitive data may be stored and processed in locations outside the collaboration-participants' physical locations, data may flow through multiple providers and operation centers, and this may allow certain participant parties to circumvent regulations and laws.

To approach the aforementioned issues, it is crucial, that provenance-based systems and in particular those including social computing, to provide mechanisms of *transparency* with which both workers and customers are informed about what type of data is collected and how it is used, to provide mechanisms of *informed consent*, and be *accountable*, which means compliant to laws and regulations, and these are only a minimum of mechanisms that should be considered.

7.6 Related Work

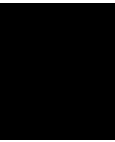
Markovic et al. in [MEC13] pose several research questions related to the utilization of provenance in social computing, mainly focusing on the assessment of worker trust, and selection of appropriate workers for tasks. In addition, by discussing a concrete scenario of social computing, they also argue that some concepts, such as task delegations and incentives are still problematic to express using Prov-DM [MM13]. A follow up work by the same authors presented in [MEC15] describe an extension to PROV-O [LSM13] and P-PLAN intended to benefit social computing scenarios. Data-provenance modeling for group-centric collaborations is presented in [PNS11].

The work presented in [HEV⁺13] discusses a method for analyzing provenance graphs based on provenance data for crowdsourcing, with which information regarding crowdsourcing activities can be assessed regarding the quality of work, through a trust metric. The authors have utilized machine learning methods with crowdsourced data to analyze the dependencies and links that can be inferred from a topological investigation of provenance graphs, such as number of nodes, edges, diameter, and build predictive models. They applied the method to assess the trustworthiness of crowdsourced data in a specific application built to crowdsource evacuation routes in emergency situations, named CollabMap ([RHVS13]).

Packer et al., discuss provenance for CAS in [PDM14] and argue that provenance data helps in making CAS more transparent and accountable, and also help in assessment of users' trust. The authors argue that provenance makes systems more accountable as provenance data provides information about the use of data and decision-making processes. The former, meaning the way user data is used by the system is also connected to systems' transparency, because users could be provided with a timely and detailed view of which type of data provided by them was used by the system and how. Moreover, provenance data may be utilized for enforcing privacy rules. The authors provide several use cases where provenance can be beneficial, including cases where it may give enough information so that the system users can be informed about the way their reputation is calculated, making a similar justification for provenance data, to one of our justifications in this work. Ways of how provenance can help in auditing compliance to privacy policies in IoT systems are introduced in [PSP⁺17], some of which mechanisms can be utilized in social computing as well.

Authors in [TBA16] have investigated whether existing provenance systems are capable of handling large-size social-provenance data and provided their own decentralized

architecture model that could better handle provenance-data as opposed to existing ones, in terms of scalability, data quality and privacy concerns. Investigation on provenance in group-centric collaboration is presented by authors in [PJN12], in which a model of provenance focusing on collaborations based on the OPM notation is presented. The work presented by authors in [BA17] discusses a large-scale synthetic social provenance database, designed for social networks, and how that provenance data can be used to define and calculate metrics such as credibility of a person. These type of databases can be useful to motivate research of large scale databases for more complex social-computing systems. A case where provenance data can be used in a specific domain, such as in crowdsourced data analysis tasks is discussed in [WGS⁺13]. Another domain-based example regarding provenance and related to geospatial data is given in [GGH14], where the authors mention processes with human-in-the-loop. We mention the two latter works as two of several existing works, which testify that provenance can also be used in providing a better Quality of Result, in addition to enabling more efficient management of task executions and resources/workers.



Privacy in Human Computation

While there is a solid amount of work on building human computation systems and mechanisms that support efficient management, such as task assignment and management (e.g., routing and delegations), worker management, incentive and payment models for workers online and quality assurance, little research has been conducted on the privacy implications in these systems. Privacy, however, is a human right and as such these users are also entitled to it. Article 12 of the Preamble of the Universal Declaration of Human Rights states for example that "No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honor and reputation." The design of human computations systems as well as everything else on the web, needs to be guided by privacy preserving principles.

The key contributions of this chapter are: 1) a discussion of why and how user data is utilized in particular human computation systems, 2) an analysis of user privacy-awareness on human computation platforms through the results of a user study that we conducted with an online survey, and 3) recommendations for human computation stakeholders, along with some research directions.

8.1 Personal Data on Human Computation Systems

8.1.1 Collected data

We investigated the collected data from some of the existing crowdsourcing/expert-labor market platforms (by actually creating accounts), such as Amazon Mechanical Turk, Microworkers, Freelancer, Upwork, PeoplePerHour, TopCoder and uTest. The information required to *build up a profile* and/or *verify a profile* sees variations from platform to platform. The following list provides some data required to build and verify profiles generalized through platforms; not every platform requires everything on the list, but everything on the list is required in various platforms.

- Full mailing address - Sometimes even documents are required to prove address, such as utility bills or bank statements.
- A government issued ID - Passport, ID card or Driving license.
- Photograph
- Code verification along with a user's face on a photograph - On some platforms workers need to send a photograph of themselves where they hold a piece of paper with a code provided by a platform written on the paper.
- Educational experience - On some of the platforms filling out at least one educational experience is mandatory.
- Job title - On some platforms filling out a job title is mandatory.
- Bank account information
- Data from mobile-sensing - Depending on the application domain, other sensitive data may be collected at runtime while users are working on tasks or just wearing a smart device. For example in mobile crowdsourcing applications, such as crowd-sensing, location information, health information (that users share through wearables to applications) and other data may be collected which can be used to identify and profile a user.
- Device and connection data - Basic system fingerprinting such as IP address, browser type and operating system.

The General Data Protection Regulation [Reg16] (with the enforcement on 18May 2018), stipulates that personal data is "any information relating to an identified or identifiable natural person ('data subject')". Moreover, identification is the singling out of an individual within a dataset [Par07], even if his or her name or other attributes that we typically associate with an identity remain unknown. Consequently, most of the aforementioned information can be used to identify a person and that means this data is personal and thus should be kept private.

8.1.2 Reasons for collecting personal data

The General Data Protection Regulation (REGULATION (EU) 2016/679) [Reg16] defines profiling as "any form of automated processing of personal data consisting of the use of personal data to evaluate certain personal aspects relating to a natural person, in particular to analyze or predict aspects concerning that natural person's performance at work, economic situation, health, personal preferences, interests, reliability, behavior, location or movements". In Human Computation, the collection of user data and building user profiles is usually not conducted for big-profit purposes (such as selling user profiles to advertisers) but for designing and developing mechanisms for effective work-management on human computation provisioning systems. In the following we discuss some of these mechanisms.

Task-Assignment and Formation of collectives

A number of existing works have presented (expert) discovery and ranking algorithms for service-oriented architectures with human-provided services ([SD10], [DT12]), as well as non-service oriented systems in which tasks assignment is based on qualifications. Research in team formation in expert networks is also being investigated ([ABC⁺10], [DD10]). These mechanisms are all based on logged and historical data about workers to come up with the appropriate workers for specific tasks in individual-crowdsourced work or specific collective-work.

Management Mechanisms

Managing individual and team-based worker performance in human computation is also important for complex systems with automated processes even for the human-in-the-loop coordination. Adaptation mechanisms in human computation within SOA for example, are discussed in [SSPD11], and an adaptation mechanism for elastic collectives based on a trust model is presented in [RTD15]. Algorithms that calculate worker's performance can be used within delegation mechanisms, where a task is delegated to another worker that may or may not belong to the initial collective. Consequently, a new worker can be added to an existing collective at run-time. Research on both task assignment and adaptation algorithms that include measurement of worker performance is in big part based on *trust* and *reputation* models. Some of these trust and reputation models include not only metrics that can be measured automatically (such as task success-rate) but also *social trust*, which mostly is defined and calculated as a trust-score that is given to a worker by collaborators or acquaintances and/or by work-requesters/clients based on their satisfaction by the results. Social trust is often subjective and in most cases requires that the person rating a worker knows the worker personally, so in some trust and reputation models worker identities are needed to be known. Hence, in some cases reputation mechanisms are in conflict with privacy, and we need to find ways of bringing these two concepts together and provide privacy aware reputation strategies.

Quality of Service

Keeping quality at a desired level also requires monitoring of workers. (see an example of SLA based QoS monitoring for crowdsourcing in [KPSD11]).

Misbehavior prevention

Personal data is sometimes used in building mechanisms for preventing worker misbehavior, such as *Sybil attacks*- cases in which workers can easily create multiple profiles and make more gain by executing the same tasks multiple times.

Incentive Mechanisms

Developing and applying appropriate incentive models for workers based on monetary and non-monetary gains is also based on data collection, and sometimes personal data, for example when reputation is used as an incentive. (See work on incentives and rewarding mechanisms in social computing in [STD13].)

Payments

In most platforms that have implemented monetary incentives, user bank/credit account is required so that workers can be paid. However, some platforms allow users to delete this data after they withdraw the required amount from it (such as microworkers.com, in the case of requesters/clients). This should be a standard practice for all data types as well. A regulation example for this is the GDPR's *Right to erasure* described in its article 17, although we are on the opinion that human computation platforms should consider the provisioning of mechanisms with which subjects would have the possibility to directly modify, update, or erase data without making a request to the controller (the owner of the crowdsourcing platforms), just like microworkers allows the deletion of bank account information without requests.

All the aforementioned mechanisms require monitoring of workers. However, in research frameworks and systems, no collection of personal data is mentioned explicitly, and the privacy aspect is not tackled, except in specific research presented in Section 2. On the other hand, almost all existing platforms in industry require users to share their personal data. Thus, we advocate for all the aforementioned mechanisms to be considered together with their privacy-related implications. In the next section we examine privacy implications by analyzing a few risk-factors that we identified as most relevant.

8.2 Privacy Risks

User Privacy Policy Awareness

Users usually agree to Privacy Policies without actually reading them, and they do not read them because they are too long or too complex to understand ([WAW15]). This contributes to their use of platforms without being aware of what they are entitled to, regardless of whether their privacy rights are violated or not.

Lack of Transparency in Privacy Policies

Very often, users are not given complete information about what is considered under *personal information*, about how their personal data will be used, whether it will be shared with third parties and for how long will this information be stored on a human computation platform-provider's servers or on the servers of their service-providers. Often, the use of personal data is defined in policies in a vague way. Consider for example the statement "we may share certain data...", using words such as "certain data" without

concretely defining what type of data the statement is referring to means getting a clear consent by users to use whatever personal data of users that the provider owns. Many privacy policies that we have read also contain phrases such as the following: "we may share information with third parties for industry analysis, research and other *similar purposes*"; the terms "similar purposes" give the providers the freedom to use personal data in any purpose fitting their needs, without the explicit consent of the platform users. In addition, consider this statement: "we may use your personal information from other services and connect to your account information when necessary", this is clearly a way to de-anonymize users even if the platform is designed to use pseudonyms, as combining various data-sets (e.g., by email addresses) de-anonymizes users. Selling user profiles to the ad industry is also a possibility, although many human computation platforms do not have an ad-based business model. However, companies called *data brokers* continuously collect data about people from multiple online (and even offline) sources and sell that data to clients in various business domains. Data that they collect can be also retrieved from websites with log-ins and visits, such as *browser* and *device fingerprints*. Because privacy policies are not straightforward, people can not be sure whether data brokers are not leveraging some data from human computation platforms as well.

Profiling

User information is collected through information that they share with platforms as well as by automatically collecting data by tracking. Among other uses such as computing reputation scores, this data may also be utilized to group people in different categories, by various contexts (e.g., country of residence, gender). This may be used to set up rules for task-assignment, and discriminate certain groups during task-assignment and rewarding. As we mention in our study results, there are cases where workers from certain countries are rewarded less than others for the same tasks. In addition, consider for example a real danger through crowdsourced tasks for online-monitoring of certain locations with the purpose of detecting and reporting criminal activities, or even a different setup, such as identifying wanted offenders from a set of pictures that are posted in a crowdsourced task online. If these platforms are profiling workers, criminals might find ways to identify workers, and workers' lives could be put at risk. Furthermore, if information from political crises-response crowdsourcing sites¹ about people reporting incidents in war-struck areas fall into the wrong hands, it might also pose risks for the reporters. These may seem as extreme examples, but serve well to prove privacy concerns. On the other hand, many companies hide behind the term "anonymity", for example they do not require real names and allow people to register with pseudonyms while collecting other personal data, in this way wrongly convincing people that they actually work online without being identified. For example, authors of [CS16] cite a study revealing that the combination of zip code, gender and birth-date data had been individually unique for around 216 million US citizens, and consequently citizens can be identified without any other additional data. In the same work, authors also cite another study showing that four data points, such

¹See for example: <https://syriatracker.crowdmap.com/>

as four sets of time and location data could be used to uniquely identify people. Thus, leaving out some data while collecting other type of data does not mean that anonymity is achieved, and in most cases people are not transparently informed of this fact.

Lack of Control

In current systems, users do not control how their private information is used (whether it is shared, sold or misused), and have no control over who accesses that information. They have to be content with what they read on privacy policies (when they read them). They are not given control to their own data, to update or delete their data when they want. Moreover, stored information is sometimes not secured enough, rules and regulations are not always respected, and data stored on foreign servers belonging to a different jurisdiction than a person's residence country (over which the user may not be given a choice), can be misused (e.g., due to security breaches, unencrypted data, unethical employees or security agencies).

Lack of Ownership

Users do not own their own data that they have shared with the platforms, rather platform providers do. Similar to the risks in not being able to control data, not owning data means intentional or unintentional sharing with third parties, access to user data by unintended parties as well as transferring/selling user information to other parties and monetizing people's data, which sometimes can be done without users' knowledge and approval.

Lack of Security

Last but not the least, in addition to the aforementioned factors, security is of paramount importance for protecting privacy. Data control and ownership do not have any effect on privacy protection if user-information is unencrypted. Needless to say, security protocols for securing internet connections and data encryption protocols should be standard features that every human computation platform should support.

8.3 Study

8.3.1 Method: Survey Design and Distribution

We conducted a study to assess user privacy-awareness in human computation with an online questionnaire, hosted on a server at Technische Universität Wien/TU Wien. We asked participants a series of questions that we designed specifically to get their opinion on their private data collected and utilized on the platforms, to get their knowledge on privacy implications on these platforms as well as their concerns.

We disseminated our survey in two ways: 1) by sharing it with fellow researchers and colleagues by email, and with acquaintances and friends on social networks by asking

them to fill it in (if registered as users on these systems) or send the survey to people that they know are using these systems as requesters or workers; and 2) by creating a task/campaign at Microworkers (<https://microworkers.com/>) and a HIT batch on Amazon MechanicalTurk, asking workers to fill in our survey (at a given link). We did two rounds of the survey, as we came up with some more questions that we saw relevant during our study. Thus, the first round of the survey had 20 questions, 16 of which were designed to assess user privacy-awareness, and 4 were statistical questions to get demographic data. We submitted this survey on Microworkers and to researchers and freelancers through private communication, while the second round of the survey had 5 additional questions, and was conducted only on Amazon MechanicalTurk. Where we have less participants for the newly added questions we mention it when discussing the particular questions.

Microworkers has a design that allows requesters to select what user-base to choose depending on worker country-information and makes payment recommendations according to country-dependent ratings. We created four tasks/campaigns and asked users from four different country-groups to fill in our survey. For the first three groups of workers, we paid workers \$0.42 per task, as by investigating other studies on these platforms (that have used payments between \$0.10 and \$1) we concluded that this amount was enough for people that are interested on the topic to accept the task and low enough to discourage misbehavior by those who may want to fill in the survey without interest and spam the results. In spite of recommendations for lower payment for a group of workers coming from countries rated lower, we decided to pay workers of a lower rated group of countries the same amount of \$0.42 and not lower. However, our survey-task for a fourth group of workers residents of high rated countries was rejected for that amount as the minimum payment was \$1.00 per task for surveys such as ours, so we paid that amount to get our survey completed by a fourth group, as we needed different demographics for the sake of the study². In this regard, we strongly encourage ethical payment methods by crowdsourcing platforms and ethical payment behavior by work requesters/clients. Some of these mechanisms could be, *equal pay per task-type* for all workers (regardless of country ratings), or individual worker payments based on *quality of results*. Nevertheless, this side-experiment allowed us to qualify the answers that were submitted in reply to our survey, for example filling in optional questions that required more elaboration. In this context, we noticed no difference in the answers of higher rated countries compared to lower rated groups. In fact, some users that were paid less gave elaborated answers while none of the higher paid participants did this. Workers on Amazon MTurk were paid \$1.

To filter submissions as well as to avoid spammers and malicious users who fill in the survey without reading the questions and answers, we included a few questions that helped us assess (to some level) the honesty of participant answers. One such "testing" question was added to check a related "yes or no question", the testing question included

²Due to ethical principles, we do not reveal here which countries are rated lower or higher on microworkers.com.

Table 8.1: Demographics of participants

Education	Percentage
Primary School	1.97%
High-School	20.6%
Undergraduate studies/BSc/BA	40.6%
MSc/MA/Specialty training	15.6%
Dr/PhD	6.86%
Postdoctoral researcher	1.4%
Other	12.7%
IT Knowledge	Percentage
Expert/Professional	20%
Medium level (good IT skills but not expert/professional)	59%
Knowledge to get around online	21%

Table 8.2: Most common collected data

Collected data	% of users who want to hide the data
Name and Surname	24%
E-mail address	22%
Phone number	61%
Birthdate	26%
Photograph	54%
Location/Mailing address	35%
Utility bills	53%
A government issued ID	68%
Bank account	54%
None of the above	1%

radio buttons with more elaborating statements to be chosen if the user answered with "Yes" on the related question, and included a radio button with the statement "I answered with "No" on the previous question" to be selected if a participant has answered with "No" on the related question. In addition, we added a "Yes or No question" asking survey participants if they have read our consent form for the survey and excluded submissions of participants who answered with "No". A few participants had filled our survey multiple times. We counted only one submission from these participants and excluded all duplicates.

8.3.2 Results and Analysis

Demographics

We had a total of 204 participants, the answers of three of which we excluded as a consequence of their negative answers to the question with which we requested participant consent for the survey (through reading our consent form). 105 participants were workers from Microworkers, 78 from Amazon MTurk (engaged in the second round of our study), whereas 21 were participants that we contacted by mail/social networks. Most of our participants were residents of US, EU countries and India.

Table 8.1 shows the level of education and IT knowledge of our participants. Participants with a PhD and Postdoc level of education were users of human computation platforms as requesters of work (for research purposes).

We asked participants to fill in the names of up to three platforms that they use and we got the following variety of platforms as answers: Microworkers, Amazon Mechanical Turk, Upwork, InnoCentive, Elance, Guru, 99designs, CrowdFlower, clickworker, RapidWorkers, ShortTask, Testbirds, cashcrate, fiverr, scribie, TranscribeMe, foulefactory, ideaCONNECTION, and OneSpace. Most of our participants worked on two or three platforms.

Privacy Awareness

We posed three type of questions assessing user privacy awareness and concerns: the first was related to a) *data collection, control and ownership of data*; the second was related to b) *anonymity online* and the third group of questions was related to c) *regulations and policies*. In the following we discuss the results.

a) *Data collection, Usage Concerns and Security* Regarding data collection, we asked participants to state their level of concern regarding the fact they they have to share sensitive data to register and verify their accounts. The *level of concern* question was set up as a 1-5 Likert scale. 19% of participants stated that they are somewhat concerned with the information that they are obliged to share when they register on the platforms, while 25% were not concerned at all. Table 8.3 gives a more detailed overview of the answers regarding participant concerns over the collection of their personal data. In addition, to get more detailed information, we listed some of the data types (mentioned in Section 8.1) collected by platforms and asked participants to select which of the given data type they would want to hide. They could chose multiple types. Most of the participants answered that they would prefer hiding: **a government issued ID** (68%), **bank account information** (54%) and **phone numbers** (61%). More details regarding the results for this question are given in Table 8.2.

Next, we asked participants if they ever provide false information when registering and creating their profiles and 17.7% reported that they do. Regarding the reasons for providing false information, 19% reported that they do not feel comfortable revealing

Table 8.3: Data concerns

Statements regarding data concerns	Very	Somewhat	Neutral	Not concerned	Not concerned at all	Likert score
How much are you concerned with the personal information that you are obliged to share so that you can register on these platforms?	19%	8%	39%	9%	25%	2.78
How much are you concerned with the personal information that you need to share so as to build your profile and verify your identity on these platforms?	19%	22%	25%	12%	22%	2.83
Are you concerned that your information will be misused (by the /platform that you are registered with)?	6%	11%	23%	28%	32%	2.16

some specific information about them, and 3% reported that they provide some false information in order to create secondary accounts.

Users' knowledge on where their data is stored is important for assessing their privacy awareness; hence, we asked participants whether they know and whether they are concerned if their data is stored on platform providers' own servers, or if platform-providers utilize cloud services (in which case an agreement should exist between platform providers and cloud providers for protecting user information and not sharing user data with other parties), and if data is stored in a location with a different jurisdiction (in which case different data protection regulations exist). Participants were given three answer choices and they reported as follows, 32.29% said "I admit I have never thought about these things and frankly I am not concerned.", 33.86% chose "I admit I have never thought about these things but I became concerned now.", and 33.85% answered with "I have thought about these things and am concerned."

Table 8.4 shows security-related statements that we added for the second round of the survey and the replies from 78 participants recruited from Amazon MTurk.

b) Anonymity Anonymity is of course fundamentally different from privacy. Privacy means that people *may* be identified online, but it should be their choice regarding how much and in what way their data is shared and utilized. Nevertheless, the two concepts are invariably related. Hence, we examined opinions on anonymity as well, and asked participants what would be the reasons in the case they prefer to work anonymously. Some workers that are working on more complex tasks (e.g., projects such as those posted on 99designs, Freelancer) and not on micro-tasks stated that they would not

Table 8.4: Security related survey statements

Survey statements regarding security	I strongly agree	I agree	Neutral	I disagree	I strongly disagree	Likert score
Having in mind that not only my personal information but also content that i produce or expect as a result from engaging on a microwork platform is sensitive, I expect the platform I perform micro-work on, the prove on a regular basis (every three months?) that it is secure, by having an independent pen-test performed and have the results published.	15.06%	24.68%	36.98%	9.59%	13.69%	3.18
"I would like to have the option to receive payouts in a privacy-friendly cryptocurrency." Please select a choice from 1 to 5, 1 indicating that you are not concerned with secure and private payouts, 5 indicating that you strongly agree with the statement.	10.96%	20.55%	32.88%	10.95%	24.66%	2.82

prefer to work anonymously online, using statements such as "working anonymously is not effective". Consequently, we assume that these workers do care about reputation as reputation mechanisms bring more clients and work. However, some replied that they would want to work anonymously in cases if they are working on some projects on which they would not want to put their name on but they are well paid, and communication and collaboration with clients is satisfactory. In addition, one participant answered that it would be nice if users are provided with the option to work anonymously online whenever they chose to (opt-in/opt-out).

On the other hand, most of the workers who work on micro-tasks answered that they would want to work anonymously for several reasons: they do not want their name to be associated to the type of work they do, to protect their banking information, they do not want the companies with which they work full-time to know that they are doing a side-job. The benefit of our survey was exactly in getting a variety of opinions and concerns from individuals, apart from the statistics. We quote some answers stating other contexts of concern for anonymity: "I would want to work anonymously so there was no bias towards me based on my demographics and/ or social class. I also would prefer to remain anonymous in case scammers entered the platform pretending to collect data, but instead, they were going to participants homes etc.", "I like minimizing my digital footprint as much as possible", "When doing microwork online, you do work for various people, potentially over dozens of people a day. I'd rather not have my sensitive information potentially available to all of them, when I'm forced to provide demographic information for much of the work anyway", "I'd not want to have that information

available for marketers or to be available to be sold. I'd not want other organizations to be able to access such information and use it to send me ads or other materials", and others. In addition, some answered that they would want to work anonymous because they do not want their earnings to be reflected on their taxes. Related to the latter, one participant stated that he uses foreign money transfer services, such as Payoneer, to avoid taxes for online work.

Furthermore, some participants stated their concern of their information being leaked to other parties. One particular participant stated that the reason he would want to work anonymously is that he can not be certain by whom and how his private information will be used, he added the statement "I want to control my "web" identity as i want".

Thus, we can conclude that workers doing complex tasks are more inclined to identification than workers executing micro-tasks that are easy to execute. Lastly, an interesting answer we encountered was: "I would want to protect my privacy", even though the specific question was related to anonymity online. Consequently, participants associated anonymity with privacy.

c) Regulations and Policies To asses participant engagement in privacy issues we went a step further and asked them if they read privacy laws, directives and policies. Figure 8.1 provides their reports. Interestingly enough, more than 50% of participants reported that they do read privacy policies when they register on platforms. However, around 40% participants reported that they do not read them. We take this result to be truthful as the number of participants is small. However, we assume that this ratio is significantly in favor of participants that do not read privacy policies because of complexity, as existing research suggests (e.g., [Klu06], [WAW15], [MRKC09]).

In order to get participants' opinion on platforms, research and regulations on privacy we included a question with a five Likert-scale agreement levels to chose from, for a few statements that we compiled. Most participants agreed that existing platform providers need to be more transparent about how they use personal information and they also agreed that research and industry should increase their efforts in providing mechanisms that will enable people to control and even own their data. For every statement we also asked participants if they are knowledgeable on the topics that the statements refer to or not. In total 57% answered that they need more information on the topics, 38% answered that they have knowledge on the topics and the rest did not answer. Detailed results are given in Table 8.5.

The questions from the first round of the survey can be found in Appendix C.

8.4 Suggestions

8.4.1 Recommendations

For platform providers, an important privacy-respecting guide in storing personal data is to only store that which is essential to the needs of the platform. With this, we mean

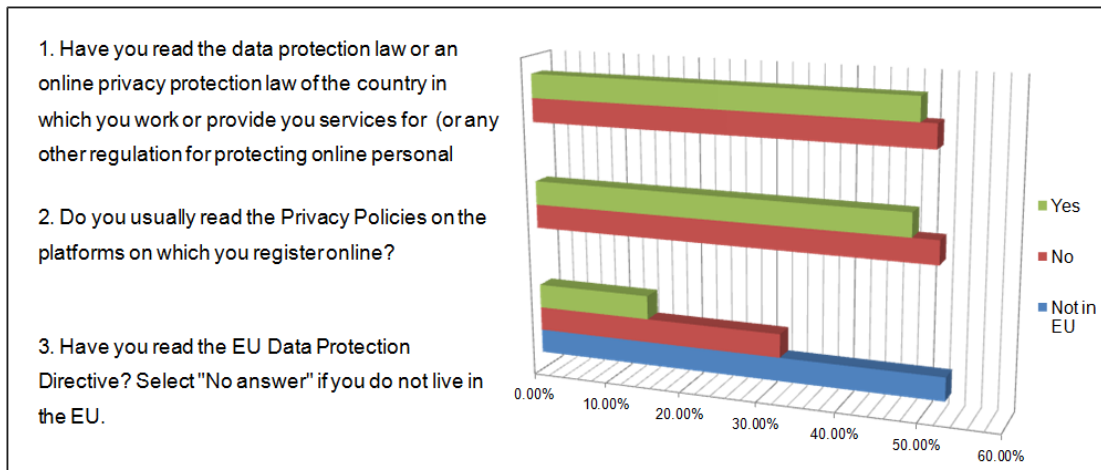


Figure 8.1: User reports on regulations

Table 8.5: Opinions on regulations, and approaches in research and industry

Survey statements	I strongly agree	I agree	Neutral	I somewhat disagree	I disagree	Likert score
Human computation Companies/Platforms should clearly state under which country/state law they operate.	28.13%	41.67%	25%	2.6%	2.6%	3.9
Companies/Platforms that enable and provide human computation should be more transparent about how they use my personal information	38.54%	29.67%	21.36%	7.31%	3.120	3.84
I am concerned about the privacy regulations/laws of the country in which I reside and work.	15.1%	30.73%	31.25%	18.23%	4.69%	3.3
Research and industry should increase their efforts in giving users more control over the use of their data.	30.73%	38.02%	19.79%	6.77%	4.69%	3.83
Research and industry should increase their efforts in enabling tools and mechanisms that will enable users to own their own data, in contrast to current standards where companies own users' data.	29.68%	33.85%	28.13%	4.17%	4.17%	3.81

the concept of data minimization; if n data points are enough to perform the task for which these points were collected, do not collect $> n$ data points. In the most general terms, according to Colesky et al., in [CHH16], there are two directions of strategies to protect the privacy of clients: data oriented, and policy oriented. These two directions lead to eight high level strategies that can be applied to the collection of data in ways that respect the privacy of data subjects.

An example authentication mechanism of identity protection is Attribute Based Credentials (ABC) [KKAH14]. ABC provides the client with a secure container for specific attributes, such as "over 18" or "holds a PhD in computer science", to use for specific cases, and the control over which attributes are shared is entirely in the control of the user. The container is called an attribute-based credential [KKAH14]. The benefit of the ABC mechanism, is in the fact that an attribute can not be tracked, because the mechanism is based on zero-knowledge proofs; thus, each time that the same client needs to prove the same attribute (e.g., birthdate, age, citizenship) it can not be determined whether the proofs of credential ownership for the same attribute came from the same individual.

In addition, there are some well-known anonymization methods such as the k -anonymity model, presented in [Swe02], and t -closeness described in [LLV07] that prevents attribute disclosure, which are regularly mentioned as one of the methods of dealing with sensitive data. However, these are also proven not to be reliable as sensitive data could be easily inferred from one or two data-points. The author in [Owm] discusses anonymization techniques (such as release-and-forget, access control, and audit trails) as well as re-identification techniques. The author concludes that re-identification techniques are much more solid than anonymization techniques, giving the examples of AOL and Netflix [Sog07] re-identification cases from their publicly released anonymized data.

However, the aforementioned strategies do not solve all the practical problems, such as payment methods. A person still has to provide at least some personal data in order to receive a reward for his work. In this context, one could argue that cryptographic currencies such as BitCoin could be used to pay rewards, but these too have been shown not to be entirely anonymous [AKR⁺13]. In addition, there are other alternatives of online services that offer means to transfer funds, such as CashU, and Perfect Money, which could be used as a "Trusted Third Party" for transactions between a bank account to them, and then transfers between the service itself, and the service and other services. Although tracking with these services is more difficult, it is still possible, as there is a specific party that still needs to know enough to be able to transfer funds.

One way of providing for giving control to users over their own data is the provisioning of human computation through decentralized systems. (One channel type for communication (and content creation) only could be provided with the communication protocols such as ActivityPub, the Diaspora protocol, OStatus which are used for social networking and microblogging platforms.) A recently published data-sharing protocol for decentralized systems worth looking into is Dat [OMM].

As far as we know, no human-computation platform has implemented multiple privacy-preserving technologies yet, still relying on cheaper, faster, and easier management methods that (might) erode the privacy of clients.

The following section discusses a few possible research directions.

8.4.2 Research directions

Transparency with rules or SLAs

System designers, developers and business actors need to come up with more transparent and direct ways of getting user consent (other than the current standard of publishing privacy policies). An interesting open challenge that we will tackle in our future work is our idea of enforcing user consent through Service Level Agreements (SLAs). Depending on the type of (human) tasks and whether their execution can be monitored and measured, introducing SLAs may come as appropriate as a mechanism to monitor, manage and adapt human computation collectives.

In relation to the aforementioned SLA application, a possible research direction is investigating the inclusion of privacy clauses (e.g., from privacy policies) in SLAs so that users will be obligated to read them and give consent when negotiating SLAs. This could be a two-way negotiation, employers could regulate personal data and content/artifact privacy in relation to the workers as well as the system, and workers could regulate their personal data in relation to employers, other workers and the system.

Privacy preserving workflows

Human-based computing in general and crowdsourcing in particular, in addition to issues with personal information, have issues with sensitive artifacts and data submitted for tasks. People who submit tasks may want to reveal only a part of the data. Thus, the design of workflows that provide enough knowledge for workers to be able to execute tasks but do not disclose the full context of requesters' work/interest, is an open (domain-dependent) research question (example work presented in [SFC⁺17]).

Payment methods

When people work in socio-technical systems individually and do not belong to an organization, even with the most efficient anonymization methods, e.g., on the assumption that all worker data is private, the payment methods are still an open question as they can be still used to identify a person.

Location

Let us assume a person consents to his/her location data being collected, e.g., in a crowdsourced traffic management of a city. In this case, developers need to pay attention for example to set some location checkpoints, which would not be used to infer sensitive

information, such as for example religion (checkpoints near religious buildings), hospitals, houses, and other institutions.

Evaluation methods

An interesting research challenge are also evaluation methods for software, i.e., evaluating the included privacy-preserving mechanisms.

Raising people awareness about privacy

Methods and techniques for raising awareness on privacy should not be a question tackled by experts working on social and legal areas only; it is crucial that computer science researchers approach these challenges as they develop software and disseminate their research. This chapter provides people opinions on privacy related issues for concrete platforms, and also provides a discussion of possible mechanisms for preserving privacy, and possible research challenges. Nevertheless, the last section is not an attempt to provide an extensive list of tools, strategies and problems; the goal of this chapter is to point to privacy challenges and provoke and motivate researchers of human computation systems to tackle them. The next section, discusses selected related work.

8.5 Related Work

Smith et al. have studied individual privacy concerns in organizations and have identified multiple dimensions of these concerns that they have presented in [SB96]. In that work, the authors list four factors critical to consider when assessing user privacy concerns: concern over data *collection*, *errors* in user data, *unauthorized secondary use* of user data (e.g., when data is used for other purposes than stated), and *improper access* to user data. Since then, a number of other models, frameworks of user privacy concerns have been presented, (some building upon the work of Smith et al.) such as [MKA04]. On the other hand, theories and models of how to control privacy and enforce privacy-aware mechanisms are also present in existing literature. Authors in [MB09] present such a privacy-control theory and discuss ways of its application in online environments.

Fischer-Hübner and Martucci in [FHM14] have inspected privacy implications for Social Collective Intelligence Systems (SCIS) by presenting an overview of the European Data Protection Legal Framework and relating the privacy rules provided by this framework with SCIS systems and mechanisms supported by these systems, such as user profiling for reputation scores, incentive models, as well as data provenance. Reputation, incentive models and data provenance are all listed as risks for user privacy as these mechanisms are inherently designed to work with user profiling. However, the authors list and discuss some of the available tools and technologies that can enable SCIS platform providers to respect and preserve user privacy, mainly via pseudonyms and anonymity. One example is by allowing users to use different pseudonyms for different roles, i.e., context-based pseudonyms that can be used only once per role (e.g., skill type) and thus

prevent misbehavior by malicious users. In addition, the authors describe anonymous credential protocols that can be used to create new credentials whenever a user wants, with less or different certificate attributes, which cannot be linked to an original certificate by the verifier and the issuer. Moreover, Fischer-Hübner and Martucci also present Privacy Policy Languages (such as PPL) with which platforms can make negotiations and come up to an agreement with platform-users on *how, by whom* (and *what*) data can be accessed, processed and logged.

Motahari et al. in [MMHJ07] have listed privacy threats in ubiquitous social computing by underlining the social inference threats in social computing, where a user can be identified for example through contextual information (e.g., location) or social links. Authors in [GGF14] present a privacy model together with a framework for task-recommendation in mobile crowdsourcing. The model is based on enabling workers to share information (e.g., location) with a recommendation server by choosing how much and what type of information they wish to share. Task recommendations are based on the information shared by workers. However, authors conclude the obvious, namely that achieving a high efficiency in task-recommendations means low level of privacy. Another work that presents a privacy-aware framework is presented in [TGS14] by To et al. The authors present a task-assignment algorithm that preserves location privacy for mobile spatial-crowdsourcing tasks, that is, for tasks that require workers to be at a specific location. Toch in [Toc14] investigates privacy preferences of users in mobile context-aware applications through crowdsourcing and presents a method to calculate user privacy tendencies. He suggests building distributed systems to tackle privacy risks (with the computation of user privacy tendencies being executed on the client side). Privacy preservation in decentralized systems is discussed in [BCKS13]. Privacy activists also advocate for decentralization and zero-knowledge systems, Balkan for example has written the Ethical Design Manifesto [Bal], which states: "Technology that respects human rights is decentralised, peer-to-peer, zero-knowledge, end-to-end encrypted, free and open source, interoperable, accessible, and sustainable."

Langheinrich in [Lan01] discusses some Privacy by Design principles, focusing on ubiquitous systems. He states that the Principle of Openness, or Notices, is an important principle during data collection. Users have to be informed when they are being monitored. In addition, Langheinrich discusses that consent should be required in a more flexible way than the "you can use our services only if you consent to our terms/policy". Users have to be able to use services while opting out of unwanted features.

Conclusions

This thesis is the results of our investigation and work on metrics and metric models for managing individual and collective human-based resources and services, as well as run-time adaptation algorithms for human-based resource collectives that utilize them. The models and mechanisms we describe are thought as mechanisms to be included in human computation applications/platforms within socio-technical systems. Thus, they are all intended to build better, more efficient, more complex and more ethical adaptive platforms that would manage online collectives, beyond the mechanisms of current existing platforms. After an investigation of current applications and systems, such as crowdsourcing ones and online collaborative platforms, as well as existing research work on managing collective intelligence, including task-assignment, worker selection, incentives, rewarding and management mechanisms, I identified several research questions which I addressed in this thesis. More specifically, this thesis contributed with the following:

1. novel *metrics* that reflect the nature of people, both performance based metrics that are monitored and calculable automatically and social metrics such as social trust, which are subjective but are calculable through voting mechanisms
2. algorithms for collective-*formation* and elastic *adaptation* of collectives, and their implementations as proof-of-concept experiments, which show how our metrics influence the adaptation of collectives at runtime and in turn how runtime adaptations affect the update of individual and collective metrics
3. elastic negotiation(SLA)-based *management process model*
4. a *provenance* model and ways of utilizing provenance in social computing
5. a discussion of ethical and *privacy issues* in social computing platforms, based on a survey we conducted on a crowdsourcing platform, and recommendations.

We believe that the utilization of the metrics and mechanisms presented can contribute in the development of further models and mechanisms that will increase the productivity

of people online, their collaboration and thus the quality of the provided artifacts. Our adaptation mechanisms advise for interactive management mechanisms between all stakeholders, the customers, the platform/system and online workers (human-based services) during task execution, in case of events and in case of requirement changes at runtime by both customers and workers.

Several open research questions connected to this work are open for future investigations, some that we have identified are:

- Setting aside organizations with known employees that automate their processes and outsource parts of them (e.g., enterprise crowdsourcing), there are open issues regarding platforms that use pools of workers where their qualifications need to be verified. Current qualification tests in crowdsourcing environments for example are not solid enough, whereas in other platforms for more complex tasks (e.g., language translation, web development), qualification mechanisms do not even exist and qualified people that register online may not get tasks for a long time unless they are in the few ones available. Thus, benchmarking is an open issue.
- The design of complex task so that their assignment and their result aggregations can be easily automated is another question, which is mostly domain dependent.
- Privacy issues are crucial for platforms in human computation in general, particularly for location tracking such as in smart-city applications, for payment methods, and finding reliable ways to identify a single human resource from a pool of resources without identifying him/her personally, which is important to avoid misbehavior such as multiple account creation and system tricking by workers, etc. To generalize it even more, researchers could think of questions raising ethical and legal issues.
- A very interesting open question is coming up with mechanisms that would allow the management of both human-based services and software services in a similar way for socially-enhanced applications, so that processes that include adaptation, task assignment and delegations would be fully automated and complex tasks can be partially executed by both types of resources.

Connected to the human-machine collaborations and mixed systems, the inclusion of social collectives in systems that work with artificial intelligence and cognitive computing is a compelling open area of research.

Finally, we advocate that research from every area in computer science should progress with having the context of privacy in mind, as the way we build applications and systems affect the direction in which our societies will further develop. Technology and society are in an interminable process of mutual effect on their change and transformation, in which process the construction of reality takes place; it is up to us to chose and build the type of reality we want to live in, and what better than when progress is accompanied by transparency, trust and consequently, autonomy.

Bibliography

- [ABC⁺10] Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, Aristides Gionis, and Stefano Leonardi. Power in unity: forming teams in large-scale community systems. In *CIKM*, pages 599–608, 2010.
- [ABC⁺12] Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, Aristides Gionis, and Stefano Leonardi. Online team formation in social networks. In *Proceedings of the 21st international conference on World Wide Web, WWW'12*, pages 839–848, New York, NY, USA, 2012. ACM.
- [ACD⁺07] Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, and Ming Xu. Web services agreement specification (ws-agreement). *Open Grid Forum*, 128, 2007.
- [AIB⁺] Mohammad Allahbakhsh, Aleksandar Ignjatovic, Boualem Benatallah, Seyed-Mehdi-Reza Beheshti, Elisa Bertino, and Norman Foo. Reputation management in crowdsourcing systems. In *CollaborateCom*, pages 664–671. IEEE.
- [AKR⁺13] Elli Androulaki, Ghassan O Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 34–51. Springer, 2013.
- [AKZ13] Aijun An, Mehdi Kargar, and Morteza Zihayat. Finding affordable and collaborative teams from a network of experts. In *Proceedings of the 13th SIAM International Conference on Data Mining, May 2-4, 2013. Austin, Texas, USA.*, pages 587–595, 2013.
- [AMT05] Paolo Avesani, Paolo Massa, and Roberto Tiella. A trust-enhanced recommender system application: Moleskiing. In *Proceedings of the 2005 ACM Symposium on Applied Computing, SAC '05*, pages 1589–1593, New York, NY, USA, 2005. ACM.
- [ARH00] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In *Proceedings of the 33rd Hawaii International Conference on*

System Sciences-Volume 6 - Volume 6, HICSS '00, pages 6007–, Washington, DC, USA, 2000. IEEE Computer Society.

- [BA17] Mohamad Jehad Baeth and Mehmet S. Aktas. A large scale synthetic social provenance database. In *Proceedings of the Ninth International Conference on Advances in Databases, Knowledge, and Data Applications*, DBKDA 2017, pages 16–22, 2017.
- [Bal] Aral Balkan. Ethical design manifesto. *Ind.ie*. Online available: <https://ind.ie/ethical-design/>. Last access: 22.07.2016.
- [BBMK11] Michael S. Bernstein, Joel Brandt, Robert C. Miller, and David R. Karger. Crowds in two seconds: enabling realtime crowd-powered interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 33–42, New York, NY, USA, 2011. ACM.
- [BCBM12] Daniel W. Barowy, Charlie Curtsinger, Emery D. Berger, and Andrew McGregor. Automan: A platform for integrating human-based and digital computation. In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications*, OOPSLA '12, pages 639–654, New York, NY, USA, 2012. ACM.
- [BCKS13] Sonja Buchegger, Jon Crowcroft, Balachander Krishnamurthy, and Thorsten Strufe. Decentralized Systems for Privacy Preservation (Dagstuhl Seminar 13062). *Dagstuhl Reports*, 3(2):22–44, 2013.
- [BKMB12] Michael S. Bernstein, David R. Karger, Robert C. Miller, and Joel Brandt. Analytic methods for optimizing realtime crowdsourcing. *CoRR*, abs/1204.2995, 2012.
- [BN16] Munmun Bhattacharya and Nashreen Nesa. An algorithm for predicting local trust based on trust propagation in online social networks. *International Journal of Computer Applications*, 156(7):8–15, Dec 2016.
- [CAK13] X Cheng, A Azadegan, and Gwendolyn L. Kolfschoten. An evaluation of trust development in group collaborations: A longitudinal case study. volume 46 of *Hawaii International Conference on System Science*, pages 78–85. IEEE computer society press, 2013.
- [CDG⁺12] Simon Caton, Christoph Dukat, Tilo Grenz, Christian Haas, Michaela Pfadenhauer, and Christof Weinhardt. Foundations of trust: Contextualising trust in social clouds. In *2012 Second International Conference on Cloud and Green Computing, CGC 2012, Xiangtan, Hunan, China, November 1-3, 2012*, pages 424–429, 2012.
- [CDS13] Meenal Chhabra, Sanmay Das, and Boleslaw Szymanski. Team formation in social networks. In Erol Gelenbe and Ricardo Lent, editors, *Computer and Information Sciences III*, pages 291–299, London, 2013. Springer London.

- [CFMT09] V. Casola, A. R. Fasolino, N. Mazzocca, and P. Tramontana. An ahp-based framework for quality and security evaluation. In *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 03*, CSE '09, pages 405–411, Washington, DC, USA, 2009. IEEE Computer Society.
- [CGR⁺13] Cristina Cabanillas, José María García, Manuel Resinas, David Ruiz, Jan Mendling, and Antonio Ruiz-Cortés. Priority-based human resource allocation in business processes. In Samik Basu, Cesare Pautasso, Liang Zhang, and Xiang Fu, editors, *Service-Oriented Computing*, pages 374–388, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [CHH16] Michael Colesky, Jaap-henk Hoepman, and Christiaan Hillen. A critical analysis of privacy design strategies. In *IWPE*, San Jose, CA, 2016. IEEE.
- [CNB⁺12] Yuan Cheng, Dang Nguyen, Khalid Bijon, Ram Krishnan, Jaehong Park, and Ravi Sandhu. Towards provenance and risk-awareness in social computing. In *Proceedings of the First International Workshop on Secure and Resilient Architectures and Systems*, SRAS '12, pages 25–30, New York, NY, USA, 2012. ACM.
- [Con13] Noshir Contractor. Some assembly required: leveraging web science to understand and enable team assembly. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 371(1987), 2013.
- [CS16] Wolfie Christl and Sarah Spiekermann. *Networks of Control*. Facultas, 1 edition, 2016.
- [CTD15] Muhammad Z. C. Candra, Hong Linh Truong, and Schahram Dustdar. Analyzing reliability in hybrid compute units. In *IEEE Conference on Collaboration and Internet Computing, CIC 2015, Hangzhou, China, October 27-30, 2015*, pages 150–159, 2015.
- [DB11] Schahram Dustdar and Kamal Bhattacharya. The social compute unit. *IEEE Internet Computing*, 15:64–69, May 2011.
- [DD10] Christoph Dorn and Schahram Dustdar. Composing near-optimal expert teams: a trade-off between skills and connectivity. *On the Move to Meaningful Internet Systems: OTM 2010*, pages 472–489, 2010.
- [DGST11] Schahram Dustdar, Yike Guo, Benjamin Satzger, and Hong Linh Truong. Principles of elastic processes. *IEEE Internet Computing*, 15(5):66–71, 2011.
- [DKR⁺11] Susan B. Davidson, Sanjeev Khanna, Sudeepa Roy, Julia Stoyanovich, Val Tannen, and Yi Chen. On provenance and privacy. In *Proceedings of the*

14th International Conference on Database Theory, ICDT '11, pages 3–10, New York, NY, USA, 2011. ACM.

- [DSSD11] Christoph Dorn, Florian Skopik, Daniel Schall, and Schahram Dustdar. Interaction mining and skill-dependent recommendations for multi-objective team composition. *Data Knowl. Eng.*, 70(10):866–891, October 2011.
- [DT12] Schahram Dustdar and Hong Linh Truong. Virtualizing software and humans for elastic processes in multiple clouds- a service management perspective. *IJNGC*, 3(2), 2012.
- [FC12] Rino Falcone and Cristiano Castelfranchi. *Highlights on Practical Applications of Agents and Multi-Agent Systems: 10th International Conference on Practical Applications of Agents and Multi-Agent Systems*, chapter Trust and Transitivity: How Trust-Transfer Works, pages 179–187. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [FHM14] Simone Fischer-Hübner and Leonardo A. Martucci. *Social Collective Intelligence: Combining the Powers of Humans and Machines to Build a Smarter Society*, chapter Privacy in Social Collective Intelligence Systems, pages 105–124. Springer International Publishing, Cham, 2014.
- [FTDC15] Pablo Fernandez, Hong Linh Truong, Schahram Dustdar, and Antonio Ruiz Cortés. Programming elasticity and commitment in dynamic processes. *IEEE Internet Computing*, 19(2):68–74, 2015.
- [FZDHC11] Maryam Fazel-Zarandi, Hugh J. Devlin, Yun Huang, and Noshir Contractor. Expert recommendation based on social drivers, social network analysis, and semantic data representation. In *Proceedings of the 2Nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '11, pages 41–48, New York, NY, USA, 2011. ACM.
- [Gd05] Matthew E. Gaston and Marie desJardins. Agent-organized networks for dynamic team formation. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '05, pages 230–237, New York, NY, USA, 2005. ACM.
- [GGF14] Yanmin Gong, Yuanxiong Guo, and Yuguang Fang. A privacy-preserving task recommendation framework for mobile crowdsourcing. In *IEEE Global Communications Conference, GLOBECOM 2014, Austin, TX, USA, December 8-12, 2014*, pages 588–593, 2014.
- [GGH14] Daniel Garijo, Yolanda Gil, and Andreas Harth. Challenges for provenance analytics over geospatial data. In *Provenance and Annotation of Data and Processes - 5th International Provenance and Annotation Workshop, IPAW*

2014, Cologne, Germany, June 9-13, 2014. *Revised Selected Papers*, pages 261–263, 2014.

- [Gol05] Jennifer Ann Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, College Park, MD, USA, 2005. AAI3178583.
- [Gol06] Jennifer Golbeck. Computing with trust: Definition, properties, and algorithms. In *Second International Conference on Security and Privacy in Communication Networks and the Workshops, SecureComm 2006, Baltimore, MD, Aug. 28 2006 - September 1, 2006*, pages 1–7, 2006.
- [Gri06] Nathan Griffiths. A fuzzy approach to reasoning with trust, distrust and insufficient trust. In Matthias Klusch, Michael Rovatsos, and Terry R. Payne, editors, *Cooperative Information Agents X*, pages 360–374, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [GRS05] Craig Gentry, Zulfiqar Ramzan, and Stuart G. Stubblebine. Secure distributed human computation. In *Proceedings 6th ACM Conference on Electronic Commerce (EC-2005), Vancouver, BC, Canada, June 5-8, 2005*, pages 155–164, 2005.
- [GW11] Virgil Gligor and Jeannette M. Wing. Towards a theory of trust in networks of humans and computers. In Bruce Christianson, Bruno Crispo, James Malcolm, and Frank Stajano, editors, *Security Protocols XIX*, pages 223–24, isbn=, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [HC08] Henry Hexmoor and Rachil Chandran. Delegations and Trust. *International Journal of Computational Intelligence, Theory and Practice*, 3(2):95–108, 2008.
- [HEV⁺13] Trung Dong Huynh, Mark Ebdon, Matteo Venanzi, Sarvapali D. Ramchurn, Stephen J. Roberts, and Luc Moreau. Interpretation of crowdsourced activities using provenance network analysis. In *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2013, November 7-9, 2013, Palm Springs, CA, USA*, 2013.
- [Hey13] Francis Heylighen. *From Human Computation to the Global Brain: The Self-Organization of Distributed Intelligence*, pages 897–909. Springer New York, New York, NY, 2013.
- [ICK⁺10] Dave Ings, Luc Clément, Dieter König, Vinkesh Mehta, Ralf Mueller, Ravi Rangaswamy, Michael Rowley, and Ivana Trickovic. Ws-bpel extension for people (bpel4people) specification version 1.1. OASIS Committee Specification, August 2010.
- [ICK⁺12] Dave Ings, Luc Clément, Dieter König, Vinkesh Mehta, Ralf Mueller, Ravi Rangaswamy, Michael Rowley, and Ivana Trickovic. Web services

human task (ws-humantask) specification version 1.1. OASIS Committee Specification Draft 12 / Public Review Draft 05., July 2012.

- [KA11] Mehdi Kargar and Aijun An. Discovering top-k teams of experts with/without a leader in social networks. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11*, pages 985–994, New York, NY, USA, 2011. ACM.
- [KAK16] Ibrahim Kamel, Zaher Al Aghbari, and Kareem Kamel. Smartrecruiter: a similarity-based team formation algorithm. *IJB DI*, 3(4):228–238, 2016.
- [Kas08] M. Kasunic. *A Data Specification for Software Project Performance Measures: Results of a Collaboration on Performance Measurement*. Technical report. Carnegie Mellon University, Software Engineering Institute, 2008.
- [KCHO13] Evgeny Kaganer, Erran Carmel, Rudy Hirschheim, and Timothy Olsen. Managing the human cloud. *MIT Sloan Management Review*, 54(2):23–32, 2013.
- [Kea12] Michael Kearns. Experiments in social computation. *Commun. ACM*, 55(10):56–67, October 2012.
- [KG07] Ugur Kuter and Jennifer Golbeck. Sunny: A new algorithm for trust inference in social networks using probabilistic confidence models. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2, AAAI'07*, pages 1377–1382. AAAI Press, 2007.
- [KG08] Sarah N. Keung and Nathan Griffiths. Trust in agent societies. chapter Towards Improved Partner Selection Using Recommendations and Trust, pages 43–64. Springer-Verlag, Berlin, Heidelberg, 2008.
- [KG10] Sarah N. Lim Choi Keung and Nathan Griffiths. *Trust and Reputation*, pages 189–224. Springer London, London, 2010.
- [KHCK13] Malinda Kapuruge, Jun Han, Alan Colman, and Indika Kumara. Road4saas: scalable business service-based saas applications. In *Proceedings of the 25th international conference on Advanced Information Systems Engineering, CAiSE'13*, pages 338–352, Berlin, Heidelberg, 2013. Springer-Verlag.
- [KK08] Aniket Kittur and Robert E. Kraut. Harnessing the wisdom of crowds in wikipedia: Quality through coordination. In *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work, CSCW '08*, pages 37–46, New York, NY, USA, 2008. ACM.
- [KKAH14] Merel Koning, Paulan Korenhof, Gergely Alpár, and Jaap-Henk Hoepman. The abc of abc: an analysis of attribute-based credentials in the light of data protection, privacy and identity, 2014.

- [Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999.
- [Klu06] Lars Kluver. *ICT and Privacy in Europe. Experiences from technology assessment of ICT and Privacy in seven different European countries. Final report October 16, 2006, European Parliamentary Technology Assessment network (EPTA)*. Wien, 2006. Online available: <http://epub.oeaw.ac.at/?arp=0x0013038d> - Last access: 26.4.2016.
- [KPSD11] Roman Khazankin, Harald Psailer, Daniel Schall, and Schahram Dustdar. Qos-based task scheduling in crowdsourcing environments. In *Proceedings of the 9th international conference on Service-Oriented Computing, ICSOC'11*, pages 297–311, Berlin, Heidelberg, 2011. Springer-Verlag.
- [KS11] Nicolas Kaufmann and Thimo Schulze. Worker motivation in crowdsourcing and human computation. In *AAAI workshop on human computation (HCOMP), 8 August, San Francisco, USA*. AAAI Press, 2011.
- [KSGM03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th International Conference on World Wide Web, WWW '03*, pages 640–651, New York, NY, USA, 2003. ACM.
- [KZA08] Robert Kern, Christian Zirpins, and Sudhir Agarwal. Managing quality of human-based eservices. In *ICSOC Workshops*, pages 304–309, 2008.
- [Lan01] Marc Langheinrich. *Privacy by Design — Principles of Privacy-Aware Ubiquitous Systems*, pages 273–291. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.
- [Law11] Edith Law. Defining (Human) Computation. In *Workshop on Crowdsourcing and Human Computation*, 2011.
- [LCCM09] Cam Tu Phan Le, Frédéric Cuppens, Nora Cuppens, and Patrick Maillé. *Collaborative Computing: Networking, Applications and Worksharing: 4th International Conference, CollaborateCom 2008, Orlando, FL, USA, November 13-16, 2008, Revised Selected Papers*, chapter Evaluating the Trustworthiness of Contributors in a Collaborative Environment, pages 451–460. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [LCGM10] Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. Exploring iterative and parallel human computation processes. In *Proceedings of the ACM SIGKDD Workshop on Human Computation, HCOMP '10*, pages 68–76, New York, NY, USA, 2010. ACM.
- [LKD⁺03] Heiko Ludwig, Alexander Keller, Asit Dan, Richard P King, and Richard Franck. Web service level agreement (wsla) language specification. *IBM Corporation*, pages 815–824, 2003.

- [LLT09] Theodoros Lappas, Kun Liu, and Evimaria Terzi. Finding a team of experts in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 467–476, New York, NY, USA, 2009. ACM.
- [LLV07] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In Rada Chirkova, Asuman Dogac, M. Tamer Özsu, and Timos K. Sellis, editors, *ICDE*, pages 106–115. IEEE Computer Society, 2007.
- [LP13] L. J. Larson-Prior. *Parallels in Neural and Human Communication Networks*, pages 39–49. Springer New York, New York, NY, 2013.
- [LSL15] Cheng-Te Li, Man-Kwan Shan, and Shou-De Lin. On team formation with expertise query in collaborative social networks. *Knowl. Inf. Syst.*, 42(2):441–463, 2015.
- [LSM13] Timothy Lebo, Satya Sahoo, and Deborah McGuinness. PROV-O: The PROV ontology. W3C recommendation, W3C, April 2013. <http://www.w3.org/TR/2013/REC-prov-o-20130430/>.
- [LZZ10] Juan Li, Zonghua Zhang, and Weiyi Zhang. Mobitrust: Trust management system in mobile social computing. In *CIT*, pages 954–959. IEEE Computer Society, 2010.
- [Mar94] Stephen Paul Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, 1994.
- [MB09] Maria Moloney and Frank Bannister. A privacy control theory for online environments. In *42st Hawaii International International Conference on Systems Science (HICSS-42 2009), Proceedings (CD-ROM and online), 5-8 January 2009, Waikoloa, Big Island, HI, USA*, pages 1–10, 2009.
- [MB12] Patrick Minder and Abraham Bernstein. Crowdlang: programming human computation systems. Technical report, JAN 2012.
- [MBO12] Nikolaos Mavridis, Thirimachos Bourlai, and Dimitri Ognibene. The human-robot cloud: Situated collective intelligence on demand. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2012 IEEE International Conference on*, page 360–365. IEEE, IEEE, 2012.
- [MCR12] A. Mazza, G. Chicco, and M. Rubino. Multi-objective distribution system optimization assisted by analytic hierarchy process. In *2012 IEEE International Energy Conference and Exhibition (ENERGYCON)*, pages 393–400, Sept 2012.

- [MEC13] Milan Markovic, Peter Edwards, and David Corsar. *Utilising Provenance to Enhance Social Computation*, pages 440–447. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [MEC15] Milan Markovic, Peter Edwards, and David Corsar. *SC-PROV: A Provenance Vocabulary for Social Computation*, pages 285–287. Springer International Publishing, Cham, 2015.
- [MGMD⁺14] Carlos Müller, Antonio M. Gutiérrez, Octavio Martín-Díaz, Manuel Resinas, Pablo Fernández, and Antonio Ruiz-Cortés. *Towards a Formal Specification of SLAs with Compensations*, pages 295–312. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.
- [MH11] W. Michalk and C. Haas. Incentives in service level agreement establishment the case of economic and social aspects. In *2011 First International Workshop on Requirements Engineering for Social Computing*, pages 30–33, Aug 2011.
- [MKA04] Naresh K. Malhotra, Sung S. Kim, and James Agarwal. Internet users’ information privacy concerns (iuipe): The construct, the scale, and a causal model. *Info. Sys. Research*, 15(4):336–355, December 2004.
- [MKC⁺13] Andrew Mao, Ece Kamar, Yiling Chen, Eric Horvitz, Megan E. Schwamb, Chris J. Lintott, and Arfon M. Smith. Volunteering versus work for pay: Incentives and tradeoffs in crowdsourcing. In Björn Hartman and Eric Horvitz, editors, *HCOMP*. AACL, 2013.
- [MKGD] Maria Maleshkova, Srdjan Komazec, Bostjan Grasic, and Ronald Denaux. iservice: Human computation through semantic web services.
- [MLJ⁺10] Luc Moreau, Ding Li, Futrelle Joe, Garijo Verdejo Daniel, Groth Paul, Jewell Mike, Miles Simon, Missier Paolo, Pan Jeff, and Zhao Jun. Open provenance model (opm) owl specification. Technical report, 2010. <http://openprovenance.org/model/opmo>.
- [MM13] Paolo Missier and Luc Moreau. PROV-dm: The PROV data model. W3C recommendation, W3C, April 2013. <http://www.w3.org/TR/2013/REC-prov-dm-20130430/>.
- [MMDC⁺07] Carlos Müller, Octavio Martín-Díaz, Antonio Ruiz Cortés, Manuel Resinas, and Pablo Fernandez. Improving temporal-awareness of ws-agreement. In *ICSOC*, pages 193–206, 2007.
- [MMH02] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation for e-businesses. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS’02)-Volume 7 - Volume 7*, HICSS ’02, pages 188–, Washington, DC, USA, 2002. IEEE Computer Society.

- [MMHJ07] Sara Motahari, Constantine Manikopoulos, Roxanne Hiltz, and Quentin Jones. Seven privacy worries in ubiquitous social computing. In *Proceedings of the 3rd Symposium on Usable Privacy and Security*, SOUPS '07, pages 171–172, New York, NY, USA, 2007. ACM.
- [Mor] Luc Moreau. Provtoolbox. java library to create and convert w3c prov data model representations. <http://lucmoreau.github.io/ProvToolbox/>. Last accessed: 22-4-2017.
- [MRKC09] Aleecia M. McDonald, Robert W. Reeder, Patrick Gage Kelley, and Lorie Faith Cranor. *A Comparative Study of Online Privacy Policies and Formats*, pages 37–55. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [MW09] Winter Mason and Duncan J. Watts. Financial incentives and the "performance of crowds". In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '09, pages 77–85, New York, NY, USA, 2009. ACM.
- [New10] Mark Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.
- [OMM] Maxwell Ogden, Karissa Mckelvey, and Mathias Buus Madsen. Distributed dataset synchronization and versioning. <https://github.com/datprotocol/whitepaper/blob/master/dat-paper.pdf>.
- [Orm13] Levent V. Orman. Bayesian inference in trust networks. *ACM Trans. Management Inf. Syst.*, 4(2):7:1–7:21, 2013.
- [Owm] Paul Owm. Broken promises of privacy: Responding to the surprising failure of anonymization. *UCLA Law Review*, 57(5):1701–1777, August.
- [Par07] Data Protection Working Party. Opinion 4/2007 on the concept of personal data. *Brussels, Belgium: European Commission*, 2007.
- [Par08] Lynne E. Parker. Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, pages 5–14, 2008.
- [PDM14] Heather S. Packer, Laura Drăgan, and Luc Moreau. *An Auditable Reputation Service for Collective Adaptive Systems*, pages 159–184. Springer International Publishing, Cham, 2014.
- [PJN12] Calton Pu, James Joshi, and Surya Nepal, editors. *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2012, Pittsburgh, PA, USA, October 14-17, 2012*. ICST / IEEE, 2012.

- [PJS⁺10] Harald Psailer, Lukasz Juszczak, Florian Skopik, Daniel Schall, and Schahram Dustdar. Runtime behavior monitoring and self-adaptation in service-oriented systems. In *Proceedings of the 2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO '10*, pages 164–173, Washington,DC,USA, 2010. IEEEComputerSociety.
- [PNS11] Jaehong Park, Dang Nguyen, and Ravi S. Sandhu. On data provenance in group-centric secure collaboration. In *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2011, Orlando, FL, USA, 15-18 October, 2011*, pages 221–230, 2011.
- [PP11] Aditya Parameswaran and Neoklis Polyzotis. Answering queries using humans, algorithms and databases. In *Conference on Innovative Data Systems Research (CIDR 2011)*, pages 160–166, January 2011.
- [PSO⁺16] Dragutin Petkovic, Marc Sosnick-Pérez, Kazunori Okada, Rainer Todtenhoefer, Shihong Huang, Nidhi Miglani, and Arthur Vigil. Using the random forest classifier to assess and predict student learning of software engineering teamwork. In *2016 IEEE Frontiers in Education Conference, FIE 2015, Eire, PA, USA, October 12-15, 2016*, pages 1–7, 2016.
- [PSP⁺17] Thomas Pasquier, Jatinder Singh, Julia Powles, David Eyers, Margo Seltzer, and Jean Bacon. Data provenance to audit compliance with privacy policy in the internet of things. *Personal and Ubiquitous Computing*, Aug 2017.
- [PSSD11] H. Psailer, F. Skopik, D. Schall, and S. Dustdar. Resource and agreement management in dynamic crowdcomputing environments. In *Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International*, pages 193–202, Aug 2011.
- [QB11] Alexander J. Quinn and Benjamin B. Bederson. Human computation: A survey and taxonomy of a growing field. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '11*, pages 1403–1412, New York, NY, USA, 2011. ACM.
- [RBAD17] M. Riveni, M. J. Baeth, M. S. Aktas, and S. Dustdar. Provenance in social computing: A case study. In *2017 13th International Conference on Semantics, Knowledge and Grids (SKG)*, pages 77–84, Aug 2017.
- [Reg16] EU Regulation. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). *Official Journal of the EC*, 119, 2016.

- [RHVS13] Sarvapali D. Ramchurn, Trung Dong Huynh, Matteo Venanzi, and Bing Shi. Collabmap: crowdsourcing maps for emergency planning. In *Web Science 2013 (co-located with ECRC), WebSci '13, Paris, France, May 2-4, 2013*, pages 326–335, 2013.
- [RKK⁺11] Jakob Rogstadius, Vassilis Kostakos, Aniket Kittur, Boris Smus, Jim Laredo, and Maja Vukovic. An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. In Lada A. Adamic, Ricardo A. Baeza-Yates, and Scott Counts, editors, *ICWSM*. The AAAI Press, 2011.
- [RMM03] Javier Ignacio Carbó Rubiera, José M. Molina, and Jorge Dávila Muro. Trust management through fuzzy reputation. *Int. J. Cooperative Inf. Syst.*, 12(1):135–155, 2003.
- [RNAD] Mirela Riveni, Tien-Dung Nguyen, Mehmet S. Aktas, and Schahram Dustdar. Application of provenance in social computing: A case study. *Concurrency and Computation: Practice and Experience*, page e4894. e4894 cpe.4894.
- [RND17] Mirela Riveni, Tien-Dung Nguyen, and Schahram Dustdar. Sla-based management of human-based services in business processes for socio-technical systems. In *Business Process Management Workshops - BPM 2017 International Workshops, Barcelona, Spain, September 10-11, 2017, Revised Papers*, pages 361–373, 2017.
- [RTD12] Mirela Riveni, Hong Linh Truong, and Schahram Dustdar. A simulation framework for socially enhanced applications. In Pilar Herrero, Hervé Panetto, Robert Meersman, and Tharam S. Dillon, editors, *OTM Workshops*, volume 7567 of *Lecture Notes in Computer Science*, pages 544–553. Springer, 2012.
- [RTD14] Mirela Riveni, Hong Linh Truong, and Schahram Dustdar. On the elasticity of social compute units. In *Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings*, pages 364–378, 2014.
- [RTD15] Mirela Riveni, Hong Linh Truong, and Schahram Dustdar. Trust-aware elastic social compute units. In *2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, August 20-22, 2015, Volume 1*, pages 135–142, 2015.
- [Sag12] A. B. Sagar. Modeling collaborative task execution in social networks. In Vidyasagar Potdar and Debajyoti Mukhopadhyay, editors, *CUBE*, pages 664–669. ACM, 2012.

- [SB96] Milberg Sandra J. Smith, H. Jeff and Sandra J. Burke. Information privacy: Measuring individuals' concerns about organizational practices. *MIS Q.*, 20(2):167–196, June 1996.
- [Sch13a] Daniel Schall. *Handbook of Human Computation*, chapter Service Oriented Protocols for Human Computation, pages 551–559. Springer New York, New York, NY, 2013.
- [Sch13b] Frank Schulz. Elasticity in service level agreements. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 4092–4097. IEEE, 2013.
- [SD10] Daniel Schall and Schahram Dustdar. *Social Informatics: Second International Conference, SocInfo 2010, Laxenburg, Austria, October 27-29, 2010. Proceedings*, chapter Dynamic Context-Sensitive PageRank for Expertise Mining, pages 160–175. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [SDB10] Daniel Schall, Schahram Dustdar, and M. Brian Blake. Programming human and software-based web services. *IEEE Computer*, 43(7):82–85, 2010.
- [SDC15] Rick Salay, Fabiano Dalpiaz, and Marsha Chechik. Integrating crowd intelligence into software. In *Proceedings of the Second International Workshop on CrowdSourcing in Software Engineering, CSI-SE '15*, pages 1–7, Piscataway, NJ, USA, 2015. IEEE Press.
- [SFC⁺17] Saiganesh Swaminathan, Raymond Fok, Fanglin Chen, Ting-Hao (Kenneth) Huang, Irene Lin, Rohan Jadvani, Walter S. Lasecki, and Jeffrey P. Bigham. Wearmail: On-the-go access to information in your email with a privacy-preserving human computation workflow. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, UIST '17*, pages 807–815, New York, NY, USA, 2017. ACM.
- [SH10] Dafna Shahaf and Eric Horvitz. Generalized task markets for human and machine computation. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI'10*, pages 986–993. AAAI Press, 2010.
- [SHS09] Osamuyimen Stewart, Juan M. Huerta, and Melissa Sader. Designing crowdsourcing community for the enterprise. In *Proceedings of the ACM SIGKDD Workshop on Human Computation, HCOMP '09*, pages 50–53, New York, NY, USA, 2009. ACM.
- [SJB⁺12] Bikram Sengupta, Anshu Jain, Kamal Bhattacharya, Hong Linh Truong, and Schahram Dustdar. Who do you call? problem resolution through social compute units. In Chengfei Liu, Heiko Ludwig, Farouk Toumani,

and Qi Yu, editors, *ICSOC*, volume 7636 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 2012.

- [SKS12] Thimo Schulze, Simone Krug, and Martin Schader. Workers’ task choice in crowdsourcing and human computation markets. In *Proceedings of the International Conference on Information Systems, ICIS 2012, Orlando, Florida, USA, December 16-19, 2012*, 2012.
- [Sog07] Chris Soghoian. Aol, netflix and the end of open access to research data, cnet news. http://news.cnet.com/8301-13739_3-9826608-46.html, Nov., 30, 2007.
- [SRTD14] Ognjen Scekic, Mirela Riveni, Hong-Linh Truong, and Schahram Dustdar. *Social Interaction Analysis for Team Collaboration*, pages 1807–1819. Springer New York, New York, NY, 2014.
- [SRTD17] Ognjen Scekic, Mirela Riveni, Hong-Linh Truong, and Schahram Dustdar. *Social Interaction Analysis for Team Collaboration*, pages 1–16. Springer New York, New York, NY, 2017.
- [SSD10a] Florian Skopik, Daniel Schall, and Schahram Dustdar. Modeling and mining of dynamic trust in complex service-oriented systems. *Inf. Syst.*, 35(7):735–757, 2010.
- [SSD10b] Florian Skopik, Daniel Schall, and Schahram Dustdar. Trustworthy interaction balancing in mixed service-oriented systems. In Sung Y. Shin, Sascha Ossowski, Michael Schumacher, Mathew J. Palakal, and Chih-Cheng Hung, editors, *SAC*, pages 799–806. ACM, 2010.
- [SSPD11] Florian Skopik, Daniel Schall, Harald Psailer, and Schahram Dustdar. Adaptive provisioning of human expertise in service-oriented systems. In *Proceedings of the 2011 ACM Symposium on Applied Computing, SAC ’11*, pages 1568–1575, New York, NY, USA, 2011. ACM.
- [STD08] Daniel Schall, Hong Linh Truong, and Schahram Dustdar. Unifying human and software services in web-scale collaborations. *IEEE Internet Computing*, 12(3):62–68, 2008.
- [STD13] Ognjen Scekic, Hong-Linh Truong, and Schahram Dustdar. Incentives and rewarding in social computing. *Commun. ACM*, 56(6):72–82, June 2013.
- [Swe02] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.
- [SZP15] Isuru Suriarachchi, Quan Zhou, and Beth Plale. Komadu: A capture and visualization system for scientific data provenance. *Journal of Open Research Software*, page p.e4., 3(1) 2015.

- [TBA16] Yucel Tas, Mohamed Jehad Baeth, and Mehmet S. Aktas. An approach to standalone provenance systems for big social provenance data. In *12th International Conference on Semantics, Knowledge and Grids, SKG 2016, Beijing, China, August 15-17, 2016*, pages 9–16, 2016.
- [TD09] Hong-Linh Truong and Schahram Dustdar. *Transactions on Petri Nets and Other Models of Concurrency II: Special Issue on Concurrency in Process-Aware Information Systems*, chapter Online Interaction Analysis Framework for Ad-Hoc Collaborative Processes in SOA-Based Environments, pages 260–277. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [TDB12] Hong Linh Truong, Schahram Dustdar, and Kamal Bhattacharya. Programming hybrid services in the cloud. In Chengfei Liu, Heiko Ludwig, Farouk Toumani, and Qi Yu, editors, *ICSOC*, volume 7636 of *Lecture Notes in Computer Science*, pages 96–110. Springer, 2012.
- [TGS14] Hien To, Gabriel Ghinita, and Cyrus Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *Proc. VLDB Endow.*, 7(10):919–930, June 2014.
- [Toc14] Eran Toch. Crowdsourcing privacy preferences in context-aware applications. *Personal and Ubiquitous Computing*, 18(1):129–141, 2014.
- [TWH14] Wei-Tek Tsai, Wenjun Wu, and Michael N. Huhns. Cloud-based software crowdsourcing. *IEEE Internet Computing*, 18(3):78–83, 2014.
- [Urb13] J Urbano. *A Situation-aware and Social Computational Trust Model*. PhD thesis, 2013.
- [VA05] Luis Von Ahn. *Human Computation*. PhD thesis, Pittsburgh, PA, USA, 2005. AAI3205378.
- [VLL10] Maja Vukovic, Mariana Lopez, and Jim Laredo. Peoplecloud for the globally integrated enterprise. In *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, pages 109–114. Springer Berlin Heidelberg, 2010.
- [WAW15] Therese L Williams, Nitin Agarwal, and Rolf T Wigand. Protecting private information: Current attitudes concerning privacy policies. 2015.
- [WF05] Molly McLure Wasko and Samer Faraj. Why should i share? examining social capital and knowledge contribution in electronic networks of practice. *MIS Q.*, 29(1):35–57, March 2005.
- [WGS⁺13] Wesley Willett, Shiry Ginosar, Avital Steinitz, Björn Hartmann, and Maneesh Agrawala. Identifying redundancy and exposing provenance in crowdsourced data analysis. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2198–2206, December 2013.

- [YKL11] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. A survey of crowdsourcing systems. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 766–773. IEEE, 2011.
- [YRM08] H. Q. Yu and S. Reiff-Marganiec. A method for automated web service selection. In *2008 IEEE Congress on Services - Part I*, pages 513–520, July 2008.
- [ZF11] Justin Zhan and Xing Fang. Trust maximization in social networks. In John Salerno, ShanchiehJay Yang, Dana Nau, and Sun-Ki Chai, editors, *Social Computing, Behavioral-Cultural Modeling and Prediction*, volume 6589 of *Lecture Notes in Computer Science*, pages 205–211. Springer Berlin Heidelberg, 2011.
- [ZKJG11] Xinwen Zhang, Anugeetha Kunjithapatham, Sangoh Jeong, and Simon Gibbs. Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. *Mob. Netw. Appl.*, 16(3):270–284, June 2011.
- [ZMM00] Giorgos Zacharia, Alexandros Moukas, and Pattie Maes. Collaborative reputation mechanisms for electronic marketplaces. pages 371–388, 2000.
- [ZSTD15] Philipp Zeppezauer, Ognjen Scekic, Hong-Linh Truong, and Schahram Dustdar. *Service-Oriented Computing - ICSOC 2014 Workshops: WESOA; SeMaPS, RMSOC, KASA, ISC, FOR-MOVES, CCSA and Satellite Events, Paris, France, November 3-6, 2014, Revised Selected Papers*, chapter Virtualizing Communication for Hybrid and Diversity-Aware Collective Adaptive Systems, pages 56–67. Springer International Publishing, Cham, 2015.

Appendix A

```
public static double overallSatisfactionScore(double metric1, double metric2, double metric3){
    double s;
    offeredValues=new double[]{metric1, metric2,metric3};
    getWeightsVector();
    overallSatisfaction=0;
    double finalSatisfaction=0;
    for (int l=0; l<3; l++){
        QoSr = QoSweight.get(l);
        QoSr = Math.round(QoSr * 100);
        QoSr = QoSr/100;
        offeredValue_l=offeredValues[l];
        requestedValue_l=requestedValues[l];
        s=satisfactionFunction(offeredValue_l,requestedValue_l,1);
        overallSatisfaction = overallSatisfaction+(QoSr*s);
        overallSofR.add(overallSatisfaction);
    }
    return overallSatisfaction;
}

public ICUList getRankedICUs() throws FileNotFoundException, JAXBException, IOException, URISyntaxException {
    //Rank appropriate ICU/ICUs for the task
    double a,b,c;
    currentICUsfromXML();
    icuList=currentICUList.getICUPool();
    for (int h=0; h<icuList.getICUPool().size(); h++){
        a=icuList.get(h).getICUTrust();
        b=icuList.get(h).getICUReputation();
        c=icuList.get(h).getICUReliability();
        overallSatisfactionScore(a,b,c);
        icuList.get(h).setICUsatisfactionScore(overallSatisfaction);
        overallSperR.add(overallSatisfaction);
    }
    double max = Collections.max(overallSperR);
    for (int k=0; k<icuList.size(); k++){
        if(icuList.get(k).getICUSatisfactionScore()==max){
            selectedICUs.add(icuList.get(k));
        }
    }
    Collections.sort(icuList, new ICUComp());
    ICUInXML.storeICUsInXML(icuList, file);
    System.out.print("\n");
    return icuList;
}
}
}
}

```

```

for(int j=0; j<taskList.size(); j++){
    tempTask=taskList.get(j);
    //tempTask.taskDelegated=false;
    tempTime=tempTask.getTaskInQueue();
    if(tempTime==min && (tempTime == tempTask.getTaskDeadline()/2)){
        currentICU=tempTask.getTaskOwner();
        System.out.println();
        System.out.println("Task waiting-time for task with id "+tempTask.getTaskId()+
            " reached the pre-set threshold, delegating task...");
        for (a=0; a<icus.size(); a++){
            icutemp=icus.get(a);
            if(tempTask.getTaskOwner().getICUID()!=icutemp.getICUID()){
                if(tempTask.getSkillReq().equals(icutemp.getICUSkill())){

                    icutemp.ICUMetrics_.updateICUTaskQueue(tempTask);
                    currentICU.ICUMetrics_.removeTask(tempTask);
                    currentICU.failedTasks++;
                    icutemp.ICUMetrics_.updateDelegatedTasks(tempTask);
                    if(!currentSCU.contains(icutemp)){
                        currentSCU.addICU(icutemp, currentSCU); //add(icu)

                    System.out.println("Task "+tempTask.getTaskId()+" delegated from ICU with id "+
                        tempTask.getTaskOwner().getICUID() + " to ICU with id "+  icutemp.getICUID());
                    tempTask.setTaskState(9);

                    tempTask.setTaskOwner(icutemp);
                    icutemp.ICUMetrics_.getAllICUTasks().getTaskwithId(tempTask.getTaskId()).taskDelegated=true;
                    break;
                }
            }
        }
    }
}

```


Appendix B

task waiting-time for task with id 32 reached the pre-set threshold, delegating task...
Task 32 delegated from ICU with id 8 to ICU with id 13

Task waiting-time for task with id 43 reached the pre-set threshold, delegating task...
Task 43 delegated from ICU with id 1 to ICU with id 3

Task waiting-time for task with id 48 reached the pre-set threshold, delegating task...
Task 48 delegated from ICU with id 7 to ICU with id 10

Task waiting-time for task with id 49 reached the pre-set threshold, delegating task...
Task 49 delegated from ICU with id 7 to ICU with id 10

Task waiting-time for task with id 18 reached the pre-set threshold, delegating task...
Task 18 delegated from ICU with id 8 to ICU with id 13

===== OUTPUT: CURRENT SCU Members =====

ICU ID	alltasks	mcts	ptrust	stt	succ tasks	success rate
1	85	0.9317	0.7624	0.8301	72	0.8471
6	88	0.6199	0.7568	0.7021	74	0.8409
8	75	0.8004	0.7680	0.7810	64	0.8533
0	109	0.3741	0.7761	0.6153	94	0.8624
7	93	0.7977	0.7355	0.7604	76	0.8172
13	11	0.8855	0.3667	0.5742	11	1.0000
3	13	0.5555	0.4333	0.4822	13	1.0000
10	17	0.3561	0.5667	0.4825	17	1.0000

===== SCU TRUST-BASED METRICS RESULTS =====

SCU Id: 0 Productivity: 0.7333 Performance trust: 0.6457 Social trust: 0.6651 Socio-technical trust: 0.6554
SCU total tasks: 491 SCU invocation delegated tasks: 9 total delegated tasks 70
Total tasks time 661

Task waiting-time for task with id 23 reached the pre-set threshold, delegating task...
Task 23 delegated from ICU with id 6 to ICU with id 11

Task waiting-time for task with id 31 reached the pre-set threshold, delegating task...
Task 31 delegated from ICU with id 8 to ICU with id 13

Task waiting-time for task with id 34 reached the pre-set threshold, delegating task...
Task 34 delegated from ICU with id 8 to ICU with id 13

Task waiting-time for task with id 37 reached the pre-set threshold, delegating task...
Task 37 delegated from ICU with id 6 to ICU with id 11

Task waiting-time for task with id 27 reached the pre-set threshold, delegating task...
Task 27 delegated from ICU with id 6 to ICU with id 11

===== OUTPUT: CURRENT SCU Members =====

ICU ID	alltasks	mcts	ptrust	stt	succ tasks	success rate
6	103	0.3465	0.7515	0.5895	86	0.8350
8	87	0.6098	0.7552	0.6970	73	0.8391
7	100	0.6237	0.7290	0.6869	81	0.8100
1	95	0.5647	0.7579	0.6806	80	0.8421
0	115	0.7614	0.7826	0.7741	100	0.8696
13	14	0.6112	0.4667	0.5245	14	1.0000
3	15	0.4790	0.5000	0.4916	15	1.0000
10	19	0.7430	0.6333	0.6772	19	1.0000
11	17	0.9214	0.5667	0.7086	17	1.0000

===== SCU TRUST-BASED METRICS RESULTS =====

SCU Id: 0 Productivity: 0.7407 Performance trust: 0.6603 Social trust: 0.6290 Socio-technical trust: 0.64
SCU total tasks: 565 SCU invocation delegated tasks: 10 total delegated tasks 80
Total tasks time 732

Appendix C

Welcome to our survey! We appreciate the time you take to fill it in!

Before you start, it is important that you read your rights as a participant and understand our research. Thus, we ask you to read your rights in the document at the following link: [click here](#).

Please enter "/", or "no other" in the second and third lines of the first two questions if you work only on one platform.

Section A: Survey Questions

- A1. On which Crowdsourcing, or Online Labor Market Platform/s do you work or request work? (If you work or request work on multiple platforms please write them all, each on a new line.)

Platform name

A second platform name

A third platform name



A2. Please enter your username on the platform you work/request work on. If you are registered on multiple platforms you can enter your username and the name of the platform (separated by a comma) on each separate line.

Note: if you use a platform with a *full name policy* please write your Name and Surname, if you have a username only, please enter only your username (not your Name and Surname)..

We require this information for the purpose of avoiding survey misuses, we are just going to check your answer against the platform and not use the data you provide for anything else.

Username (or Name and Surname) and the corresponding platform name on which you work/request work

Another username and platform name

A third username and platform name

A3. How much are you concerned with the personal information that you are obliged to share so as to *register* on these platforms?

Not concerned

Somewhat concerned

Very concerned

A4. How much are you concerned with the personal information that you need to share so as to *create/build your profile* and *verify your identity* on these platforms?

Not concerned

Somewhat concerned

Very concerned

A5. Which of the following personal information would you prefer to seclude/hide/not share with the platform that you use (please specify if there is some other information that you are asked to provide and would not want to share) .

Name and Surname

E-mail address

Phone number

Birthdate

Your photograph

Location information/(Home)Mailing address

Utility bills (sometimes used to verify address)



A government issued ID (Passport, Personal ID, Drivers license)

Bank account information

None of the above

Other

Other

A6. Are you concerned that your information will be misused (by the /platform that you are registered with)?

I trust the platform

Somewhat concerned

Very concerned

A7. Have you ever been concerned about what storage the platforms that you work on utilize for storing your personal data? For example, have you ever thought about the following questions: do platforms store data on their own servers; do they use Cloud providers to store your data, in which case your data may be shared with a third-party; are they storing data offshore, in which case different privacy protecting laws exist etc.

I admit I have never thought about these things and frankly I am not concerned.

I admit I have never thought about these things but I became concerned now.

I have thought about these things and am concerned.

A8. Have you ever provided false information to one of these services/platforms, when asked for personal information?

Yes

No

A9. If you have ever provided false information, why did you do so?

They required information I could not provide (*marked questions are sometimes a bad idea...)

I did not feel comfortable providing this information (Why do they always need a phone number...)

I did so to create an additional account

My answer was "No" on the previous question.

Other

Other



A10. If you have provided false information on these platforms, please mention one information type.

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

A11. If you would want to work anonymously online, what would be the reasons for it? Please give a short answer. Leave it blank if not working anonymously doesn't bother you.

--

A12. Please state your agreement with the following statements (by also selecting your level of familiarity with the topics):

	I strongly agree	I agree	I somewhat agree	I disagree	I do not want to answer
Companies/Platforms than enable and provide human computation should be more transparent about how they use my personal information.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Human computation Companies/Platforms should clearly state under which country/state law they operate.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I am concerned about the privacy regulations/laws of the country in which i reside and work.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Research and industry should increase their efforts in enabling users to have more control over how their data is used.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Research and industry should increase their efforts in enabling tools and mechanisms that will enable users to own their own data, in contrast to current standards where companies own users' data.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
				I answered but I need more information on the topic.	I answered being knowledgeable enough on the topic.
Companies/Platforms than enable and provide human computation should be more transparent about how they use my personal information.					<input type="checkbox"/>
Human computation Companies/Platforms should clearly state under which country/state law they operate.					<input type="checkbox"/>
I am concerned about the privacy regulations/laws of the country in which i reside and work.					<input type="checkbox"/>
Research and industry should increase their efforts in enabling users to have more control over how their data is used.					<input type="checkbox"/>
Research and industry should increase their efforts in enabling tools and mechanisms that will enable users to own their own data, in contrast to current standards where companies own users' data.					<input type="checkbox"/>

