Human-Machine Collective Intelligence Environment for Decision Support: Conceptual and Technological Design

Alexander Smirnov, Andrew Ponomarev
St.Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences
St.Petersburg, Russian Federation
{smir, ponomarev}@iias.spb.su

***Abstract*—The paper describes a conceptual and technological design of a novel class of environments, providing means for leveraging collective intelligence of *ad hoc* human-machine teams for decision support. The paper describes theoretical background used for creating human-machine collective intelligence environment, principles guiding the design and foundational technologies. The core of the proposed environment is an ontology-based representation of the decision-relevant information that can be processed by both human and machine participants. The proposed environment can be used for decision-making support in a variety of domains characterized by high levels of uncertainty and dynamics (emergency, natural disaster, government and business scenarios).**

I. Introduction

The complexity of many real-life problems (especially, in managing complex systems) is significantly higher than the complexity that is addressable by purely automatic tools and systems. That is why human is still an important part of many complex information processing workflows. However, the availability and the speed of information processing of humans are low compared to software and hardware. With the development of global communication networks it has resulted in the creation of crowdsourcing and crowd computing platforms, consolidating human resources and providing them on the on-demand basis. There are two major trends in the evolution of globally accessible dynamic collaboration. First, crowdsourcing is mostly about solving simple problems (but in large quantities); adapting crowdsourcing for complex work is an active research issue [1]–[3]. Second, there is a growing understanding that novel forms of human-machine collaboration are possible, resulting in new types of human-machine collective intelligence [4], [5].

One of the types of intelligent activities that normally requires human intelligence is decision-making. While in many areas where the situation can be described by a relatively small number of parameters and there is quite accurate model of the system dynamics operative decision-making can be done automatically (essentially it is the domain of automatic control – various brake anti-lock systems, autopilots etc.). However, in a much wider range of problems, the list of parameters describing the situation is huge, many of them are unobservable, there is no formal model connecting these parameters, and often no predefined list of alternatives (possible controls). Besides, there may be some accountability issues, and there is no agreed upon framework for resolving accountability issues for decisions taken by automatic systems. In such settings, decision-making requires human experience, ability to generalize and infer information from tacit signs. In other words, decision-making remains human's responsibility, performed with an extensive (and increasing) help of software tools.

On the other hand, current developments in the AI area (especially, in natural language processing, formal reasoning and multi-agent systems) provide a solid background for collaborative human-machine systems where intelligent endeavor is shared between heterogeneous entities acting collaboratively and in a coordinated way.

An important aspect that distinguishes this paper is the *ad hoc* nature of the created collectives. In some sense, it derives from crowdsourcing systems, where a problem is given to an undefined community. However, in most crowdsourcing scenarios, first, there are no interactions between participants, they don't have to establish connections, second, there are no seamless integration of software tools (other than problem specific human input processors). There are some exceptions to that, they are described in more detail in Section II.

The functioning of the ad hoc teams (not necessarily in the Internet) is an interesting problem from the organization research perspective [6], [7]. There are mechanisms and practices inherent to such systems. Understanding these mechanisms and practices is important to provide computer support of collective intelligence in *ad hoc* teams. Therefore, the paper briefly outlines some of the most important results that influence the design of the environment.

The paper describes theoretical background used for creating human-machine collective intelligence environment, as well as some aspects of technical implementation. It is also shows how different parts of such system interact during the collective decision support.

ISSN 2305-7254

The structure of the paper is following. Section II describes some related research areas that influence the concept and design of the human-machine collective intelligence environment, as well as competing developments. Section III outlines the outcomes of the general coordination theory findings that solve as a foundation for the environment. Section IV describes the core design of the human-machine collective intelligence environment and its mechanisms.

II. Related Work

This work is especially related to two lines of research. The first line grows from organization research, as well as sociological and psychological sciences, where the processes of *ad hoc* teaming are studied and principles of *ad hoc* teams are identified. The second line represents other attempts to create environments for *ad hoc* teaming and human-computer collaboration. In this section, we discuss the most relevant results from the both lines of research.

A. Socio-psychological research of dynamic teams

Dynamic creation of (relatively) short-lived teams is typical in some areas of human activity. We argue, that the findings of socio-psychological research of mechanisms allowing efficient composing and functioning of such teams can be useful for any computer-based environment supporting such teams (although, not all of them are directly applicable).

There are several important publications in this area. The authors of [6] analyze the coordination processes of medical trauma center where fast-response and error-free activities are essential requirements. Teams there are collected from medical specialists of various specializations (a surgeon, an anesthesiologists, nurses) in a temporal basis (one shift) and effective coordination between them is crucial for ensuring the best possible treatment for incoming patients. The authors find two categories of coordination mechanisms employed in such organizations: expertise coordination practices (used most of the time, on the habitual patient's trajectory – reliance on protocols, plug-and-play teaming, community of practice structuring, knowledge sharing), and dialogic coordination practices (used for a problematic patient's trajectory – epistemic contestation, joint sense-making, cross-boundary intervention, protocol breaking) [6].

The paper [7] analyzes coordination mechanisms of a filming crew. Again, filming crew is usually collected for relatively short period of time from people most of whom previously didn't work together. However, each part of a filming team has a specific role and has understanding of the role's responsibilities, as well as typical responsibilities of the roles he/she tightly collaborates with. The authors describe the mechanisms that members of a filming crew use on a day-by-day basis to concretize the relatively general understanding of responsibilities to particular actions, as well as to establish connections between people playing 'neighboring' roles in a crew.

Both papers notice role-based coordination mechanisms with relatively flexible contents associated to the roles.

B. Computational environments and human-computer collaboration

Computational environments leveraging the resources of loosely connected people interacting via Internet usually are discussed in the scope of crowdsourcing or crowd computing. In today's crowdsourcing work is rarely collaborative, participants usually do their tasks independently and then their results are processed with some computer algorithm (to generalize or to check the quality). Besides, individual tasks are usually rather simple (like describing a single picture, recognizing something on an image etc.). However, in crowdsourcing research there are continuous attempts being taken to adapt crowdsourcing to more complex problems and (these trend are related) to develop collaborative workflows [8].

It has also been shown, that for complex problems pre-programmed workflows are too limited (e.g., [3]), therefore, the potential of crowdsourcing for complex work is tightly connected with the mechanisms of dynamic team formation, workflow adaptation. The first experimental crowdsourcing systems where human participants were able to change the initially proposed workflow appeared in almost ten years ago [9], but the problem is getting the closest attention of the research community only recently. Particularly, in the works [2], [3], [8] where the limitations of workflow-based solutions are studied and the ways to overcome these limitations with a help of dynamic organizations from members of the crowd (the so-called "flash organizations" [2]) are proposed. Interestingly, these works also acknowledge the results from socio-psychological research and build upon it. While the concept of "flash organization" represents an important step in understanding how crowd computing can be applied to complex problems, it deals only with human participants. In this research, however, we are building an environment where heterogeneous agents (human and software) would be able to collectively decide on the details of the workflow.

Problems closely to the problem of collective decision-making (including human-machine teams) are also present in many publications in the area of computer-supported collaborative work (e.g., [10], [11]). The results of the Dicode project implemented within the framework of the European FP7-ICT program [12], [13] deserve special attention. In the framework of the project, in particular, an ontological presentation of the argumentation process and a number of visual tools for working with a thus formalized set of interrelated arguments are proposed. The importance of these results lies in the fact that for sharing information about the problem by experts and software agents, it should have a structured representation, one of the options of which is a graph of the relationship of arguments. However, explicitly encoding all the arguments may be too difficult.

The problem of intelligent human-machine 'teamwork' is also posed in the context of modern production systems (the so-called cyber physical production systems (CPPS)). E.g., paper [14] proposes a management portfolio matrix for examining the feasibility of optimal collaboration between humans and cyber-physical resources. The optimal collaboration refers to the exchange of knowledge, reciprocal learning, and interaction of human and CPPS in smart factories. Further, paper [15] approaches the collaboration of human and CPPS in problem-

solving from the angle of complementarity whereby "human competences" and "CPPS autonomy" together derive supplementary capability and reciprocal learning. It proposes an ontology for reasoning out the competence questions, i.e. in which situation and under which conditions human and/or CPPS is dominant or eligible to solve a problem. However, these works are focused mostly on a closed environment of a production system.

A high-level paper [4] summarizes what is required to support human-machine collectives and current technological limitations for building them. In particular, the authors underline the need for the flexible autonomy of the agents, agile team-building, incentivization and accountability. The proposed environment account for all these prerequisites.

III. Background From Organization Research

This section discusses some important outcomes from the organization research, coordination theory [16] and socio-psychological research that influenced the design of the proposed environment.

1) Research of social systems and current practices in designing such systems agree that the elements of self-organization are critical, especially when solving complex problems. Despite the fact that there are a number of works close to the proposed, there are currently no solutions supporting the collective intelligence formed by artificial intelligent software services and people.

2) One of the most important roles in the functioning of dynamic self-organizing teams consisting of people is played by social norms and the adaptation of the behavior policies of team members to follow these standards. Accordingly, when software services are included in the team, it should be ensured that they can read and interpret social norms, as well as follow them. Moreover, since software services can be created by various users of the environment, it would be rational to separate the layer of implementation of social norms from the code of software services directly, making it part of the general infrastructure of the environment.

3) Another important feature of self-organization in dynamic human groups is role-based coordination [2], [3]. The idea that role-based coordination is a nearly optimal mechanism for *ad hoc* teams. Especially, allowing non-rigid role responsibilities structure, becoming more concrete and precise as collaboration moves on [2], [6], [7]. Interestingly, that *ad hoc* teaming is the most effective, when there is some well-understood by different people set of roles (even if these roles are not strict).The concept of the role, therefore, should also be provided by the environment, as well as the mechanism of "switching" the software agent between the roles (if possible) and clarifying the specific content of the role within the framework of this team [17].

4) The manifestation of team self-organization can be structured using patterns [18], structures and coordination schemes that participants come to under certain conditions.

5) A formal representation of problem-related information is necessary, allowing one to track the cause of certain arguments or results (provenance) [4], [12], [13], which is "understood" by both the participating people and the software agents. Therefore, information exchange infrastructure should not only allow all participants to get relevant information, but also to support accountability and provenance.

6) The environment should provide some incentive management mechanism allowing to reward participants [4]. It is crucial due to the openness of the environment. For example, in smart manufacturing systems it may not be as important, because all the infrastructure is already integrated and dedicated to the cooperation.

IV. Collaborative Environment Design

The purpose of the environment is to leverage the coordination procedures between heterogeneous agents, allowing information exchange on two levels: information concerning the problem being solved (available data, opinions, arguments and models), and process information (role distribution, responsibilities and so on).

The primary goal is to support cooperation of relatively short-lived (hours to several days) ad hoc teams, that is why much attention is paid to the process of forming a team (this process is not only important because the set of participants have very high impact on the efficiency of a cooperation, but also because team formation is relatively large part of a whole lifecycle of the team).

The distinguishing features from another systems for organized complex work (like [2]) is maintaining the structured representation of the problem and process information, allowing software agents interpret current situation and participate in the process of preparing the solution.

It should be noted, that the environment is inherently dedicated to decision support problems. Therefore, the design is influenced by decision-making methodologies (e.g., [19]–[21]) and the workflow implemented by a team mostly corresponds to a typical decision-making process.

We describe the environment design in two levels. First, conceptual design, which shows what are important concepts of the environment and how they interact during the work. Second, technological design, showing how (by what technologies and software components conceptual design is implemented).

A. Conceptual design

There are following principal actors differentiated by the environment design: end-user (decision-maker), participant, and service provider. End-user (decision-maker) uses the environment to get help in making a decision. He/she describes the problem and posts so that the problem description is visible to a specified community. Participant is an active entity (human or a software service) working on a problem given by the end-user. Finally, service provider develops, integrates to the environment, and supports software services that can act as participants working on some problem given by the end user. Service provider is also responsible for the deployed services, assuaging the problem of service accountability.

The core entities involved in most of the processes taking place in the environment are *problem* and *team*. Problem is introduced by an end-user and then is addressed by a *team* of participants. The description of a problem has a complex structure and representation. First of all, it contains not only information, specified by the end-user (initial statement), but also includes all the information produced by the team. So, during team's activity, the problem becomes more and more detailed. Second, to enable (at least, partly) effective interpretation by software agents, problem description is represented in a semi-structured way. In particular, machine readability is achieved via using ontologies. It is a crucial point in design. Since their introduction to the IT industry during Semantic Web initiative, ontologies proved themselves both quite efficient in solving interoperability problems and quite non-friendly for non-expert users. Still, we argue that for mixed human-machine teams, the ontologies provide a kind of *lingua franca*, acceptable by both human and software participants of the process. However, to make the use of ontologies easier for people, the environment makes the use of ontologies as implicit as possible by relying on three techniques:

- Implicit ontological representation of the structure of problem information. It means, that a human participant sees the problem description as a set of constraints, alternatives, criteria, alternative evaluations, arguments and so on, in other words, in terms that are widely used in decision-making literature and, in particular, by nearly every decision-making methodology. However, "under the hood" this structure is encoded with the decision-making ontology, therefore, software agents are also able to analyze the problem structure.
- Natural language processing. Using advances in this area it is possible to infer the role of some information pieces, its relationship with the goal and/or some line of argumentation and so on.
- GUI-based nudging participants to encode problem structure in an ontology-compatible way. Digital nudging [22] is a technique of growing importance, due to the fact that currently so many choices people do by interacting with virtual environments. In particular, nudging is quite effective alongside with the natural language processing of problem information, as it allows to ask user to disambiguate some terms and relations in the problem information.

The environment defines two basic ontologies, representing different aspects of the collaborative decision support:

- Decision-making ontology. This ontology defines main concepts that are used during decision-making (criterion, alternative, evaluation etc.) and interaction between them. The ontology is based on the analysis of existing decision-making methodologies and has been built in such a way to support majority of them.
- Collaboration and coordination ontology. It defines the concepts used in distributing work among team members (role, responsibility, dependency etc.).

The use of these ontologies allow artificial agents to 'understand' the processes taking place in the team and contribute to them. However, for the ontology-based decision-support agents, there is also a possibility to define an application ontology and map it to the decision-making ontology. By this process, some parts of the problem situation become connected to the general decision-making terminology. For example, in the problem of building a tourist route (e.g., in tourist support applications), an entity of a problem ontology 'route' becomes connected to the entity 'alternative', therefore, automatically enabling all the software team members to treat routes as alternatives (e.g., in visualizing them or building Pareto optimal set [23]). Besides, it allows problem-specific services to also contribute to the decision-making (e.g., estimate typical traffic of the route, its length).

The way problem information becomes richer and grows via interaction of agents, to some extent resemblances to stigmergy [24] and intelligent systems based on the principle of blackboard interactions. Especially, with research on ontology-based smart space technology [25]–[28]. The difference is that the smart space technology mostly considers intelligent collaborations of agents in some physical neighborhood (e.g., one room [25]). However, in the proposed environment, smart space represents a virtual space dedicated to solving a particular decision-making problem. At the same time, as the progress of a team on the problem is reflected by structural changes of ontological description of the problem, it allows to leverage ontology-based publish-subscribe mechanisms to intelligently perform some actions by software agents in response to particular situations during problem-solving.

Another core entity, as it was already mentioned, is a *team*. Team in the context of the environment is defined as a heterogeneous group (consisting of human participants and software services) working towards solution of a particular *problem*. Each problem has a team dedicated to it. Obviously, a participant may be a member of several teams, or not be a member of any team.

Initial team formation is based on the same principles that are used in most of the crowdsourcing platforms and knowledge networks (e.g., [29]): each participant has a profile describing key specializations, problem-solving history, as well as the history of previous collaborations (with mutual evaluations). There is a massive list of publications why each of these components of the profile is necessary and how it affects the efficiency of teaming. The initiative in this process is mixed in the sense that a contributor should send a proposal to the end-user, consisting of one or more team members (proposal may include several participants that already have some positive experience of working together), and end-user has to collect the initial team. However, decisions of the both parties – participants and end-users – are assisted by environment. The participants may choose to receive recommendations in case some problem touching his/her area of competences is posted. On the other hand, end-users may explore the description and history of all the participants mentioned in the proposals.

Due to much uncertainty typically associated with decision-making, it is often the case that during the work on the problem

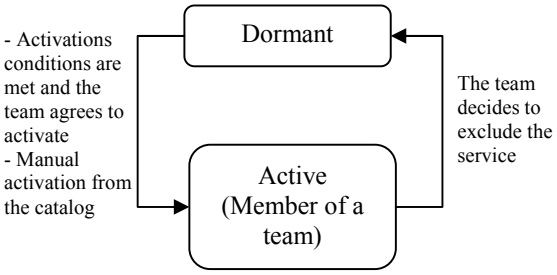

Fig. 1. Service states with respect to a team

team understands that it lacks some competencies or resources. Therefore, the team may create a new resource requirements, that are registered in the environment and resolved in a manner, similar to the initial team formation process (participants have to actively apply for the positions in the team, however, both sides are assisted by the environment mechanisms).

It should be noted, that it doesn't fully apply to the software participants (services). As the throughput of software services is not as limited as the throughput of humans, and the execution is relatively cheap, software services are passively connected to any team and by the mechanisms of the environment (ontology-based publish-subscribe) are watching the processes taking place with the problem. There are two states a software service can be in with respect to a team (Fig. 1): dormant and active. Initially, all services are in the dormant state and are waiting specific conditions during the problem-solving. If these conditions defined by a particular service are met, the service tries to activate, describing its purpose and terms of use. If the team agrees that the service is useful for the problem, the service is allowed to activate (change state to active) and become a member of the team. Otherwise, the service remains dormant. Active service may also be transferred to the dormant state by a decision of the team. Besides, the services can be accessed via a service catalog and activated manually by team members.

Active services can be used by the members of a team. The mechanics of their usage depends on the kind of a service. There are following types of services:

- Problem-solving service;
- External tool and database access service.

Problem-solving service accesses the problem information described in the form of ontology and natural text and can actively add information pieces to it. An example of such service is a statistics-based question answering service – if it detects a question about some facts (e.g., "How many people die from tuberculosis in the World in one year?") and can answer it in some form it adds an answer to the question. Another example is a service that derives from the problem information a current set of alternatives and their evaluations, builds a Pareto optimal set and adds it to the problem information.

External tool and database access services in activated form only provide an access to a specified resource. For example, if a team need epidemic database, it can activate the service that grants access to this database and use it for queries.

Simultaneously two processes take place when team works on a problem: *solution preparation* and *decision support (re)organization*. Both of these processes are supported by mechanisms provided by the environment. Solution preparation is main productive process, during which problem is enriched with new information and artifacts created by team members. General scheme of the solution preparation process (Fig. 2) is driven by decision support methodologies (e.g., Simon's model [19], DECIDE [20], or GOFER [21]). It should also be noted, that these methodologies provide the general scheme of the solution preparation, main stages that have to be performed, while execution of each stage is done via activities relevant to the problem at hand. For example if the environment is used by an organization to decide which motivation policy to implement for the workers, then the stage of Considering all the alternatives may contain a task of identifying relevant scientific publications about the effectiveness of motivation policies. The necessary activities (contents of the respective stage) are defined and enacted by the team itself, the environment allows to track these activities and connect their results with the problem definition and the concepts of decision-making ontology.

Most of the modern decision making methodologies include the evaluation phase (for example, this is the case with the DECIDE methodology – it defines the same stages as depicted in Fig. 2, but also Evaluate stage, responsible for the letter E in the name), however, it turns out that when the decision making support is 'outsourced' to an external collective, the collective actually cannot evaluate and monitor the results of the decision. Therefore, while this stage is perfectly valid for the decision maker, it is not supported by the environment.

Physically, the process of solution preparation can be viewed as adding information to the problem definition initially provided by the end-user (in the smart space). First, by adding explicit criteria and constraints (during the activities of initial stages – Define the problem and Establish the criteria), then by adding alternatives and their evaluations (later stages). The result of this process is fully detailed description of a problem situation, weighted alternatives and their estimated

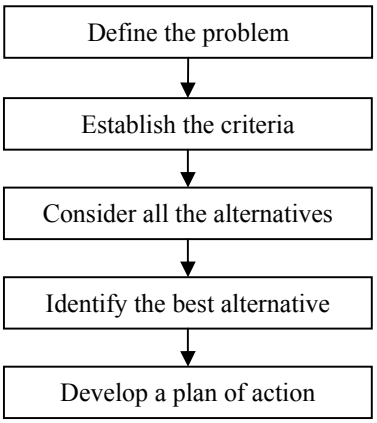

Fig. 2. Scheme of solution preparation process

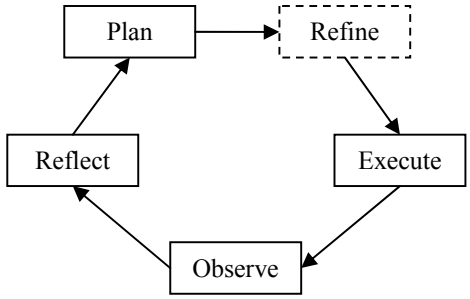

Fig. 3. Scheme of decision support (re)organization process

consequences – accepted by the end-user.

Decision support (re)organization process represents all the activities aimed on planning and organization of team work (e.g., deciding whether additional resources are required, assigning team member responsibilities, setting task deadlines and identifying new tasks to be solved in order to reach the goals of the whole process). Complex activities cannot be fully planned (and the literature review supports that), therefore, the team can adapt the plan as the solution process advances. General scheme of the decision support (re)organization process is shown in Fig. 3. During the Plan activity the team (or a responsible member of the team) builds a list of activities and assigns responsibilities using the features of the environment. If it is necessary, some plan items should be refined (this is especially the case when there are software services assigned to them). The planned activities are executed (this is actually a solution preparation process, described earlier); the results are observed and reflected on. If during the execution it turns out that some additional activities have to be performed, then the plan is adjusted and the cycle continues.

B. Technological design

Technologically the environment (Fig. 4) is implemented as a web application where each problem (and, therefore, a team) has its own workspace, supporting both informal text-based communication (similar to popular collaboration environments like Slack [30] or Mattermost [31]), but also have integrated 'nudging' modules allowing to tie the information into decision-making ontology formalized view, making it accessible for software services. Besides, humans can also browse the problem information in this structured ontology-based view and edit it if it is necessary.

The 'nudging' functionality is implemented as a set of Discourse Support Components, that maintain both the structured (ontology-based) and unstructured problem description and constantly perform mapping between them. Besides, these components provide an interface for other components of the system to access the contents and requirements of the team.

Team Member Recommender is responsible for performing assistance on finding team members for problems. It issues recommendations to participants who fit the active teams according to the implemented fit estimation models.

Software services are integrated into the collective intelligence environment in the following way. Service provider registers the service by specifying activation context, terms of usage and runnable image of a service (Software Service Image Repository component). The description is encoded via SPARQL-based declarative language to enable ontology-based subscription mechanisms. In order to preserve confidentiality of the team work, the services are run in an isolated way. Until their presence is accepted by the team, watching is done by the environment (on behalf of a service, particularly, by Team Activity Monitoring component). After the activation condition has been fired, a service instance is created from the runnable image and run in a new isolated container without internet access making sure that details of the problem will not be accessible by service provider who is not member of a team.

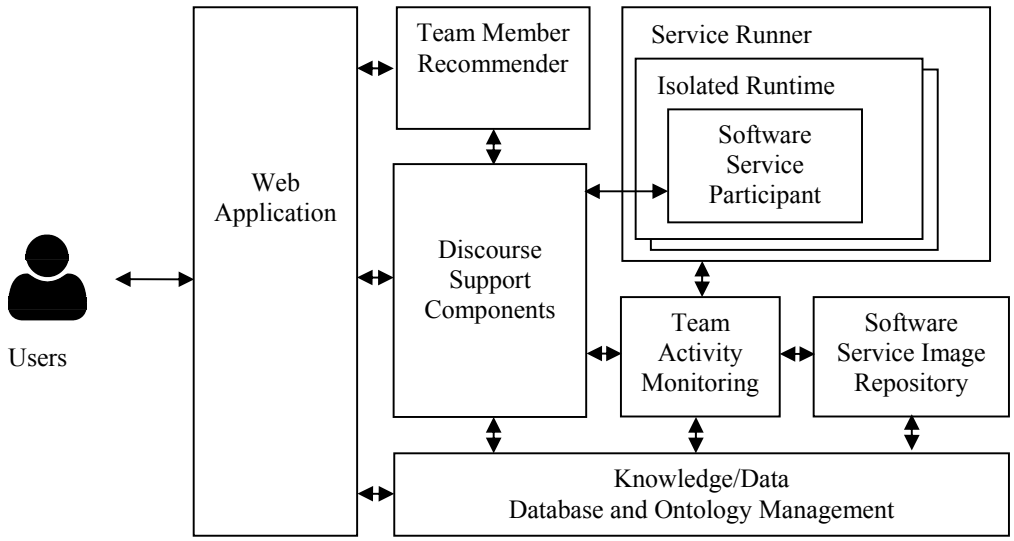

Fig. 4. Technological design of the environment

V. Conclusion

The paper describes a conceptual and technological design of a novel class of environments, providing means for human-machine collective intelligence for decision support. The proposed environment is rooted in modern research in ad hoc team coordination and ontology-based smart spaces technology.

The core of the proposed environment is an ontology-based representation of the decision-relevant information that can be processed (and augmented) by both human and machine participants. The ontology-based representation is built via the combination of natural language processing and GUI-based nudging participants to precisely connect information to the ontology-structured description. The integration of software services is implemented with a help of ontology-based smart space technology.

The proposed environment can be used for decision-making support in a variety of domains characterized by high levels of uncertainty and dynamics (emergency and natural disaster response, government and business scenarios).

Acknowledgment

The research is funded by the Russian Science Foundation (project # 19-11-00126).

References

[1] A. Kittur, B. Smus, S. Khamkar, and R. E. Kraut, "CrowdForge: Crowdsourcing Complex Work," in *Proceedings of the 24th annual ACM symposium on User interface software and technology UIST '11*, 2011.

[2] M. A. Valentine, D. Retelny, A. To, N. Rahmati, T. Doshi, and M. S. Bernstein, "Flash Organizations," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*, 2017, pp. 3523–3537.

[3] D. Retelny, M. S. Bernstein, and M. A. Valentine, "No Workflow Can Ever Be Enough: How Crowdsourcing Workflows Constrain Complex Work," *Proceedings of the ACM on Human-Computer Interaction*, vol. 1, no. 2, p. Article 89, Dec. 2017.

[4] N. R. Jennings *et al.*, "Human-agent collectives," *Communications of the ACM*, vol. 57, no. 12, pp. 80–88, Nov. 2014.

[5] O. Scekic *et al.*, "A Programming Model for Hybrid Collaborative Adaptive Systems," *IEEE Transactions on Emerging Topics in Computing*, vol. 6750, no. c, pp. 1–1, 2017.

[6] S. Faraj and Y. Xiao, "Coordination in fast-response organizations," *Management Science*, vol. 52, no. 8, pp. 1155–1169, 2006.

[7] B. A. Bechky, "Gaffers, gofers, and grips: Role-based coordination in temporary organizations," *Organization Science*, vol. 17, no. 1, pp. 3–21, 2006.

[8] D. Retelny *et al.*, "Expert crowdsourcing with flash teams," *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST '14*, pp. 75–85, 2014.

[9] A. Kulkarni, M. Can, and B. Hartmann, "Collaboratively crowdsourcing workflows with turkomatic," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work - CSCW '12*, 2012, p. 1003.

[10] L. G. Terveen, "Overview of human-computer collaboration," *Knowledge-Based Systems*, vol. 8, no. 2–3, pp. 67–81, Apr. 1995.

[11] N. Elmarzouqi, E. Garcia, and J.-C. Lapayre, "CSCW from Coordination to Collaboration," 2008, pp. 87–98.

[12] N. Karacapilidis, Ed., *Mastering Data-Intensive Collaboration and Decision Making*, vol. 5. Cham: Springer International Publishing, 2014.

[13] N. Karacapilidis and V. Tampakas, "On the Exploitation of Collaborative Argumentation Structures for Inducing Reasoning Behavior," in *Proceedings of the 18th International Conference on WWW/Internet 2019*, 2019, pp. 78–84.

[14] F. Ansari and U. Seidenberg, "A Portfolio for Optimal of Human and Cyber-Pysical Production Systems in Problem-Solving," in *13th International Conference on Cognition and Exploratory Learning in Digital Age (CELDA 2016)*, 2016, pp. 311–315.

[15] F. Ansari, M. Khobreh, U. Seidenberg, and W. Sihn, "A problem-solving ontology for human-centered cyber physical production systems," *CIRP Journal of Manufacturing Science and Technology*, vol. 22, pp. 91–106, Aug. 2018.

[16] T. W. Malone and K. Crowston, "The Interdisciplinary Study of Coordination," *ACM Computing Surveys (CSUR)*, vol. 26, no. 1, pp. 87–119, 1994.

[17] K. M. Lhaksmana, Y. Murakami, and T. Ishida, "Role-Based Modeling for Designing Agent Behavior in Self-Organizing Multi-Agent Systems," *International Journal of Software Engineering and Knowledge Engineering*, vol. 28, no. 01, pp. 79–96, Jan. 2018.

[18] N. Gilbert, D. Anzola, P. Johnson, C. Elsenbroich, T. Balke, and O. Dilaver Kalkan, "Self-organizing dynamical systems," *International Encyclopedia of the Social & Behavioral Sciences*. Elsevier, pp. 529–534, 2015.

[19] H. Simon, "Rational Decision Making in Business Organizations," *American Economic Association*, vol. 69, no. 4, pp. 493–513, 1979.

[20] K. L. Guo, "DECIDE: a decision-making model for more effective decision making by health care managers," *The Health Care Manager*, vol. 27, no. 2, pp. 118–127, 2008.

[21] L. Mann, R. Harmoni, and C. Power, "The GOFER course in decision making," in *Teaching decision making to adolescents*, J. Baron and R. V. Brown, Eds. Hillsdale: Lawrence Erlbaum Associates, 1991, pp. 61–78.

[22] C. Schneider, M. Weinmann, and J. vom Brocke, "Digital nudging," *Communications of the ACM*, vol. 61, no. 7, pp. 67–73, Jun. 2018.

[23] D. T. Luc, "Pareto Optimality," in *Pareto Optimality, Game Theory And Equilibria. Springer Optimization and Its Applications, vol 17*, Springer, New York, 2008, pp. 481–515.

[24] F. Heylighen, "Stigmergy as a universal coordination mechanism I: Definition and components," *Cognitive Systems Research*, vol. 38, pp. 4–13, Jun. 2016.

[25] D. G. Korzun, S. I. Balandin, and A. V. Gurtov, "Deployment of Smart Spaces in Internet of Things: Overview of the Design Challenges," 2013, pp. 48–59.

[26] D. Korzun, "On the Smart Spaces Approach to Semantic-Driven Design of Service-Oriented Information Systems," 2016, pp. 181–195.

[27] L. Roffia *et al.*, "A Semantic Publish-Subscribe Architecture for the Internet of Things," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1274–1296, Dec. 2016.

[28] L. Roffia, P. Azzoni, C. Aguzzi, F. Viola, F. Antoniazzi, and T. Salmon Cinotti, "Dynamic Linked Data: A SPARQL Event Processing Architecture," *Future Internet*, vol. 10, no. 4, p. 36, Apr. 2018.

[29] S. Ahmad, A. Battle, Z. Malkani, and S. Kamvar, "The jabberwocky programming environment for structured social computing," *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*, pp. 53–64, 2011.

[30] "Slack - Official Site." [Online]. Available: https://slack.com/.

[31] "Mattermost - Official Site." [Online]. Available: https://mattermost.com/.