Ontology-based Cloud Platform for Human-Driven Applications

Alexander Smirnov, Andrew Ponomarev, Tatiana Levashova, Nikolay Shilov
SPIIRAS
St.Petersburg, Russia
{smir, ponomarev, tatiana.levashova, nick}@iias.spb.su

Abstract—**The number of systems and applications where large groups of people are included into the information processing "loop" is growing. Common problem with this kind of systems is that each of them requires large number of contributors and collecting this number may take significant time and effort. This paper aims at development of an ontology-driven cloud platform that would support deployment of various human-based applications and therefore reuse the existing crowd. Three features that distinguish the proposed platform from similar developments are ontologies, digital contracts and resource monitoring facilities. Ontological mechanisms (ability to precisely define semantics and use inference to find related terms) are used to find and allocate human resources required by software services. Digital contracts are used to achieve predictability required by cloud users (application developers). Finally, explicit mechanisms for resource monitoring are essential, as human resources are always limited and the developers of applications deployed in the platform should be aware of particular limitations.**

I. INTRODUCTION

The spread of the Internet and continuing growth of the number of connected devices paved the way for a new kind of hybrid human-machine systems, where a distributed group of people (usually called crowd) communicating via the Internet is included into the information processing together with software components. It is especially important for problems for which there is lack of fast and reliable algorithms (e.g., for problems with incomplete definitions, involving intensive usage of common sense knowledge or dealing with "computationally hard" information representations, like audio and visual). Examples of this new kind of systems are microtask markets (with the most prominent Amazon Mechanical Turk), various citizen science projects ([1], [2]), community sense and response systems (e.g., [3]), general collaborative mapping (e.g., OpenStreetMap, Google Map Maker, WikiMapia), crisis mapping (e.g., [4], [5]) and many others.

There are several inherent problems with human-machine systems limiting their wide adoption. Most important problems are the problem of quality of the results, huge difference in productivity for the same task between different individuals, and very limited nature of human resources. This paper mostly addresses the last of these problems. Each human-machine application needs a large number of human contributors (who will accomplish tasks not tractable by software components). The number of available contributors influences both task processing latency and expected quality of the results (e.g., large number of contributors allows to implement various quality assurance mechanism where results provided by one contributor are verified by others). On the other hand, collecting large number of contributors requires significant effort and time. Therefore, the motivation of this research is to develop a unified resource management environment, that could serve as a basis on which any human-based application could be deployed much like the way cloud computing is used nowadays to decouple computing resources management from application software. That would significantly streamline the development and deployment of human-based applications and services that are important in some application areas. In terms of NIST recommendation document [6] this functionality is classified as Platform-as-a-Service (PaaS).

Other problems listed in the previous paragraph dictate additional requirements to the proposed platform. Say, significant difference in productivity and huge number of possible skills and knowledge a human may possibly possess (unlike hardware or software resources typical for current cloud environments) requires extensive (and dynamic) description of each human resource and intelligent discovery and allocation mechanisms. Limited nature of human resources requires that the amount of resources available for the particular human-based application deployed in the cloud has to be estimated and made available for the application developer. It is essential, because each application, in its turn, provides some services and their capacity is effectively limited by the amount of human resources; particular value of this limitation may influence QoS policies of the application, its pricing and so on.

Distinguishing features of the proposed human-computer cloud platform are ontologies and digital contracts. Ontological mechanisms (ability to precisely define semantics and use inference to find related terms) are used to find and allocate human resources required by software services. While digital contracts are used to achieve predictability required by cloud users (application developers). These digital contracts specify terms on which a contributor agrees to provide his/her competencies to the cloud application developer, rewarding and possible penalties. Cloud environment uses these contracts both to allocate service's task and to inform users about possible capacity.

The review of existing developments has shown that there are no current crowd computing platforms with this combination of features and capabilities.

One of the potential application areas is e-Tourism (and decision support in e-Tourism), where human input and human

involvement is very important due to subjective nature the domain. Therefore, to motivate the development of the human-computer cloud, the proposed cloud environment mechanisms are described with examples of tourist services and applications that may leverage these mechanisms.

II. Related work

The design of human-computer cloud touches several areas. First, existing attempts to extend cloud computing paradigm by non-traditional types of resources. Second, as we see ontologies as a way to solve some of the resource management problems, our research touches current endeavors in ontological modeling of cloud environments. Third, we need to represent human competencies, and it gives another area of work. Current developments in each of these areas are briefly discussed in separate subsections.

A. Cloud extensions

Despite vast majority of modern cloud computing systems manage only traditional computational resources, there are also several research initiatives that try to extend principles of elastic resource management to other types of resources. Relevant developments in applying cloud principles of elastic resource provisioning to a wider spectrum of resources can be classified into two groups: 1) cloud sensing and actuation environments and 2) cloud-managed human resource environments.

One of the examples of cloud sensing and actuation environment is [7], where sensing resource is regarded as a service. Later, based on this work [8] proposes a cloud architecture for mobile crowdsensing MCSaaS (Mobile CrowdSensing as a Service), which defines a unified interface allowing any smartphone user to become a part of a cloud and allow to use his/her smartphone sensors in some way that he/she finds acceptable in exchange for some monetary reward or even voluntary.

ClouT (Cloud+IoT) project [9] falls in the same category and is aimed on providing enhanced solutions for smart cities by using cloud computing in the IoT domain. It proposes multi-layer cloud architecture where lower (infrastructure) layer manages both sensing and computing resources. Both ClouT and MCSaaS approaches are highly relevant to the platform being designed, however, they are focused mostly on sensing resources and consider human resources only due to the fact, that human provides the access to his/her smartphone and can control it to make some operations (i.e. playing a role of virtual sensor). A human, however, may be not only a supplier of information (like sensor), but a processor of it.

The second group, namely cloud-managed human resource environments, has another perspective aiming on managing member's skills and competencies in a standardized flexible way (e.g. [10], [11]), regarding human as a specific resource that can be allocated from a pool for performing some tasks. For example, in [10] the cloud consisting of human-based services and software-based services is considered. Like hardware infrastructure is described in terms of some characteristics (CPU, memory, network bandwidth), human-computing unit in this model is described by the set of skills.

However, no existing attempts to build human-computer cloud leverage the expressive power of ontologies. Besides, ideas discussed in these references are mostly focused on human-computer IaaS, whereas this paper discusses principles that lie behind human-computer PaaS.

There are also numerous relevant research papers on crowdsourcing/crowd computing (including crowdsensing). While using some of the ideas of the existing crowd computing platforms (e.g., typical workflows ensuring quality control, elements of skill management), the proposed cloud platform is aimed on a standardization of human effort and seamless integration of it into a cloud stack.

B. Cloud ontologies

The authors of [12] define several directions in applying ontologies to the cloud computing field: a) cloud resources and servce description; b) cloud security; c) cloud interoperability; d) cloud services discovery and selection.

Some of the concrete ontologies that are successfully applied to model various aspects of a cloud system are [13]: SKOS, mOSAIC, Linked USDL, CoCoOn, UFO-S. UFO-S [14], for example, being a core reference ontology for services, is able to explain a number of perspectives on services, including those that emphasize services as value co-creation, as capabilities and as application of competences. As a core ontology, UFO-S refines concepts of a foundational ontology (the Unified Foundational Ontology (UFO) [15]) by providing a conceptualization for services that is independent of a particular application domain.

The modeling of SLA in cloud is another problem that attracts much attention [13], there are approaches WSLA for the definition of SLAs of WebServices [16], WS-Agreement was developed from the Grid Resource Allocation Agreement Protocol Working Group (GRAAP-WG), SLAng [17], LinkedUSDL-SLA (in the context of FI-WARE project) etc.

Although, these developments are important in the domain of cloud computing, they have to be adapted to the human-computer cloud, as they do not consider human part.

C. Human competencies

The analysis of the research dealing with competence modelling shows that competency is derived from three factors: *knowledge*, *skills* (abilities), and *attitudes* (not necessarily connected to specific knowledge or skills) [18]. These features a human resource may have or need to do a task in a specific context. Most approaches share this idea (e.g., [19], [20], [21]). In particular cases some additional factors are taken into account.

Inside the IEEE RCD standard [22], the competency term points out the aspects of the competence, skill, attitude, ability and includes the *learning* goals. The relation to learning is widely exploited in the competence ontologies intended to various learning organizations (e.g., [23], [24]).

An ontology [25] for competency modeling combines the concepts of knowledge, skill, attitudes, and *performance*. The author [25] defines competencies as statements that someone

can demonstrate; the application of a generic skill to some knowledge, with a certain degree of performance. Performance is defined as quality of produced outcomes related to the required skills and competencies. The generic skills are processes acting on knowledge in an application domain, for instance, to perceive, memorize, assimilate, analyze, synthesize, or evaluate knowledge items (e.g., "establish a diagnosis"). In that ontology, competencies are statements that link together skills and attitudes to knowledge required from a group of persons and, more generally, from resources. Each competency is composed of a single competency statement, exactly one generic skill that may require precision using performance indicators, and at least one knowledge entity.

The ontology for skill and competence management [26] introduces the notion of *proficiency*. The ontology specifies skills at particular levels of proficiency as what enable the performance of activities, and skill statements as properties that have degrees of belief associated with them and that can change over time. The ontology is intended to organizations could evaluate whether an individual actually possesses a skill at a level of proficiency.

III. Platform concepts

This section introduces main design rationales and concepts of the proposed platform. It identifies main actors, interacting with the platform, major requirements that drive the design, information associated with applications and contributors, allowing to fulfill the requirements. The section finishes with main use cases presenting a general view of the platform.

There are three main categories of actors involved in the proposed cloud platform:

End users (*application/service developers*), who use the platform to create and run applications and/or services that require human effort. Of course, these applications can also use other services and hardware, like in any other PaaS.

Contributors, who are available to serve as human resources in a human-computer cloud environment.

System Administrators and Infrastructure Providers, who own and maintain the required infrastructure.

Primary requirements from the side of *End users* considered in the platform design are following:

- The platform must provide tools to deploy, run, and monitor applications that require human information processing.
- The platform must allow to specify what kind of human information processing is required for an application (as some human-based services, like, e.g., image tagging, require very common skills, while others, like tourism decision support, require at least local expertise in certain location).
- The platform must allow to estimate and monitor human resources available for the application. This requirement contrasts to conventional cloud applications, where overall amount of resources possessed by cloud provider is considered to be inexhaustible, and the capacity consumed by an application is in theory limited only by the available budget. However, human resources are always limited, especially when it comes to people with some specialized competencies and knowledge. Besides, the particular rewarding scheme designed by the application developer may be not appealing and not able to collect the required number of contributors. Therefore, application developer should be able to have information to know what capacity is available to the application. Based on this information he/she may change the rewarding scheme, set up his/her own SLA (for his/her consumers) etc.
- The platform must account for temporal dimension of resource availability. Like in the previous requirement, this is specific mostly for human resources. For example, some contributors are ready to participate in information processing activities controlled by the cloud platform in their spare time (non-working hours) only. It means that resource capacity during non-working hours will be larger than during working hours. However, for some applications (e.g., online tourist support) reaction time is important. Therefore, tourist decision support service developers should have temporal perspective of the resource availability.

A. Application description

The aim of any cloud environment providing the PaaS service model is to streamline the development and deployment of the applications by providing specialized software libraries and tools that help developers to write code abstracting from many details of resource management. Instead, those resource management operations are performed automatically by PaaS environment usually according to some description (declarative configuration) provided by the developer of the service being executed. The human-computer cloud environment being developed supports similar approach, however, with inevitable extensions caused by the necessity of working with human resources. To streamline the development of applications that require human actions, the platform allows both a developer to describe what kind of human resources are required for this particular application, and a contributor to describe what kind of activities he/she can be involved in and what competencies he/she possesses. Declarative specification of service requirements is quite typical for cloud environments. They are used, for example, in cloud orchestration definition language TOSCA, that allows, for example, to specify how many virtual machines and with what services should be created for a particular application running in cloud. However, these definitions turn out to be insufficient for the purpose of human-computer cloud. One of the reasons is multifarious nature of possible human skills and competencies. While virtual machine can be described with a very limited number of features (e.g. cpu, ram, i/o capacity), human contributor's skills and abilities are highly multidimensional, they can be described in different levels of detail and be connected to wide range of particular application areas. Besides, the same skills can be described in different ways, and, finally, most of the skill descriptions in real world are incomplete (however, there

might be a possibility to infer some skills that a human might possess from those that he/she explicitly declared).

Application that is to be deployed in the proposed human-computer cloud beside the source code must contain a descriptor that includes following components (here we list all the components, but focus on those, relevant to human part):

- configuration parameters (e.g. environment variables controlling the behavior of the compiled code);
- software dependencies of the application (what platform services and/or other applications it relies on, e.g., database service, messaging service, etc.);
- human resource requirements, specifying what human skills and competencies the application needs to function. These requirements are also (as software requirements) are resolved during the service deployment, but as (1) resolving these requirements employs ontology matching which may result in some tradeoffs, (2) human resources are much more limited than hardware/software, the status and details of the requirements resolution are available to the developer and can be browsed via the management console;
- digital contract template for each type of human resources. By the type of human resource we mean each specific profile of requirements. For example, an itinerary planning application may require people with significant local expertise as well as people with shallow local expertise but good language skills. The application defines these two requirements profiles and may associate different digital contracts for them (in terms of reaction time and/or payment).

More formally, the general structure of an application descriptor can be drawn as a UML diagram (Fig. 1).

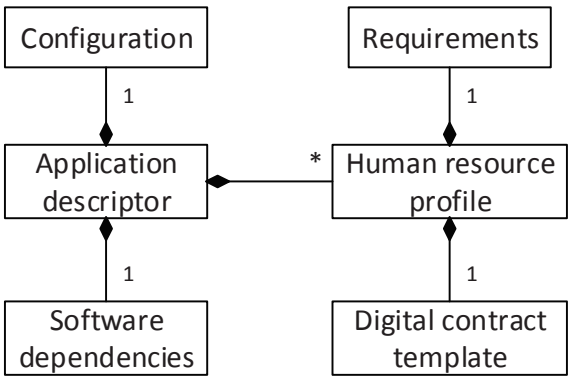

Fig. 1. Application descriptor's general structure

One of the distinguishing features of the proposed platform is the way to formally describe requirements addressing the different ways of describing the same human capabilities. First of all, we adopt the three-way understanding (knowledge, skill, and attitude) of human competencies, as it is common in current literature. Then (and the most important) we allow to use arbitrary ontology concepts as specific skills or knowledge areas.

Therefore, the application requirements definition is equivalent to *SELECT* pattern of a SPARQL [27] query. E.g.:

```
?contributor hcc:knows gno:660129.
```

Here `hcc:knows` is a property defined in the platform ontology, `gno` is a namespace for geographical objects described with Geonames ontology [28], and 660129 is the identifier of an object in Geonames ontology (Espoo). In other words, this pattern is to match all contributors who specified that they have knowledge of Espoo. The major benefit of using ontologies is that it is possible to discover resources defined with related, similar but not exact terms. This is done either by using existing public mappings between ontologies (stating equivalence between concepts of different ontologies), or by ontology inference (e.g., local knowledge of Espoo concept implies to some degree local knowledge of Helsinki concept and Uusimaa concept), or potentially even by ontology matching [29]. So, the pattern above will also match those contributors who described their knowledge not using Geonames ontology but other geographical ontologies.

However, human resource requirement of the form above is quite rare (it only usable as a requirement for local applications for Espoo). Application that need local knowledge in any region are more widespread. To account for this, more general, case requirement definition may contain placeholders, e.g.:

```
?contributor hcc:knows
        [?p a gn:A;
         ?p gn:countryCode "FI"].
```

Here `?p` is a placeholder, and its value has to be an administrative boundary (due to Geonames `A` concept) and be located in Finland (due to Geonames `countryCode` property). Therefore, this requirement will be fulfilled for any contributor that has local experience in some region in Finland (and the respective contributor will be attributed the value of placeholder).

It is important that we do not fix any particular set of ontologies, but support ontology-based resource discovery. It allows tourist applications deployed on the platform to use public cultural, historical, and geographical ontologies, whereas, e.g., applications, that employ human-based information processing in the area of medicine or biology use the ontologies of the respective domain. The only restriction is that these ontologies have to be expressed in OWL 2 [30]. Moreover, to guarantee computational efficiency, they have to conform to OWL 2 EL profile.

Another distinguishing feature of the approach is the concept of *digital contract*. It is an agreement between contributor and platform about terms of work, quality management principles and rewarding. This contract may be as lightweight as commonly accepted in modern microtask markets (like Amazon Mechanical Turk), specifying that a contributor may pick tasks from common service pools when he/she is comfortable and as many as he/she can. However, this contract may also be rather strict, requiring that a contributor

should be available during the specified time and be able to process not less than specified number of tasks per time interval. Terms of this *digital contract* are essential for estimating the amount of resources available for a service and its capacity (including time perspective of the capacity). The necessity of this *digital contract* is caused by the fact that human resources are limited. In case of ordinary hardware managing cloud the cloud infrastructure provider can buy as many computers as needed, human participation is less controllable due to free will, therefore, attracting and retaining contributors can be a complex task. As a result, the abstraction of inexhaustible resource pool that is exploited in the provider-consumer relationship of current cloud environments turns out to be inadequate for human-computer cloud. A consumer should be informed about the human capacity available for his/her application to make an informed decision about revising *digital contracts* (making contribution to this application more appealing), or providing changes to their own service level agreements. This creates a competition between consumers for the available resources and finally will create a kind of job market where different digital contract statements will have its own price.

Important platform feature that is also provided by *digital contracts* is the support of different types of contributor allocation for a service. Namely, per task allocation and per time allocation. In the former case, the contract usually sets per task rewards and can optionally require that certain number of tasks be processed over some relatively large time span (e.g., 20 requests per day). While in the latter case, the contract enforces particular time spans of availability and therefore quick reaction of a contributor in the respective time span which is crucial for some applications.

B. *Contributor description*

When a contributor joins the cloud platform he/she provides two main types of information, that are very similar to pieces of application descriptor. Namely, competencies definition, and work conditions (Fig. 2). Competencies definition is made in terms of any ontology the contributor is aware of. For those contributors who cannot use ontologies there is another option, the definition of competencies is made iteratively via contributors' text description analysis followed by ontology-based term disambiguation. In any case, internally, each contributor is described by skills, knowledge and attitude, linked to shared ontology concepts.

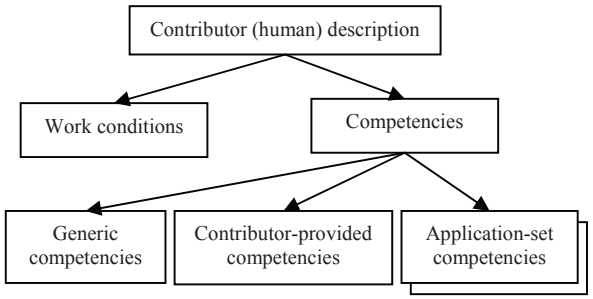

Fig. 2. Contributor's description structure

Contributor's competency definition is multi-layered. The first layer is provided by the contributor him-/herself, but additional layers are added by applications in which a contributor takes part. For this purpose, human resource management API available for application code can manage application-specific skills and qualifications, which also can be described in some ontology (application-specific or not).

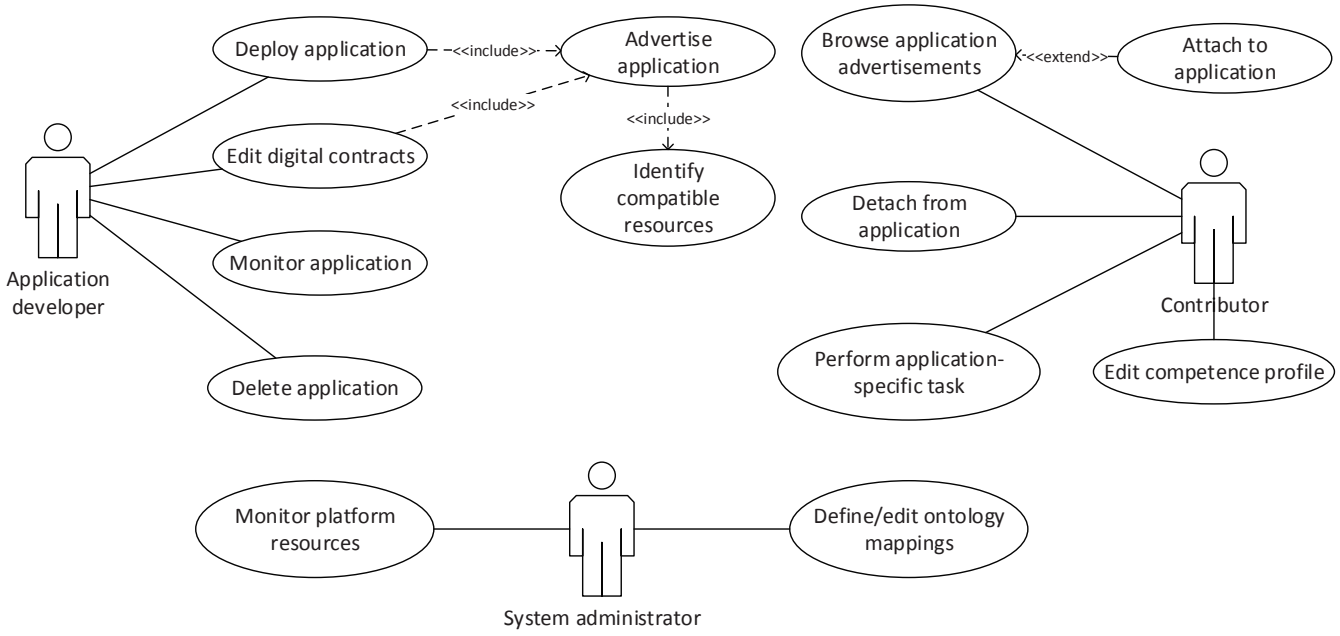

Fig. 3. Simplified use case diagram of the platform

Therefore, despite initial description of competencies may be rather terse, during the contributor's participation in different applications run over the platform it becomes more and more rich. And that alleviates further human resource discovery. Note, however, that each application can access its own contributor description layer, all the layers are visible only for deployment and resource discovery services of the platform.

Work conditions include preferred skills, as well as payment threshold, reactivity and availability limitations. This parameters are also similar to those included in *digital contract* template in the application descriptor. During application enquiry and application deployment its contract template is matched against contributors' work conditions. Moreover, this matching touches not only contributor's declared work conditions and one application contract (of the deployed application) but also other applications which contracts this contributor has already accepted, controlling overall responsibilities that are taken by the contributor and resolving possible conflicts.

C. Main use cases

Use case diagram showing actors and main use cases of the platform is presented in Fig. 3. This diagram connects platform functions involving core concepts introduced earlier and user categories.

Application/service developers can deploy application (which initiates advertisement process to identify human resources available to this application), edit digital contracts, monitor, and delete application (this releases all resources allocated for the application, including human resources). Editing digital contracts is necessary, for example, when application developer is trying to compete for resources with other applications by offering higher rewards. This effectively changes the descriptor of the deployed application (producing a new version of it) and leads to a new wave of advertisements. Monitor application use case generalizes various functions that are necessary for application developer and are common to many current PaaS, like reading usage statistics, logs and other. This also includes monitoring of the available human resources by each requirement type as well its prediction. Inner scenario of application advertising includes identifying compatible resources based on matching of resource definition (competence profile) and requirement specification.

Contributor can edit competence profile, providing the initial version of it or reflecting further changes, browse application advertisements routed to him/her (compatible with his/her competence profile and work conditions) with an option to accept some of them by signing digital contract and attaching to the respective application. Further, contributor can perform application-specific tasks and detach from application.

System administrator, besides monitoring the status of the platform (usage of hardware/human resources, communication channels throughput, platform services' health) can also do some activities in tuning the platform parameters. The diagram highlights only one kind of tuning which is editing ontology mappings. These mappings are used by the platform during identification of compatible resources (contributors) for advertising applications. The explicit mappings represent one of the ways for the platform to match competencies expressed in different ontologies.

IV. Service deployment scenario

This section describes the mechanism of human-based service deployment. When a contributor registers at the human-computer platform he/she is not immediately available for requests of all human-based applications that may run on this environment. However, he/she starts to receive *advertisements* of applications, which are being deployed (or are already deployed) based on the similarity of a declared competence profile (including additional layers created by applications a contributor participates) and applications' competence requests and correspondence of digital contract templates of the application and working conditions of contributor. These advertisements include description of service (its end-user functionality), type of human-based activities that are required for this service, the proposed rewarding scheme, specific performance evaluation techniques and so on. Based on the information contained in the *advertisement,* a contributor makes a decision if he/she will receive tasks from this application in future and what is acceptable schedule and maximum load. In other words, if a registered contributor agrees to contribute to the particular service a *digital contract* is signed specifying intensity of task flow, the rewarding and penalty details and quality measurement strategy. In general, there is many-to-many relation between applications and platform contributors, i.e, one contributor may sign *digital contracts* with several applications.

After a contributor and the platform (on behalf of particular application) signed a *digital contract*, application's requests for human operations are made available to the contributor. A contributor also can detach from an application (however, the mechanism and terms of this detaching can also be a part of *digital contract* to ensure the application provider can react to it accordingly).

As it was already noted, the process of service advertising is based on ontological representation of service requirements and human competencies and their matching.

V. Conclusion

The paper describes main principles of the proposed novel Platform-as-a-Service cloud environment aimed on simplifying the development and management of applications that require human information processing operations. Main specific problems (caused by the inclusion of human into the information processing loop) that influence the design of the platform are unpredictable availability and limited nature of human resources, as well as multifarious nature of human abilities (compared to typical hardware/software resources).

These problems are addressed by the distinguishing features of the platform, which are a) ontological representation of human competencies allowing intelligent and flexible resource discovery, b) digital contracts concept allowing to estimate resources currently available to application (and make predictions on the future capacity), and c) special tools for application developers for monitoring the

amount of available human resources, allowing to make decisions on rewarding policies and application's SLA for end users.

In general, the proposed resource management mechanism based on digital contracts makes the behavior of human-based applications more predictable and opens a way for building reactive human-based applications.

ACKNOWLEDGEMENT

The research is funded by the Russian Science Foundation (project # 16-11-10253).

REFERENCES

[1] C. Franzoni and H. Sauermann, "Crowd science: The organization of scientific research in open collaborative projects," *Res. Policy*, vol. 43, no. 1, 2014, pp. 1–20.

[2] L. Shamir, D. Diamond, and J. Wallin, "Leveraging Pattern Recognition Consistency Estimation for Crowdsourcing Data Analysis," *IEEE Trans. Human-Machine Syst.*, vol. 46, no. 3, 2016, pp. 474–480.

[3] M. Faulkner et al. "Community Sense and Response Systems: Your Phone as Quake Detector," *Communications of the ACM*, vol. 57, no. 7, 2014, pp. 66–75.

[4] Ushahidi, Web: http://www.ushahidi.com/.

[5] P. Meier "How Crisis Mapping Saved Lives in Haiti." Web: http://voices.nationalgeographic.com/2012/07/02/crisis-mapping-haiti/.

[6] Mell, P. and Grance, T. 2011 .The NIST Definition of Cloud Computing. *Recommendations of the National Institute of Standards and Technology*, NIST Special Publication 800-145.

[7] S. Distefano, G. Merlino, and A. Puliafito "SAaaS: A Framework for Volunteer-Based Sensing Clouds," *Parallel and Cloud Computing*, vol. 1, no. 2, 2012, pp. 21–33.

[8] G. Merlino, S. Arkoulis, S. Distefano, and C. Papagianni, A. Puliafito and S. Papavassiliou, "Mobile Crowdsensing as a Service: A Platform for Applications on Top of Sensing Clouds," *Future Generation Computer Systems*, vol. 56, 2016, pp. 623–639.

[9] C. Formisano, D. Pavia, L. Gurgen et al. "The Advantages of IoT and Cloud Applied to Smart Cities," *in 3rd International Conference Future Internet of Things and Cloud*, 2015, Rome, pp. 325–332.

[10] S. Dustdar and K. Bhattacharya, "The social compute unit," *IEEE Internet Computing*, 15(3), 2011, pp. 64–69.

[11] B. Sengupta, A. Jain, K. Bhattacharya, H.-L. Truong, and S. Dustdar "Collective Problem Solving Using Social Compute Units," *International Journal of Cooperative Information Systems*, vol. 22, no. 4, 2013.

[12] D. Androcec, N. Vrcek, and J. Seva "Cloud computing ontologies: A systematic review," *in Proc. MOPAS 2012: The Third International Conference on Models and Ontology-based Design of Protocols, Architectures and Services*, 2012, pp. 9–14.

[13] P. Bellini, D. Cenni, and P. Nesi, "Cloud Knowledge Modeling and Management," *Encyclopedia of Cloud Computing* / ed. Murugesan S., Bojanova I. Chichester, UK: John Wiley & Sons, Ltd, 2016, pp. 640–651.

[14] B. Livieri et al., "Ontology-based modeling of cloud services: Challenges and perspectives," *in PoEM (Short Papers) CEUR Workshop Proceedings*, vol. 1497, 2015, pp. 61–70.

[15] G. Guizzardi, Ontological foundations for structural conceptual models. Ph.D. thesis, Universiteit Twente, 2005.

[16] H. Ludwig, A. Keller, and A. Dan, et al, "Web Service Level Agreement (WSLA) Language Specification," January 28 2003. Web: http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf.

[17] D.D. Lamanna, J. Skene, and W. Emmerich, "SLAng: A language for defining service level agreements," *in Proc. of the 9th IEEE Workshop on Future Trends in Distributed Computing Systems-FTDCS*, 2003, pp. 100-106.

[18] D.G. Sampson and D. Fytros, "Competence Models in Technology-Enhanced Competence-Based Learning," *Handbook on Information Technologies for Education and Training*, International Handbooks on Information Systems, Springer Berlin Heidelberg, 2008, pp. 155–177.

[19] K.O. Lundqvist, K. Baker, and S. Williams "Ontology supported competency system," *International Journal of Knowledge and Learning*, 7 (3/4), 2011, pp. 197–219.

[20] V. Janev and S. Vraneš, "Ontology-based Competency Management: the Case Study of the Mihajlo Pupin Institute," *Journal of Universal Computer Science*, vol. 17, no. 7, 2011, pp. 1089–1108.

[21] S. Miranda, F. Orciuoli, and V. Loia, D. Sampson, "An Ontology-Based Model for Competence Management," *Data & Knowledge Engineering*, vol. 107, 2017, pp. 51–66.

[22] Unapproved Draft Standard for Learning Technology – Data Model for Reusable Competency Definitions. IEEE Unapproved Draft Std P1484.20.1/D5, Jan 2007.

[23] K. Rezgui, H. Mhiri, and K. Ghédira, "Extending Moodle Functionalities with Ontology-based Competency Management, *Procedia Computer Science*, vol. 35 (2014), pp. 570–579.

[24] Y. Telnov and I. Savichev, "Ontology-Based Competency Management: Infrastructures for the Knowledge Intensive Learning Organization," *in Biologically Inspired Cognitive Architectures (BICA) for Young Scientists*. Advances in Intelligent Systems and Computing, vol. 449, 2016.

[25] G. Paquette "An Ontology and a Software Framework for Competency Modeling and Management," *Educational Technology & Society*, 10 (3), 2007, pp. 1–21.

[26] M. Fazel-Zarandi and M.S. Fox, "An Ontology for Skill and Competency Management," *in Proc. of the 7th International Conference on Formal Ontologies in Information Systems (FOIS 2012)*, Graz, Austria. Web: http://www.eil.utoronto.ca/wp-content/uploads/km/papers/MFZ_Fox_FOIS_2012_CR.pdf.

[27] SPARQL Query Language for RDF. Web: https://www.w3.org/TR/rdf-sparql-query/.

[28] Geonames Geographical Database. Web: www.geonames.org/.

[29] J. Euzenat and P. Shvaiko, *Ontology Matching*, 2nd edition, Springer-Verlag, Berlin Heidelberg (DE), 2013.

[30] OWL 2 Web Ontology Language Document Overview (Second Edition). Web: https://www.w3.org/TR/owl2-overview/.