Defining an Embodiment Space for Intelligibility

Tan Chiew Seng Sean, Kris Luyten, Karin Coninx
Hasselt University – tUL – IBBT
Expertise Centre for Digital Media,
Wetenschapspark 2, 3590
Diepenbeek, Belgium
{sean.tan, kris.luyten, karin.coninx}@uhasselt.be

ABSTRACT

One fundamental barrier to the wide-scale adoption of intelligible systems is the lack of commonly available software designs that allow system designers to readily refer to and complete the finished product with intelligible capabilities. A number of intelligible systems and toolkits that provide different kinds of explanations have been proposed. It is still unclear how the additional information needed to provide these explanations into the engineering process of such an intelligible system. In this paper, we propose a framework *Embodiment Space* that uses Activity Theory as an instrument to help model the interaction for improving end-user understanding of intelligible systems. We then connect the Embodiment Space with software architectural patterns as a tool for supporting the implementation process. We present several examples, rooted in the Embodiment Space framework, in which explanations are provided for each of the difficult circumstances experienced by the end-user.

Categories and Subject Descriptors

H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms

Human Factors, Design.

Keywords

Intelligibility, Activity Theory, Embodiment Space, Task Model, Software Architectural Pattern.

1. INTRODUCTION

As computational power grows exponentially and computer systems are becoming cost-efficient over time for practical everyday use, they play an increasing role in our lives, providing us with increasingly diverse forms of services and information access. Interactive applications, incorporating context-awareness with our environment, creates whole new forms of interaction and experience that we would never otherwise have imagined.

Grudin [10] describes the history of interaction moving from being directly focused on the physical machine to participating the user's world and the social setting in which the user is embedded. Wegner [21] brought forward the interactional approaches conceptualizing computation as dynamic interaction amongst different components in context to the user.

The ability to generate explanations or at least display the system's confidence is an important aspect for any system to be perceived as intelligible [18, 1]. A wide variety of knowledge engineering methodologies exists that focus on the explanatory knowledge. Some of these features of explanations enhance the user experience by adding either a level of self reflection about the actions of the system or the importance of such explanations into gaining the user's trust towards the system's capabilities [14].

In order to let users intervene in the computer system, especially when something goes wrong, the system first needs to present itself and the way it works to end-users with plausible logical explanations. The design of explanatory capabilities should be made an integral part of the system's design process and made commonly available for adoption by every system designers and software engineers.

Understanding users and their intents is the first and foremost critical principle in the software engineering process [20] as it essentially drives all other stages in the process. The goal of our work is to outline a design approach that starts from usage scenarios and uses the Embodiment Space as a reference framework for analysis of these scenarios. We expect to be able to identify the expectations an end-user might have towards the explanations provided by an intelligible system more precisely.

We see our work fitting in as part of the Architectural design phase for planning the necessary software blueprint. Vermeulen's design space for intelligibility and control [19] could subsequently be applied in the User-centric interface design phase. Lim's Intelligibility Toolkit [15] could be used to implement the intelligibility capabilities into the system during the Coding phase.

In this paper, we present our well-structured reference framework that forms the design of intelligible systems based on practices and experiences in both the social science and software development domains. Thus, the expected outcome is a better understanding how to expose the behavior of the system according to the user expectations.

2. MOTIVATION AND CONTEXT

By investigating the intelligible capabilities that a system can have starting from its conception, we can assure that the finished system can sufficiently generate insights for its own behavior under different circumstances. The user should be able to understand *what the system knows, how the system deduced this information and what the system is currently doing*. Therefore, an analysis of how user consciousness and activities fit with the system behavior should also be part of the software engineering process. To the authors' knowledge, this high-level approach has not been investigated in literature to its full extent.

In related work by Ju et al [11] and Cassens [4], different approaches taken for development of design framework for building intelligible systems are presented. The design framework described by Ju et al for reasoning about transitions between implicit and explicit interaction utilizes three interaction

techniques [11]. These techniques are intended for users to overcome errors in system's behavior through user reflection, system demonstration, and override. The first two can be seen as interaction techniques for improving intelligibility, while the latter is a technique for providing control. The theoretical framework described by Cassens for context-aware applications uses Activity Theory and Semiotics [4]. He had proposed to use problem frames as the main approach to capture explanations for transparency, justification, relevance, conceptualization, and learning.

In this paper, we investigate the use of Activity Theory [2, 13] as a facilitating instrument to determine how improving intelligibility and providing control to the user could help in the achievement of predefined goal in work process. Many applications are not "goal-driven" anymore rather they are context-aware and thus become more and more dependent on the situation for interaction with the user. Our approach of using Activity Theory is to determine and understand the activity to achieve the goal from the user's point of view.

We utilize the descriptive properties in Activity Theory [16] and its usage in workplace situations [9], to help us understand the unity of consciousness and activity of individual and collective work practice. Interaction encompassing both social nature and dynamics features of human practices can be addressed within an integrated framework that we define as the Embodiment Space.

We are now investigating how these different considerations – socio-technical analysis, knowledge ontology synthesis, and explanations generation – can fit into a design methodology that can be handled by knowledge and software engineering community. From the discussion of Yee et al. [22] on the metaphor of embodiment and the expectations of artifacts representation within virtual worlds, we hope the Embodiment Space can uncover the following questions:

- How a user makes use of artifacts to perform a task? E.g. navigating the neighborhood using a GPS device;
- How virtual representation aids user to complete a task? E.g. searching places of interest using Augmented Reality applications such as Layar Reality Browser[1];
- How a user draws experience from reality and applies them in virtual world? E.g. experiencing the lifestyle of alternate reality in Second Life.

These questions will be revisited and discussed through three examples in Section 3.3. The next step is to identify which aspects of an Activity Theory-based analysis can help us to capture a user's interaction within the Embodiment Space. This Embodiment Space should include knowledge about the acting subject, the objects towards which activities are directed and the community as well as knowledge about the mediating components, like rules and tools.

We present an example scenario to illustrate our approach. Alice is attending a conference in downtown San Francisco and she is unfamiliar with the restaurants in the vicinity. She uses her smart-phone to search for a posh Italian restaurant. The restaurant has to be within walking distance from her current location. Location tracking is done by a traditional GPS device, available in most mobile devices nowadays. Her social network provides the information to identify the most appropriate restaurant.

[1] URL: www.layar.com

3. THE EMBODIMENT SPACE

Computers should be able to find the common skills and abilities to interact with us, based on the way we create relationships between physical and social interaction. Dourish [8] suggested expanding on the observation that enables both human and computer systems to interact by exploiting our sense of familiarity. Intelligibility turns this around and we propose to make the system state – and what lead to this – perceivable by the human user, allowing user to understand and control the system behavior, alongside this same sense of familiarity.

3.1 Parameterization of Activity Theory

By turning interaction into something that can be modeled in a three-dimensional space, we are able to describe the interaction in broader context. These models are relatively easy to convert to designs that support the engineering process of intelligible systems. The reference framework we suggest, Embodiment Space, should lead to an informed development process that allow system designers to focus on improving the end-user understanding of the system from the conception stage onward.

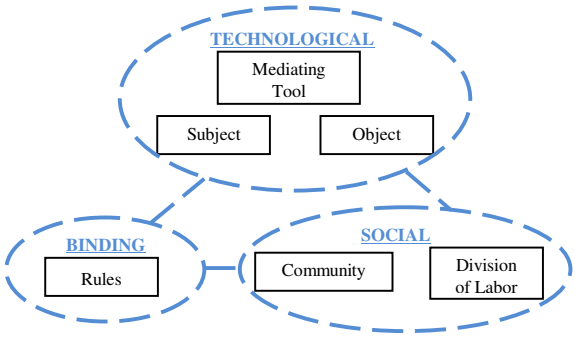

Figure 1. Basic aspects of an activity.

For example, we want the Embodiment Space to contain information about the user as acting subject, the tools and its social settings so that the user can receive real-time feedback about actions that occur. To this end, we propose a mapping from the basic structure of an activity into the taxonomy of three distinct yet interrelated aspects as depicted in Figure 1. By logical aggregation of the basic activity structure, we are able to associate the activity elements into three distinct functional groupings namely, technological, social and binding aspects.

We also identified parallels with the work on activity theory from Leont'ev [13]. The technological aspect includes physical changes in the environment whilst the social aspect includes acquisition of intangible assets from the environment. The binding aspect involves the driving forces behind development in the technological and social aspects and provides meanings through engaged interaction with objects in the world.

Using our example case, we can see that the technological aspect contains information we would associate with the acting subject itself, the mediating tool and the object, constituting towards an activity. For instance, Alice (acting subject) is looking up her smart-phone (mediating tool) to search for an Italian restaurant (object) near her present location. The social aspect contains information on the community entanglement and division of labor

which can be attributed to the proper individual and in turn better supports the accumulation of social capital. E.g. Alice searches through her network of online friends (community) and receives many responses (division of labor) from them. The binding aspect contains the information on the rules which promote accountability, trust, responsibility and predictable behavior that affect the overall technological and social fabric of the user. Such as the GPS location tracking and path-finding algorithms (rules) in Alice's smart-phone that helps her navigate to the nearest recommended Italian restaurant. The smart-phone incorporating these rules, as part of its intelligibility capability, enables Alice to understand how its computational states are linked, processed and managed over time to reach the outcome of restaurant selection.

Until now, we have only proposed a reference framework for the description of technological, social and binding aspects from the instrumentation of Activity Theory. These descriptions need to be bridged with an integral view with the implementation of the intelligible system. Embodiment Space is a foundational component that can be put in used as a (set of) software architectural pattern(s). With our reference framework, system designers are able to identify possible expectations an end-user might have towards the explanatory capabilities of an intelligible system, and we shall explain the working details in Section 4.

3.2 Notion of Embodiment Space

Embodiment Space denotes a three dimensional space of the status of entities embedded in the world with the granularity of their exposure in the world depending on what is being embedded. Embodiment Space defines the ways in which their presentation (both real and virtual) as shown in figure 2 depends on where they can be situated on the Embodiment, Environment and Event dimensions.

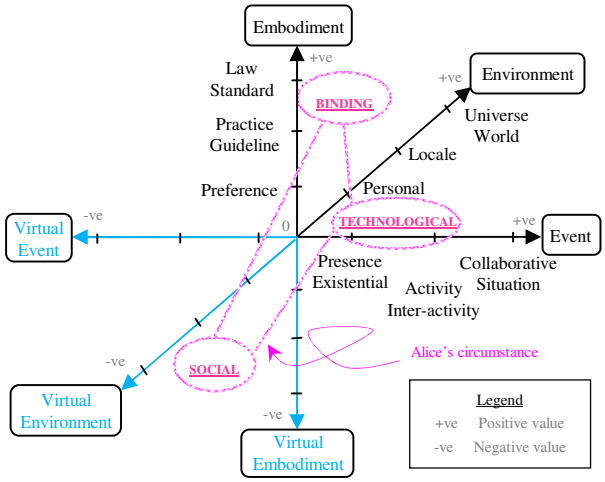

Figure 2. Three dimensional framework of Embodiment Space

Just as tangible computing, described in [8], explores this in three corresponding ways: the configurability of space, the relationship of body to task, and the physical constraints; the Embodiment Space is intended as a superset comprising both tangible computing and social computing. By providing a reference framework that situates the model that describes our engagement with the world, the Embodiment Space is contextualized with different plausible interpretations according to the activities that are taking place in a larger-scale organizational context. Likewise, the same interpretation can be part of distinctive representations which is based on the task at hand.

The technological, social and binding aspects of the user take into account the progression of user interaction, technology involvement, and the overall social atmosphere to be model in the Embodiment Space. The complexity of the interaction in each aspect is directly proportional to their representations within the Embodiment Space. This is explained using the circumstance that Alice is experiencing as depicted in figure 2. Our usage of the term "circumstance" refers to the condition that the user is experiencing in term of bearings along the Embodiment, Environment and Event dimensions.

The binding aspect for Alice is the strict compliance to the physical laws involving distance and time, in the Embodiment dimension, which are used to determine her current location and shortest path to the recommended Italian restaurant. The representation of the binding aspect contains the direct manipulation of physical laws into algorithms for location tracing and path finding computation. The social aspect can be described as somewhere between Virtual Locale and Virtual World (depending on the social circle of Alice) along the Virtual Environment dimension. The social aspect representation encompasses the communication and friend-listing services for Alice to keep in touch with her friends. Her technological aspect involves her physical presence for location update in the Event dimension. The technological aspect representation shows the use of GPS technology for location updates as well as computation methods for processing the restaurant recommendations.

3.3 Examples involving Embodiment Space

Activity Theory is capable of capturing changing contexts in breakdown circumstances [16], we will include such breakdowns in our three examples. Our examples focus on the user shifting away from the task at hand to the problem source and being involved in a different task where she will have to work with the intelligible system for an explanation for her difficulty. Therefore other factors of the activity, such as the community, division of labor, rules, subject and object will change as well. It is clear that the intelligibility should reflect these changes to the users.

In the first example, a user is using domestic activity monitoring assistant, such as Unified Room Control Interface [6] or Home Activity Recognizer [15], to customize the room settings for her reading purpose. However, she is experiencing difficulty in setting the illumination to the correct comfort for reading when she is leaning against a particular wall. The operation of using the domestic activity monitoring assistant now becomes a part of the qualitative evaluation process.

Figure 3 shows the possible parameters for searching and generating the information to explain the physical difficulty as experienced by the user. The binding aspect can be described as somewhere between the confinement of Standard and Practice/Guideline along the Embodiment dimension for her physical qualitative evaluation purpose. There is lesser consideration for the social aspect in the Environment dimension as the user is working within her personal space, and the technological aspect involves the physical activity of working with the domestic activity monitoring assistant and the room lightings control in the Event dimension.

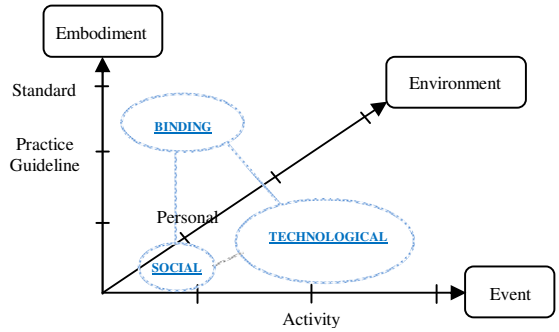

Figure 3. User unable to set the correct illumination for reading purpose

In the second example, a group of engineers working remotely through their meeting boards, such as DUMMBO [7], experience a screen update on their meeting boards. The operation of using the meeting boards now becomes a conscious action for the investigative process on who made the update and how the update fits into the project discussion. Figure 4 show the possible parameters for searching and generating the information to explain the screen update on the meeting boards. The binding aspect can be described as extension of the Preference toward the Practice and Guideline of the meeting boards in the Virtual Embodiment dimension for traceability purpose. The social aspect can be viewed as physical locale in the Environment dimension where the meeting boards are remotely situated. The technological aspect involves inter-activity among users working with the meeting boards in the Event dimension.

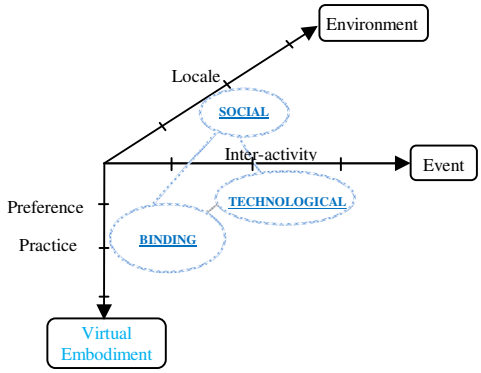

Figure 4. Meeting boards showing a screen update to remote group members

In the third example, a group of online gamers playing a massively multiplayer online role-playing game (MMORPG such as EverQuest, Star Wars Galaxies or World of Warcraft) experience a rejection error when they wanted to join a particular quest. The operation of playing the game now becomes a conscious action for the troubleshooting process. Figure 5 shows the possible parameters for searching and generating the information to explain the rejection error as experienced by the gamers. The binding aspect can be described as the software configuration settings between the Standard and the Practice along the Virtual Embodiment dimension for the rejection error troubleshooting purpose. There is high consideration for the social aspect in the Virtual Environment dimension as the gamers are living through their game character(s) or avatar(s) in the immersive virtual world with other gamers. The technological aspect involves inter-activity among multiple gamers playing towards collaborative situation in the Virtual Event dimension.

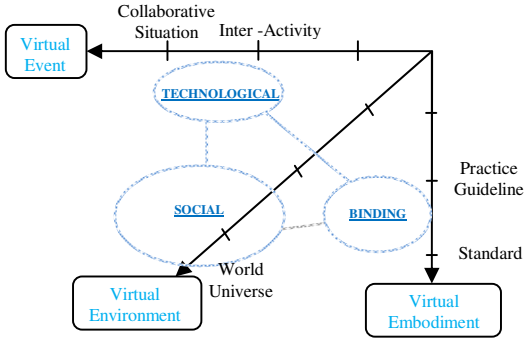

Figure 5. A massively multiplayer online role-playing game refusing a particular quest for online gamers

4. Models for Intelligibility

Design patterns have been described across many problem domains, including user interfaces, reactive systems, real-time processes, hypermedia and transport systems [3]. As such, we think they can capture a substantial amount of information that is required to create an intelligible system.

A set of intelligibility patterns represents part of the body of knowledge that can be accessed for creating suitable system interfaces. They are reusable models that include the core concepts for providing correct explanations at the right time. Task modeling, for example, is an implementation of design patterns and is used to elicit requirements in early stages of development by describing a set of tasks people perform to achieve goals.

In our study, we use the Tasks Interactor Modeling concept [17] for the explanation generation task to update user of the system state. The task models are used as patterns to inform the software architectural design – in this case – and attribute it with the capabilities to provide explanations. Mapping tasks directly to interactors, which forms the building elements of software architecture, ensures integration of information to steer the design of the behavior of a system into the software. Notice the similarity with the COMETS approach by Demeure et al. [5], in which an interactive system is considered to be a graph of interconnected models that can be queried for their semantics at runtime.

We adapted the example from Paternò [18] on his Search Architectural Software for our explanation generation purpose, and then applied to our Alice example as depicted in figure 6.

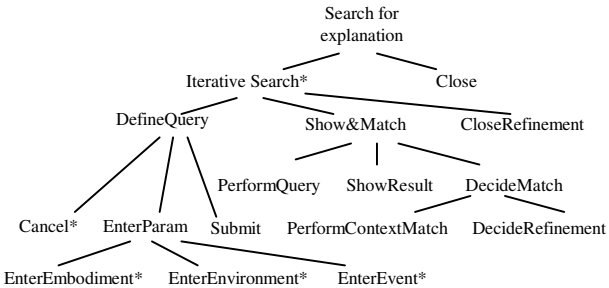

Figure 6. Explanation generation task specification

Our task for generating specific explanatory information is done through the matching of query results to the perceived circumstance of Alice (derived from the Embodiment Space

analysis). From our example, the input for Embodiment, Environment and Event are as described in Section 3.2. An explanation generation task is used to inform Alice of the system state when it is determining the nearest Italian restaurant based on the recommendations from Alice's friends. Whenever the system has no precise ideas on searching for the desired data, it automatically refines the query along each dimension of the Embodiment Space until a matching result is found.

Alice defines a first query (DefineQuery task) and only after the performance of this task, the application tasks will perform the query execution in the Embodiment Space and show the result. When Alice's friends responded with their recommended Italian restaurant in downtown San Francisco, the system starts searching for an explanation as to which recommendations is most suitable while presenting the system state and its related information. For instance, Alice asks the system why the restaurant recommended by Bob is selected. The task (DecideMatch) receives information from an application task (ShowResult) and produces input for the next interaction task (DecideRefinement). An example could be Alice trusted Bob's culinary taste and will place high priority on any restaurants that Bob recommends.

The system refines the query for each dimension in the Embodiment Space several times until the Close Refinement task is performed. At that time, it will be possible to start another completely different search without closing the session. In such a case, the restaurant that Bob recommended (satisfactory result matched in Environment dimension) is within walking distance to Alice's location (second result matched in Event dimension) and has highest priority score (third result matched in Embodiment dimension), and so the system selects Bob's recommendation and presents the restaurant details and directions to Alice.

The objects identified at the task level will be associated with interactors in Table 1 and actions among objects have been used to identify the composition among interactors.

Table 1. Association between objects and interactors

Object Name	Type	Interactor associated
InputParam-Embodiment	Perceivable	IEnter Embodiment
InputParam-Environment	Perceivable	IEnter Environment
InputParam-Event	Perceivable	IEnter Event
Init-submit	Perceivable	ISubmit
Init-cancel	Perceivable	ICancel
Database	Application	IEmbodimentSpaceDatabase
Init-close	Perceivable	IClose
Close-refining	Perceivable	ICloseRefining
ReturnQuery-data	Perceivable	IShowQueryResult

For each task of editing a data attribute we have a corresponding interactor, which receives input data from the system and sends them to the Database. The system has to wait for the control event (input_trigger) from the ISubmit interactor which provides it when it receives the input from the user and disables it. E.g. Alice starts querying her friends for restaurant recommendations. The Database sends the result data to the IShowQueryResult interactor which then presents the result to Alice. E.g. the selected restaurant details and directions.

After showing the result to Alice, the IShowQueryResult interactor re-enables the IEnter for the Embodiment, Environment and Event inputs. The ISubmit interactors then allow the system to input data again. An example could be a new recommendation is received from Charlie 10 minutes later, whom Alice had also placed same culinary priority weightage as Bob.

The IClose interactor disables all the other interactors when it receives an input from the system. The ICloseRefining interactor allows the system to start a new search, so it sends an input_trigger to all of IEnter type and IShowQueryResult interactors to cancel the data associated with the previous query. E.g. Charlie's recommendation is nearer to Alice than Bob's recommendation; in this case, the system will then prompt Alice for a selection between Bob's or Charlie's recommendation.

The software architectural pattern for explanation generation, in figure 7, has six outcomes and the procedure is as described:

1. Input_send to the disable_gate of: IEnterEmbodiment, IEnterEnvironment, IEnterEvent, ICancel
2. Input_send to the enable_gate of: ICloseRefining, IEmbodimentSpaceDatabase
3. Input_send to the enable_gate of: IShowQueryResult
4. Input_send to the enable_gate of: IEnterEmbodiment, IEnterEnvironment, IEnterEvent, ISubmit
5. Input_send to the trigger_gate of: IEnterEmbodiment, IEnterEnvironment, IEnterEvent
6. Input_send to the disable_gate of all interactors

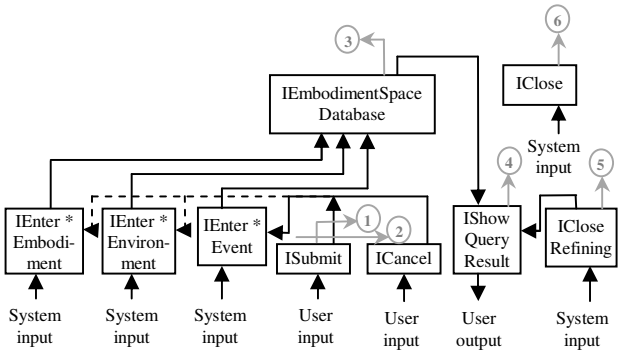

Figure 7. Software architectural pattern for explanation generation

The result of the tasks-to-interactors association in figure 8 depicts the interactors associated to each task. The relationships between tasks and interactors are straightforward for one task to be performed by one interactor, with exception for the Database interactor which actually supports the performance of two tasks.

As part of our future work, we consider the hierarchical compositions of interactors as they allow both input and output flows of information. This will make the compositions among the interactors, dynamic and reconfigurable according to specific circumstances, allowing us to study the developmental transformation which will lead to the improvement for user to understand the system.

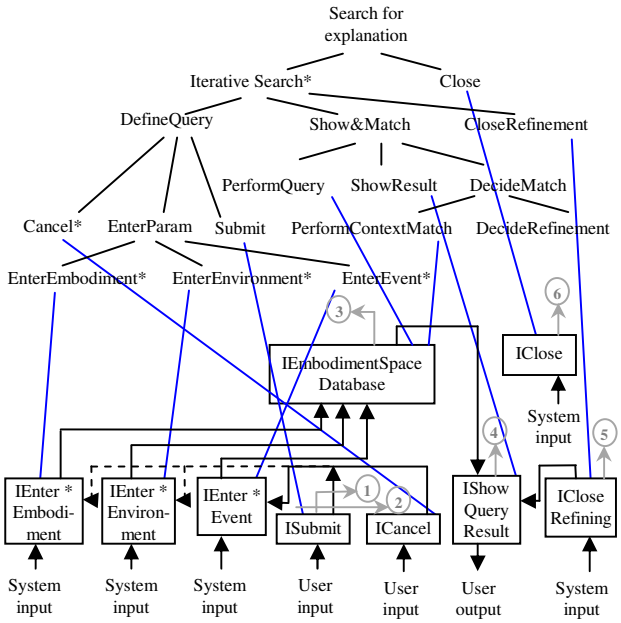

Figure 8. Tasks-to-interactors association

5. CONCLUSION

We presented a reference framework for intelligibility based on activity theory: the Embodiment Space framework. We are exploring how this framework can be used to improve the users' understanding of system behavior. The framework uses Activity Theory as an instrument to identify the technological, social and binding aspects that can be mapped to the task model. As such, the framework is a foundational component that can be employed for a (set of) software architectural pattern(s) that enables system designers to build the software blueprint for intelligible systems.

Our next step is to formalize the relationship between the technological, social and binding aspects with the Embodiment Space to ensure scalability of intelligible system design for large and complex applications.

6. REFERENCES

[1] Antifakos, S., Kern, N., Schiele, B. and Schwaninger, A. 2005. Towards improving trust in context-aware systems by displaying system confidence. In Proc. MobileHCI '05, pp 9–14. 2005.

[2] Bødker, S. 1991. Activity theory as a challenge to systems design. In Information Systems Research: Contemporary Approaches and Emergent Traditions. pp 551–564, 1991.

[3] Bosch, J. 2000. Design & Use of Software Architectures - Adopting and Evolving a Product Line Approach, Addison-Wesley, 2000.

[4] Cassens, J. 2008. Explanation Awareness and Ambient Intelligence as Social Technologies. Doctoral Thesis for the degree doctor scientiarum, Norwegian University of Science and Technology, Trondheim, Norway, 2008.

[5] Demeure, A., Masson, D. and Calvary, G. 2011. Graphs of models for exploring design spaces in the engineering of Human Computer Interaction. In Proc 2nd SEMAIS workshop of the IUI 2010 conference. 2011

[6] Dey, A. K., Salber, D. and Abowd, G. D. 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. In Human-Computer Interaction Journal, pp. 97-166, 2001.

[7] Dey, A. K. and Newberger, A. 2009. Support for context-aware intelligibility and control, Proc. 27th Int'l conference on Human factors in computing systems, pp 859-868, 2009.

[8] Dourish, P. Where the Action Is: Foundations of Embodied Interaction. MIT Press, 2001.

[9] Fjeld, M., Lauche, K., Bichsel, M., Voorhoorst, F., Krueger, H. and Rauterberg, M. 2002. Physical and Virtual Tools: Activity Theory Applied to the Design of Groupware. CSCW 11, pp 153–180, 2002.

[10] Grudin, J. 1990. The computer reaches out: The historical continuity of interface design. Proc. CHI '90, pp 261-268, 1990.

[11] Ju, W., Lee, B., and Klemmer, S. 2008. Range: Exploring implicit interaction through electronic whiteboard design. In Proceedings of the ACM 2008 Conference on Computer Supported Cooperative Work, pp. 17-26, 2008.

[12] Kuutti, K. 1995. Activity theory as a potential framework for human-computer interaction research. Context and consciousness: Activity theory and human computer interaction, pp 17-44, MIT Press, 1995.

[13] Leont'ev, A. N. 1978. Activity, Consciousness, and Personality. Prentice-Hall, 1978.

[14] Lim, B. Y., Dey, A. K. and Avrahami, D. 2009. Why and why not explanations improve the intelligibility of context-aware intelligent systems, In Proc. 27th Int'l conference on Human factors in computing systems, pp 2119-2128, 2009.

[15] Lim, B. Y. and Dey, A. K. 2010. Toolkit to support intelligibility in context-aware applications. In Proc Ubicomp'10, pp 13-22, 2010.

[16] Nardi, B. A. 1995. Context and Consciousness: Activity Theory and Human-Computer Interactions. ACM Interactions, 1995.

[17] Paternò, F., 1994. A Theory of User-Interaction Objects, Journal of Visual Languages and Computing, 5, 3., pp 227-249, Academic Press, 1994.

[18] Roth-Berghofer, T. R. and Cassens, J. 2005. Mapping Goals and Kinds of Explanations to the Knowledge Containers of Case-Based Reasoning Systems. Case Based Reasoning Research and Development 2005, pp 451-464, 2005.

[19] Vermeulen, J. 2010. Improving Intelligibility and Control in Ubicomp. *In Ubicomp '10 Doctoral Colloquium*. pp 485-488, 2010.

[20] Warfel, T. Z. 2009. Prototyping: A Practitioner's Guide, Rosenfeld Media, 2009.

[21] Wegner, P. 1997. Why interaction is more powerful than algorithms. In *Communications of the ACM*, 1997.

[22] Yee, N., Ellis, J., and Ducheneaut, N. 2009. The Tyranny of Embodiment. *Artifact*, 2, pp 1-6, 2009.