Decision Support Service Based on Dynamic Resource Network Configuration in Human-Computer Cloud

Alexander Smirnov, Maksim Shchekotov, Nikolay Shilov, Andrew Ponomarev
SPIIRAS
St.Petersburg, Russia
{smir, shekotov, nick, ponomarev}@iias.spb.su

***Abstract*—Recently, there is an upsurge of systems where a human plays not only a role of a consumer, but serves as an integral part of information processing workflow, providing some services to other parts of a system. The concept of a human-computer cloud aims to mitigate development of such systems by providing a unified resource management environment that could serve as a basis on which any human-based application could be deployed. One of the most important upper-level applications of such human-computer cloud environment is an ability to form dynamic networks of human contributors and software resources to solve *ad hoc* tasks given by the end user. This paper discusses main models, methods and algorithms leveraged for building such task-driven resource networks.**

I. Introduction

Large-scale distributed human-computer systems, where humans play not only a role of consumers of information and services, but also participate in their provisioning are becoming more and more numerous. Slightly different types of these systems have received a bunch of names: crowdsourcing, crowd computing, human computations, social machines, hCAS (hybrid collective adaptive systems) are some of them. These systems are currently applied to variety of information processing problems in which solely computational approaches still don't work well.

Common problem with the systems that require human attention and human input is that each of these systems usually needs a large number of contributors to function, while collecting this number of contributors may require significant effort and time. This problem is partially alleviated by crowdsourcing platforms (like Amazon Mechanical Turk, Yandex.Toloka etc), that provide tools for requesters to post tasks and an interface for workers to accomplish these tasks. However, existing platforms bear two main disadvantages: a) most of them implement only 'pull' mode in distributing tasks, therefore not providing any guarantees to the requester that his/her tasks will be accomplished, b) they are designed for mostly simple activities (like image/audio annotation). The ongoing project is aimed on the development of 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 resource management issues from application software. The proposed human-computer cloud (HCC) environment addresses all three cloud models: infrastructure, platform and software. Infrastructure layer is responsible for resource provisioning, platform layer provides a set of tools for development and deployment of human-based applications, and on top this environment there are several software services leveraging human problem-solving abilities.

Previous research on the HCC was concentrated primarily on the design and refinement of the foundational concepts and mechanisms of the HCC environment as well as on the development of some proof-of-concept human-based applications leveraging the functions of HCC. For example, see [1] for the general structure and use cases, [2] for description of Platform-as-a-Service for deployment of human-based applications and [3], [4] for a number of applications in tourism. Those applications implemented some pre-defined and fixed information processing workflows. It is quite typical and can be used in many contexts, but may be limiting in case of decision support systems. Decision-making often requires some experimentation and iterative (and interactive) exploration of the problem, when a decision-maker deepens his/her understanding of a problem, possible alternatives and various information that can affect the decision. Therefore, it can be hard or even impossible to foresee and implement *a priori* all possible services a decision-maker can find useful during such exploration.

To meet these specific requirements arising from decision-support applications, this paper focuses on an upper-level functionality provided by the HCC platform, namely, the ability to dynamically build networks of available (human and software) cloud resources to solve domain tasks given by the end user (usually, a decision-maker). The proposed solution is to give a decision-maker a toolset, which can help to build missing services via an automated composition of the existing ones. This functionality complements the PaaS layer of the HCC which allows to deploy various human-based applications (applications that leverage human abilities at some point of execution) and handles the situation when there is no service/application to solve the exact task the decision-maker has. Task analysis and network building are implemented in *René* service provided according to SaaS cloud model and leveraging the resource management mechanisms of the underlying layers of the HCC.

The rest of the paper is structured as follows. Section II presents discussion of the related work, connected to the

ISSN 2305-7254

problem of dynamic workflow formation. Section III briefly describes the organization of the HCC necessary to understand dynamic workflow methods. In Section IV the approach to task decomposition is presented, showing how the task provided by the end user is transformed to the set of sub-tasks assigned to various resources with a help of task ontology. Section V describes the proposed algorithm for distribution the subtasks among human-computer cloud resources.

II. Related Work

This section describes main developments in the area of automatic transformation of task hierarchies, and building possible decompositions and workflows from them.

The problem of decomposition of a complex tasks into subtasks is usually solved with a help of tasks and methods ontology, allowing to reuse problem-solving knowledge coding in with a help of standardized dictionary [5], [6]. This ontology defines roles played by different pieces of knowledge in the process of logical inference [7], [8], [9]. Like in the cited works, task decomposition in the proposed approach is also based on the task ontology. I.e., general and problem domain oriented tasks are described in the ontology, as well as their input and output values. However, unlike in the papers above, there are no explicit methods in the ontology, because it is assumed that a method is anything that the service provided by a resource does. So, instead of methods there are resources being some kind of "black boxes" exposing interfaces with semantically annotated inputs and outputs.

The problem of distributing a number of subtasks among resources varying by some parameters (quality, cost, performance) in order to maximize some overall quality function is also a very acute problem that raises in a variety of distributed systems (see, [10], [11]). The problem is computationally complex, therefore in most practical cases heuristics are used. In the areas of work distribution between robots or agents most similar to the one under consideration, one of the most common methods is the instantaneous task allocation method [12], focused on a dynamic uncertain environment. It is a *greedy* method which assigns the task to resources that currently provide the maximum "benefit" according to given priorities. However, it does not take into account that at some point all resources possessing the required competencies can be occupied, as a result, it is usually supplemented by some heuristics specific to the specific area of the application [13]. Also, mechanisms based on the theory of coalition games (e.g., [14], [15]), evolutionary algorithms (e.g., [16], [17]) and probabilistic models (e.g., [18]). In this paper, an algorithm based on the multi-agent modeling and elements of the coalition games theory is adapted to the task distribution in human-computer cloud.

III. Main Concepts of the HCC

This section provides a brief overview of concepts and architecture of the HCC, necessary to understand the role of decision support service deployed on top of it and the context it works with.

A. Architecture overview

Although NIST recommendation document [19] identifies five types of actors, the proposed HCC architecture adopts two most important of them (i.e., Cloud Consumer and Cloud Provider) and adds one new specific actor for humans who provide their resources via cloud environment. Therefore, following actors are identified:

Cloud consumers, who use the applications and services deployed in the cloud environment (and provided by Cloud providers). Further, this category of actors can be divided into End users and Service developers. This division is mostly determined by the kind (and a level) of services a consumer deals with. For example, when using the cloud for decision support in tourism, possible end users are travelers or tourism managers, because they use cloud services (mostly, on the SaaS layer) to solve domain specific tasks. Service developers use the services of the platform layer to create application services for the end users.

Contributors, i.e., citizens, who are available to serve as human resources in a HCC environment.

Cloud providers, individuals or organization who own and maintain the required hardware and software infrastructure provided to Cloud Consumers. This includes, for example, system administrators.

All the three models of cloud computing (IaaS, PaaS, SaaS) can be adapted to include human resources.

Infrastructure layer: Infrastructure layer unifies different types of capabilities: traditional computing and storage capabilities, sensing capabilities and human expertise capabilities. Contributors can join HCC and define the resources they can provide, time and load restrictions, a type of tasks they may participate. In the infrastructure layer resources (including human resources, or contributors) are not locked to some particular domain. Instead, they describe their competencies and possible kinds of activities using some of the available ontologies to leverage the resource identification phase that happens when some application that require human participation is deployed in the cloud environment. Ontology-based resource discovery service performs ontology search involving ontology matching techniques as necessary. Infrastructure layer management monitors contributor connections and disconnections, collects information about effectiveness of each contributor (separately for each skill a contributor is allocated by) and uses it in further allocation requests.

Platform layer: This layer consists of a set of multi-purpose utility services that can be leveraged for building applications relying on human expertise, and development tools, that are used to deploy and run human-in-the-loop services in the cloud environment.

Development tools of the platform layer allow to deploy services in cloud environment and to monitor them. Each service being deployed includes an ontology-based descriptor, specifying:

- building/configuration instructions;

- hardware and software requirements of the service (what platform services it relies on, e.g., database service, human workflow service, etc.);
- human resource requirements (if any), specifying what human skills and competencies this service need to function. These requirements are also 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;
- description of the service functions and entry points to be published in the application domain service repository and used by the ad hoc dynamic workflow service.

Typical interoperability scenario that is initiated in the platform layer during deployment is the following: the human resources connected to the cloud environment describe their capabilities using some problem-specific dictionaries. Each application/service that is deployed in this cloud environment contains a description of its requirements (including the requirements to the resources), which is expressed in terms of the most appropriate ontology selected (or even designed) by the application developers. It is very unlikely that human resources have used this exact ontology to describe their capabilities when connecting to the system. However, the advantage in using ontologies here is that due to the formal semantics inherent to them different ontologies can be matched. Hence, in the process of service deployment, human resources that are potentially able to fulfill the human requirements of the service are identified (despite the fact that they are not described initially in terms of the application ontology). Later, during the functioning of the service, the participants' description can evolve, because his/her performance in the capabilities required by the service (and expressed in terms of service's ontology) is recorded and processed. For each further service that is deployed in this environment, the process of aligning requirements with the capabilities of human resources becomes easier, as human resources definition becomes more and more detailed.

Software layer: This layer consists of a suite of (potentially human-based) services and applications designed for a particular problem area.

B. René: General Scenario

The *René* decision support service is an application, running on top of the human-computer cloud infrastructure exposed as a SaaS and leveraging some features of the platform (e.g., resource management and provisioning). Expected user of *René* is a decision-maker who passes some task specification to the service to build an on-the-fly network capable of performing the task. It should be noted, that *René* exposes an API, by which ontology-based structured representation of the task specification is passed. The problem of creating such specification (for example, as a result of text analysis) is out of the scope both of this paper and of *René* functions.

To decompose a task specification into a smaller tasks *René* uses a problem-specific task ontology, where domain tasks and their input and output parameters are described. Clearly, the decomposition is only possible when the task specification is written in terms of the same ontology. That is why a) task ontology is a common knowledge of *René* and decision-maker, b) if the ontology is not familiar to the decision-maker he/she may need some automatic helpers in the process of creating task specification (these helpers are out of the scope of the paper). The method and algorithm of the decomposition are described in detail in Section IV.

After performing the decomposition *René* tries to distribute the elementary subtasks among the available resources. The list of available resources is taken by API from underlying layers of the environment, which monitors all the contributor's connections and disconnections and software resource registrations. The resource management service under the hood treats human and software resources a bit differently. Human resources (contributors) describe their competencies with a help of ontology and receive *advertisements* to join human-based applications if their requirements are compatible to the declared competencies of the user. In this sense, *René* is one of these human-based applications and may only distribute subtasks to those contributors who agreed to work with it. Software services are made available to *René* by their developers and maintainers by placing a special section into deployment descriptor. All the resources available to *René* are modeled as agents in the process of subtask distribution that is described in detail in Section V. Practical assignment is also done via interfaces of underlying resource management layer, aware of the current status and load of the resources and terms of their digital contracts.

Finally, *René* monitors the availability of the resources (via the underlying layer) during execution of the subtasks and rebuilds the assignment if some of the resources fail or become unavailable.

IV. Task Decomposition

This section describes the proposed approach to task decomposition, which is the first step for building the resource network. Main operation that drives the decomposition is (a bit ironically) task composition, i.e. deriving chains of tasks connected by input/output parameters. Furthermore, task decomposition in this approach can be viewed as finding such composition of basic tasks defined in the ontology that is equivalent to the task given by the user.

A. General approach for task composition

For the purposes of decision support system, the structure of the task ontology is proposed, consisting of a set of tasks and subtasks, sets of input and output parameters of task, sets of valid values of parameters, as well as the set of restrictions of belonging subtasks tasks in the hierarchy of task

composition, parameters and sets of valid values parameters:

$$O = (T, IP, OP, I, E) \quad (1)$$

where T is set of tasks and subtasks, IP – set of input task parameters, OP – set of output task parameters, I – set of valid

parameter values, *E* – restrictions on membership of the sub-tasks of a task hierarchy task composition, the parameters of the task and domain parameters.

Unlike task decomposition ontology containing relationships between task and their subtasks in explicit form [20], task composition ontology is implemented so that the relationships are implicit. Such principle of ontology structure allows on one the hand to specifying task and subtasks in the same axiomatic form, and on the other hand to derive task composition structure by reasoning tools. Thus, the proposed ontology opens the possibility to describe a number of different tasks in the same form and after that to construct a number of their compositions using appropriate criteria.

For the purpose of task composition ontology development the ontology language OWL 2 is used. The ontology is expressed by *ALC* description logic, which is decidable and has PSpace-complete hardness of concept satisfiability [21] and ABox consistency [21] in the case when TBox is acyclic. In addition, SWRL-rules are specified for composition chain deriving. The main concepts of the ontology are "Task", "Parameter". The concept "Parameter" is used to describe a task semantically. The main requirement for TBox definition is to it doesn't contain cyclic and multiple definitions, it contains only concept definitions specified by class equivalence.

The task should have at least one input and one output parameter. The parameter taxonomy in the OWL 2 ontology is presented by a number of subclasses of the class "Parameter". The type of parameters related to their input or output role are defined by appropriate role construct. The corresponding OWL 2 syntax expression is the Object Property. In the ontology the appropriate object properties are "hasInputParameter" and "hasOutputParameter". The domain of the properties is "Task" and the range – "Parameter". Thereby the parameter could be input parameter of one task and output parameter of another. The task definition is expressed formally as follows:

$$T \equiv (\exists R.IP_1 \sqcap \exists R.IP_2 \ldots \sqcap \exists R.IP_N) \sqcap (\exists R.OP_1 \sqcap \exists R.OP_2 \ldots \sqcap \exists R.OP_N) \quad (2)$$

where T is the task, IP_i – the input parameter subclass, OP_i – the input parameter subclass, R – the appropriate role.

The proposed task definition (2) is used for task composition process because there is the notion that one parameter is the input for the one task and output for the other one. This knowledge is used to construct task composition by the SWRL rule. The corresponding SWRL rule specifies input and output parameter match condition in the antecedent and the result relationship in the consequent. For this purpose the object property "nextTask" is created which binds two tasks, where the domain is the previous task and range – the next one. Neither task is connected by the property explicitly. Thus, the rule of task composition can be expressed as follows:

$$\text{hasInputParameter}(?ta, ?p)\wedge \text{hasOutputParameter}(?tb, ?p) \rightarrow \text{nextTask}(?tb,?ta) \quad (3)$$

where hasInputParameter, hasOutputParameter, nextTask are the mentioned object properties, *ta* – the next task, *tb* – the previous task, *p* – the parameter.

The proposed rule (3) allows to deriving all task connections by the object property "nextTask". The example of task composition is presented in Fig. 1. The abbreviation "ip" and "op" denote input parameter and output parameters accordingly.

The advantage of the described approach is the possibility to easy expressing the tasks, dynamical deriving task compositions. The shortcoming are the possible deriving complexity and the lack of the support of alternative task compositions.

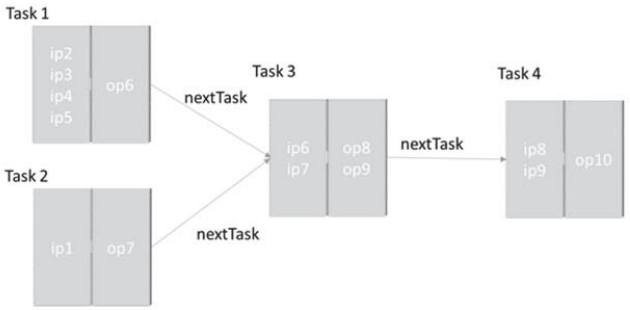

Fig. 1. Task composition structure

B. The example in domain "e-Tourism"

This section illustrates the proposed approach applied to building a tourist itinerary. Typical context of this task is that a tourist would like to see the most interesting attractions in the area in some constrained time and in most convenient way. Itinerary planning requires not only information about the popular attractions in the area of interest, but also user preferences as well as transportation options. There are currently various approaches to build itineraries both by software services and by humans. The proposed HCC offers a way to implement and deploy itinerary planning that leverages both software and human (contributor) resources for different subtasks.

TABLE I. SUBTASKS FOR ITINERARY CONSTRUCTION

Symbol	Subtask description	Input parameters	Output parameters
a_1	Collect context information	d_2, d_3	d_4
a_2	Search for attractions based on preferences	d_1, d_2	d_5'
a_3	Improve the list of attractions based on context information	d_4, d_5'	d_5'
a_4	Create candidate itineraries, respecting local context information	d_5'	d_6
a_5	Refine candidate itineraries	d_6	d_6

Formally, input parameters for the itinerary planning are (simplified): tourist (user) preferences (Preference), location of

a tourist itinerary (d_2), time for the itinerary being planned (d_3), context information (d_4), set of attractions (d_5), ranked list of attractions (d_5'), ranked plans of tourist itineraries (d_6). Parameters d_5 (set of attractions) and d_5' (ranked list of attractions) have the same structure, but different semantics. Subtasks for the itinerary construction, as well as their input and output parameters are shown in Table I. Note, the task a_5 has the same type of parameter (d_6) both as input and output. It makes sense as the workflow is evaluated not only based on the number of subtasks, but on time, cost and artifacts quality.

```
Class: Context_collection

    EquivalentTo:
        (hasInputParameter some Place)
         and (hasInputParameter some Temperature)
         and (hasInputParameter some Tour_duration)
         and (hasInputParameter some Weather)
         and (hasOutputParameter some Context)

    SubClassOf:
        Task

Class: Attraction_search

    EquivalentTo:
        (hasInputParameter some AttractionDB)
         and (hasInputParameter some Attraction_List)
         and (hasInputParameter some Place)
         and (hasInputParameter some Preference)
         and (hasOutputParameter some Ranged_Attraction_List)

    SubClassOf:
        Task

Class: Attraction_list_improvement

    EquivalentTo:
        (hasInputParameter some Attraction_List)
         and (hasInputParameter some Context)
         and (hasOutputParameter some Ranged_Attraction_List)

    SubClassOf:
        Task


Class: Tour_generation

    EquivalentTo:
        (hasInputParameter some Ranged_Attraction_List)
         and (hasOutputParameter some Tour_plan)

    SubClassOf:
        Task
```

Fig. 2. Task description using Manchester syntax

According to the model, the task of planning a tourist itinerary is determined by the output parameter "tourist itinerary". In the proposed example, the user can choose, for example, a variant with the input parameter "ranked list of attractions" from the possible subtasks, which will lead to the need to solve only one subtask a_4 (Create candidate itineraries, respecting local context information), or, a variant in which to solve the task it is necessary to solve all the subtasks defined above (discussed here). Based on the analysis of input and output parameters, it can be concluded that the subtasks a_1 (Collect context information) and a_2 (Search for attractions based on preferences) are independent and can be solved in parallel. The solution of the subtask a_3 (Improve the list of attractions based on context information) should be preceded by the solution of subtasks a_1 and a_2, and the solution of the subtask a_4 (Create candidate itineraries, respecting local context information) is the solution of subtask a_3 (Fig. 3). The subtask a_5 (Refine candidate itineraries) can be performed only the last.

The task description example related to this domain description is implemented is presented in Fig. 2 using Manchester syntax.

After the decomposition of the task, resources (represented by agents – see Section V) start to negotiate which resource will accomplish what subtasks. If the process of the negotiations succeeds, each task will be performed by one resource in the resulting schedule.

V. SUBTASK DISTRIBUTION

A. General approach of task distribution

The specifics of the distribution of tasks in cloud computing systems lies in the fact that the presence of a very large number of available computing resources, which are usually interchangeable (alternative) [10],[11]. The research is focused on the solution of specialized tasks (subtasks) that require certain competencies, which on the one hand narrows the range of resources capable of solving these subtasks, and on the other hand requires taking into account these competencies. Therefore, the algorithms used in this field cannot be used directly.

In the areas of distribution of tasks among the robots or agents that are most similar to those under consideration, the most common approach of instant distribution of tasks (instantaneous task allocation) [12],[13] focused on the dynamic uncertain environment. This approach involves tying tasks to resources that currently provide the maximum "benefit" according to the given priorities. This approach does not take into account that at some point all resources with the required competencies may be occupied. Thus, it is usually supplemented by some heuristics specific to a particular application area (see Section II).

Let, A – is a task, which contains several subtasks a_i:

$$A = \{a_i\},\ i \in \{1, \ldots, n\} \tag{4}$$

Let, O – is the vocabulary of competencies:

$$O = \{o_1, o_2, \ldots, o_m\} \tag{5}$$

Thus, the matrix of competencies required to accomplish subtasks can be defined as:

$$(ao_{i,j} \in \{0, 1, \ldots, 100\}),\ i \in \{1,\ldots, n\}, j \in \{1,\ldots, m\} \tag{6}$$

The set of human-computer cloud resources R is defined as:

$$R = \{r_1, r_2, \ldots, r_k\} \tag{7}$$

The set of resource characteristics (speed, cost, etc) C is defined as:

$$C = \{c_1, c_2, \ldots, c_l\} \tag{8}$$

Thus, each resource r_i is described by the following pair of competencies and characteristics vectors:

$$r_i = ((ro_{i,1}, \ldots, ro_{i,m}), (rc_{i,1}, \ldots, rc_{i,l})) \tag{9}$$

where $i \in \{1, \ldots, n\}$, $ro_{i,j} \in \{0, \ldots, 100\}$ – is the value of competency j of the resource i, and $rc_{i,j}$ is the value of the characteristic j of the resource i.

The solution of the task A describes the distribution of work among system resources and is defined as:

$$S_A = (s_{i,j}),\ i \in \{1, \ldots, n\}, j \in \{1, \ldots, k\} \quad (10)$$

where $s_{i,j} = 1$, if the resource j is used for solving subtask i, and $s_{i,j} = 0$ otherwise.

The objective function, which also performs normalization of various characteristics, is defined as follows:

$$F(S_A) = f(F_1(s_{1,1}, s_{2,1}, \ldots, s_{n,1}), \quad (13)$$
$$F_2(s_{1,2}, s_{2,2}, \ldots, s_{n,2}), \ldots,$$
$$F_k(s_{1,k}, s_{2,k}, \ldots, s_{n,k})) \rightarrow \min$$

Specific formulas for calculating partial assignment efficiency (F_i) can use values of resource characteristics (e.g., speed or cost) $rc_{i,j}$, as well as competence values of both resources ($ro_{i,j}$) and subtasks ($ao_{i,j}$).

The minimization must be performed with respect to the following constraints. First, each subtask must be assigned to some resource:

$$\forall i : \sum_{j=1}^{k} s_{i,j} \geq 1 \quad (14)$$

Second, assignment can only be done if the competency values of the resource are not less than the required competency values of the subtask:

$$\forall i,j,q : ((s_{i,j} = 1) \rightarrow (ro_{j,q} \geq ao_{i,q})) \quad (15)$$

B. Instantaneous distribution of tasks algorithm

Since the problem is NP-complete, it is not expedient to solve it by an exhaustive search method in a reasonable time (provided that a real-world problem is solved). As a result of the analysis of existing methods it is proposed to use the approach of instantaneous task allocation.

With regard to the problem, the algorithm based on the approach of instantaneous distribution of tasks is as follows:

1) Take the first subtask from the existing *ai,* and exclude it from the set of subtasks *A;*

2) Select such resource j from the available resources to satisfy all conditions and $F(S_A) \rightarrow \min$, where $S_A = (s_{1,1} = 0, \ldots, s_{1,j} = 1, \ldots, s_{1,k} = 0)$;

3) If a suitable resource is not found, assume that the problem is unsolvable (the system does not have a resource that meets the required competencies);

4) Repeat steps starting from step 4 until set A is empty (i.e. all tasks are assigned to resources).

C. Multi-agent distribution of tasks

There are two types of agents that are used to perform multi-agent modeling: the customer agent that is responsible for generating jobs and making the final decision, and the execution agents that represent the resources of the cloud environment and perform on-premises optimization for each resource. In the optimization process, agents form coalitions that change from step to step to improve the values of the objective function.

In the process of negotiations, agents of 3 roles are singled out: a member of the coalition (an agent belonging to the coalition), a leader of the coalition (an agent negotiating on behalf of the coalition) and an applicant (an agent who can become a member of the coalition).

At the beginning of the negotiations, each agent forms a separate coalition (SC, which has the structure of the SA solution), and is its leader. Suggestions of agents (tabular representation $F(s_{1,1}, s_{2,1}, \ldots, s_{n,1})$ are published in all available agents repository of information on the blackboard (Fig. 3). In this and subsequent figures, the agents are represented as black circles, and the coalitions are outlined with a dashed line.

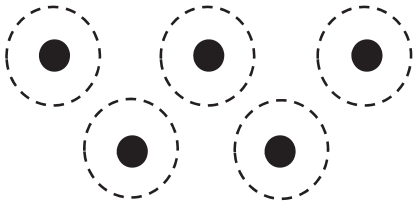

Fig. 3. Initial state of multi-agent task destribution

At each stage of the negotiations, the agents analyze the proposals of other agents, and choose those whose proposals can improve the coalition: to solve a larger number of subtasks or the same number of subtasks but with a better value of the objective function ($F(SC) > F(SC')$, where SC is the current coalition, SC' – possible coalition). Coalition leaders make appropriate proposals to agents, and the latter decide whether to stay in the current coalition or move to the proposed one. The transition to the proposed coalition is considered if one of the above conditions is met: the proposed coalition can solve more subtasks than the current one, or the same number of subtasks, but with a better value of the objective function (Fig. 4).

The process is terminated if at the next stage there is no changes in the composition of coalitions, after a specified time, when the permissible value of the objective function is reached.

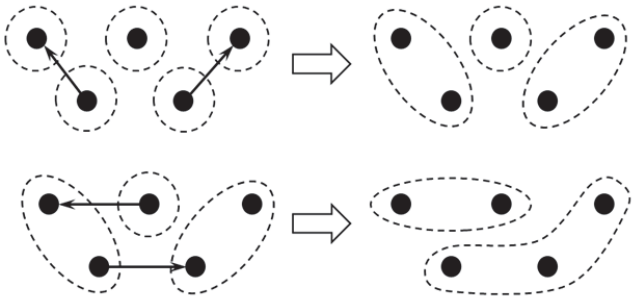

Fig. 4. Example of agent transitions to coalitions

VI. Conclusion

Several methods and algorithms for building "on-the-fly" decision support tools in the human-computer cloud have been proposed. Namely:

- A method and algorithm to decompose a task into subtasks based on task ontology. As a result of applying this algorithm, a (probably complex) task set by the decision-maker can be decomposed into several simpler subtasks that can be accomplished by resources – either human or software.
- A method and algorithm to distribute the subtasks among resources based on coalition games.

The proposed methods and algorithms are used to implement decision support service *René* on top of the HCC, which allows to decompose a task received from a decision-maker and dynamically build a resource network (consisting of both software and humans) for it, capable of solving the task. *René* can be used in variety of domain areas characterized by rapid changes of the situation for building flexible automated decision support tools automatically configured by decision-maker.

One of the primary directions for future work is to design helper mechanisms for a decision-maker to mitigate building of a formalized task description passed to decision support service.

Acknowledgement

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

References

[1] A. Smirnov, A. Ponomarev, N. Shilov, A. Kashevnik, and N. Teslya, "Ontology-based human-computer cloud for decision support: architecture and applications in tourism", *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, 2018, vol. 9(1), pp. 1–19.

[2] A. Smirnov, A. Ponomarev, T. Levashova, and N. Shilov "Ontology-based cloud platform for human-driven applications", *in Proceedings of the 21st Conference of Open Innovations Association FRUCT*, Helsinki, Finland, 6-10 November 2017, pp. 304–310.

[3] A. Smirnov, A. Ponomarev,T. Levashova, N. Teslya, "Decision support in tourism based on human-computer cloud", *in Proceedings of the 18th International Conference on Information Integration and Web-based Applications & Services (iiWAS2016)*, Singapore, 28-30 November 2016, pp. 127–134.

[4] A. Smirnov, A. Ponomarev, N. Teslya, and N. Shilov, "Human-computer cloud for the smart cities: tourist itinerary planning case study", *Business Information Systems Workshops, 20th International Conference on Business Information Systems (BIS 2017)*, Poznan, Poland, 28-30 June 2017, *LNBIP*, vol. 303, pp. 179–190.

[5] B. Chandrasekaran, J.R. Josephson, and V.R. Benjamins, "Ontology of tasks and methods", *in Proceedings of the Eleventh Workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*. Inn, Banff, Alberta, Canada, 1998, Web: http://spuds.cpsc.ucalgary.ca/KAW/KAW98/chandra/.

[6] D. Dou, H. Wang, and H. Liu, "Semantic data mining: A survey of ontology-based approaches", *2015 IEEE International Conference on Semantic Computing (ICSC)*, pp. 244–251.

[7] J.U. Kietz et al., "Semantics inside!" But let's not tell the data miners: intelligent support for data mining", *European Semantic Web Conference*, 2014, pp. 706-720.

[8] S. Balakirsky et al., "Towards a robot task ontology standard", *ASME 2017 12th International Manufacturing Science and Engineering Conference collocated with the JSME/ASME 2017 6th International Conference on Materials and Processing*, 2017, pp. V003T04A049-V003T04A049.

[9] E. Coelho and G. Lapalme, "Describing reusable problem-solving methods with a method ontology", *in Proceedings of Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'96)*. Catelonia, Spain, 1997, Web: http://ksi.cpsc.ucalgary.ca/KAW/KAW96/coelho/kaw.html.

[10] D. Ergu et al., "The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment", *The Journal of Supercomputing*, 2013, vol. 64, issue 3, pp. 835–848.

[11] Y. Kong, M. Zhang, and D. Ye, "A belief propagation-based method for task allocation in open and dynamic cloud environments", *Knowledge-Based Systems*, 2017, vol. 115, pp. 123–132.

[12] P. Sujit, G. George, and R. Beard, "Multiple UAV coalition formation", *in Proceedings of the American Control Conference*, 2008, pp. 2010–2015.

[13] M.H. Kim, H. Baik, and S. Lee, "Resource welfare based task allocation for UAV team with resource constraints", *Journal of Intelligent & Robotic Systems*, 2015, vol. 77, issue 3-4, pp. 611–627.

[14] P.S. Pillai and S. Rao, "Resource allocation in cloud computing using the uncertainty principle of game theory", *IEEE Systems Journal*, 2016, vol. 10, issue 2, pp. 637–648.

[15] Y. Zhang and L.E. Parker, "Considering inter-task resource constraints in task allocation", *Autonomous Agents and Multi-Agent Systems*, 2013, vol. 26, issue 3, pp. 389–419.

[16] J. Yang et al., "Task allocation for wireless sensor network using modified binary particle swarm optimization", *IEEE Sensors Journal*, 2014, vol. 14, issue 3, pp. 882–892.

[17] N. Gülpınar, E. Çanakoğlu, and J. Branke, "Heuristics for the stochastic dynamic task-resource allocation problem with retry opportunities", *European Journal of Operational Research*, 2018, vol. 266, issue 1, pp. 291–303.

[18] A.W. Palmer, A.J. Hill, and S.J. Scheding, "Modelling resource contention in multi-robot task allocation problems with uncertain timing", 2017, Web: https://arxiv.org/abs/1607.04358.

[19] P. Mell and T. Grance, "The NIST definition of cloud computing", *Recommendations of the National Institute of Standards and Technology*, Special Publication 800-145, 2011.

[20] R.K.L. Ko, E.W. Lee, and S.G. Lee, "BusinessOWL (BOWL) - a hierarchical task network ontology for dynamic business process decomposition and formulation", *IEEE Transactions on Service Computing*, vol. 5, issue 2, 2012, pp. 246–259.

[21] F. Baader, M. Milicic, C. Lutz, U. Sattler, and F. Wolter, "Integrating description logics and action formalisms for reasoning about web services", *LTCS-Report 05-02*, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2005, Web: http://lat.inf.tu-dresden.de/research/reports.html.