**Journal of Computer and Communications, 2015, 3, 212-219**
Published Online May 2015 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2015.35027

# Approach to Model-Based Development of Context-Aware Application

**Imen Jaouadi[1], Raoudha Ben Djemaa[2], Hanene Ben Abdallah[1]**

[1]MIRACL, FSEG, Sfax, Tunisia
[2]MIRACL, ISIMS, Sfax, Tunisia
Email: jaouadiiimen@gmail.com

Received April 2015

## Abstract

**In order to address challenges posed by advances in mobile computing, ubiquitous devices software engineering, wireless and sensor technologies in many applications running, this paper provides an approach that simplifies the design and the implementation of context-aware Interactive systems called CAISDA (Context-Aware Interactive System Development Approach). This approach is based on the analysis and synthesis of context-aware frameworks proposed in literature.**

## Keywords

**Context-Aware, Model Based Development, Runtime, Adaptation**

## 1. Introduction

Nowadays, the current contexts of use of interactive systems vary according to the user profiles, platforms, devices, and environment. The differences posed by these contexts require applications with interfaces and functionalities are able to adapt to the change of these contexts. Since it is neither feasible nor scalable to implement only one version for each change in context, developers are moving towards the use of models to facilitate the development phases of context-aware applications.

Context information is captured using sensors. Thus, applications are required to interact with sensors [5]. Moreover, Context is gathered from heterogeneous sources which use different representation. Thus, the context information may be requiring additional interpretation to be significant for applications [5]. In addition, the context is dynamic and applications must change their behavior following the change of each context element. So, in between perceiving context and adapting to it, there is a whole process which must be taken into account in the development phases. Subsequently, building context-aware system that can automatically adapt to the context of use is a challenging task that requires additional efforts and resources. Hence, developers of applications will permit to have context-aware applications that, on the one hand, capture, intercept, and represent context and, on the other hand, adapt themselves to the change of the context in runtime. Some of these requirements have been addressed by researchers in [1] [3] [7] [8]. Nevertheless, there exists not a single approach that addressed a complete solution to context-aware application development that included theses requirements. This is the motivation behind this work. In this paper we propose an approach called CAISDA for context-aware system interactive which incorporates all the requirements and bridges the gap between design of CAA and its implementation. Via a full software development lifecycle and a software framework, we aim to support all development phases of context-aware application.

This paper is organized as follows: Section 2 presents related research work on context-aware framework. In section 3, we discuss theses framework and we propose our motivation. Then we describe the architecture of CAISDA in section 4. Section 5 presents the methodology to be follow in the future work to realize our work. Finally, section 6 offers our conclusions and perspectives.

## 2. Related Work

A lot of work has been done in the area of context-aware computing in the past few years. This section presents a selection of works that focus on building context-aware applications based on models.

Seminal work has been done by Dey *et al.* [6] which defining an architecture for building context-aware applications. They develop a context toolkit that enabled rapid prototyping of context-aware applications. The architecture of ContextToolkit is composed of sensors to collect context information. The widgets encapsulate the contextual information and provide methods to access to such information. The Interpreters transform the context information into high-level formats that are easier to handle. The Aggregators are used to group the context information related to a subject or situation. This layered architecture is simple to implement and allow the separation of acquisition process and context representation of the adaptation. However, it is not based on a reliable and robust context model for organizing the wide range of possible contexts in a structured format, or for writing rules about contexts.

SOCAM (A service oriented context-aware middleware) [7] is a service oriented context-aware middleware which supports the development of services context-aware applications. The architecture of SOCAM is based on the following components: The Context providers which abstract useful contexts from heterogeneous sources and convert them to OWL representations. The Context interpreter provides logic reasoning services to process context information. The Service-locating service provides a mechanism where context providers and the context interpreter can advertise their presence. The Context-aware services make use of different level of contexts and adapt the way they behave according to the current context. The Context data base stores context ontology and past contexts for a sub-domain. This architecture has a robust system of reasoning about the context based on ontology for context description. It allows the recording of rules containing context-based events and conditions, and notify the system when the context changes. This middleware is easy to manage but it is specific to an application domain. Thus, it is not based on a generic context model and introduces concepts that do not make sense for some areas. In addition, this middleware provides a support for most tasks involved in dealing with acquiring context from various sources; interpreting and carrying out dissemination of context. But it does not offer detail about the implementation of each task.

WildCAT [4] is an extensible Java framework which provides mechanism for realizing context-aware applications. It provides a dynamic data model to represent the execution context which considers context to be made of several domains, which separates different aspects of the execution context. Each context domains is modeled as an XML tree of named resources. Wildcat provides an interface programming to discover, interpret and monitor the events occurring in an execution context and records every single change occurring on the context model. To support the context modeling, Wildcat provides a dynamic context model, but does not implement any of the mentioned context domains; it provides interfaces to realize them. Thus, Wildcat presents a programming interface, but they have not proposed mechanisms for the deployment and the configuration to help programmers to use this framework. Furthermore, it follows the notion of active and passive sensors. However, it does not provide means to gather the data but rather provides methods to monitor the sensors.

In the same context, Badram [1] proposed a framework called JCAF (JAVA context context-awareness framework) based on Java language. JCAF is a service oriented, event based infrastructure suitable for the development of context-aware applications. It is composed of the following component. The *Context Clients* are the context-aware applications using the JCAF infrastructure through subscribing or requesting context information or by subscribing to an entity listener. The Context service is responsible for handling specific context. Access to a Context Service is controlled through the *Access Control* component, which ensures correct authentication of client requests. A monitor is responsible for communication with sensors for acquiring context information. A context actuator is designed to work together with one or more actuators to change the context.

The core modeling abstraction in JCAF is the interfaces and the classes: Entity, Context, Relationship, and ContextItem. This framework is extensible and supports adaptation in runtime based on event. However, context representation proposed by this framework does not provide the ability to define other abstractions over simple context information. Moreover, wildcat aid technology expert but certainly do not assist novice user; they have not offered mechanisms for the deployment or the configuration to help programmers to use JCAF to develop their applications. Furthermore, it supports various sensors for monitoring locations and base classes for describing relevant entities used in context-aware applications can be implemented. However, it is very hard to extend it with new computational resources to cope with an increased number of sensors.

A software framework that addresses the challenges of building context-aware systems is presented in [8] which specify six layers in the context-aware system development: the context gathering layer acquires context information from sensors and processes it through interpretation and aggregation. The context reception layer translates inputs from the context gathering into the Context Modeling Language CML representation. The *context management layer* maintains a set of CML context models and their instantiations. The query layer provides an interface to query the context management layer. The adaptation layer manages repositories of situation, preference and triggers definitions, and evaluates these on behalf of applications' needs. The *application layer* supports applications through their programming toolkit model. To support of this architecture and the management of models, Henricksen *et al.* [8] propose a software methodology composed for five main activities: Analysis and specification of the context fact types; Design of the triggering mechanisms for the application; Implementation of the application; Customization of the abstract models and Testing. This approach offers domain independent support on developing context aware system that combine an extensible architecture, context metamodel and process design. Nevertheless, the proposed methodology only presents the high level activities to be performed and does not specify details related to each activity. Moreover, the proposed notation to context model does not have a tool to support the context modeling which makes it difficult to use in practice. Also, their context model does not consider information about the context-aware system behavior.

Costa [3] proposes an architecture based on the Event-Control-action, consists of three components, namely the *context processor*, the *controller* and the *action performer. The context processor* component gathers context information from the user's environment performs context reasoning and generates context and situation events. The *controller Component* observes events from context processors, monitors conditions rules, and triggers actions on action performer when the condition is satisfied. This approach provides a mechanism that facilitates the dynamic configuration and execution of particular application behaviors which based on a rule engine that gathers context and situation values from context processing components. To support this architecture, Costa proposes a context model based on three foundational concepts: *Entity*, *Context*, and *Context Situation*. Drawbacks of this approach is that does not consider reusing existing application models, compelling context models to be developed from scratch as a new separate model. Also, the context metamodel is not rich, it does not provide concepts that are as important to describe the dynamic aspect of context such as focus, temporal constraints, and the associations types that are defined by the work reviewed in [10]. The major drawback of this approach that is proposed only a design methodology for structuring context-aware applications, which is based on the service-oriented architectural style; no concrete implementation was suggested.

Vieira *et al.* [12] propose a framework named CEManTIKA to support context modeling and context-aware system design, in a generic domain independent way. This framework composed of context Source, context Manager and context consumers. Each *context Source* provides a specific context element. The Context Manager is responsible for controlling the activities related to context acquisition, processing, dissemination and storage. In addition, it manages the context sources used by the Context-aware system. The Context Consumer change their behavior according to the condition related to the context. This approach is based on a generic context metamodel which models the structural and behavioral aspects of context. But, it proposes only guidelines to support the design of context-aware system, they do not propose any implementation or code for development.

In [11] the authors propose a conceptual framework named TriPlet to support the adaptation of user interface of an interactive system. It is structured in three core components. A CAMM (context aware metamodel) described concepts required to implement and run a context-aware application. Context-aware Reference Framework (CARF) is a reference framework created to list the most relevant concepts for implementing and executing CAA. It can be used before the implementation phase of an application, as an extensive catalogue to guide developers in taking design decisions, or after the implementation phase of an application, to analyze and to evaluate the concepts that were considered, aiding also to identify underexplored areas for future extensions. Context-aware Design Space (CADS) is responsible for analyzing, comparing and evaluating coverage levels of CAA of user interfaces with well defined criteria. It supports stakeholders in the phases of implementation, analysis, and evaluation of adaptive and adaptable applications. The major drawback of this approach is that it offers only methodology for building applications. They have offered neither implementation nor code nor development or configuration mechanisms to help programmers to develop applications.

## 3. Synthesis and Motivation

In order to identify relevant Works in literature and specify and characterize each approach, we must choose the criteria allowing a comparison of these approaches. Some of these criteria are inspired from works reviewed in the previous section. Others have been identified based on the requirements of our problematic.

*Context Model.* The data model used to represent the context should be :

- *Domain Independent:* it should be more generic as possible to make it used by different applications and in different areas.
- *Rich*. It must be rich enough to represent all relevant aspects of the context.
- *Dynamics*. It ust allow to represent dynamic aspect of the context.
- *Behavioral and structural* It should allow to modeling the structural and behavioral aspect of the context.

• *Architecture*. To support the building of context-aware applications, approach should support the:

- *Context collecting*. It gathers information context from different data sources attached to the application,

- *Context interpreting*. It transforms context information collected from data sources into significant format easier to handle.

- *Behavior specification*. It specifies different application behavior to the context information.

- *Context storage*. Applications can make use of not just the current context, but also the past contexts to adapt their behavior for better interacting with users.

- *Context Controlling.* It observes the dynamic elements of the context and detects changes that occur in the context,

- *Adapting of application*. It adapts application to the constantly change in the context.

- *Separation context management and adaptation*. An approach must separate between the context management and the adaptation mechanisms for reusability and extensibility of approach.

• *Detailed* Development *process*. An approach must propose a methodology to guide the development of applications using their tools (context metamodel, framework, etc)

• *Rapid development and deployment of applications.* An approach should offer to context-aware application mechanisms for the deployment and the configuration of applications which are easy to use. These mechanisms should not require much programming and configuration efforts; the developers should not be forced to become experts in the field. Subsequently, the development and the deployment time should be reduced with respect to the time required to develop and deploy an application without the approach of context- aware development.

• *Support for implementation*. An approach must exist concretely as a usable implementation.

**Table 1** summarizes each of the previously mentioned context-aware frameworks, while taking into consideration these requirements.

Based on the analysis provided in **Table 1**, we can conclude that no single approach has the features to address all the issues that were identified.

Most of proposed work based on not rich context models [1] [4] [6] [7] David *et al.*, 2005) or they don't offer solutions to integrate the behavioral aspects of context modeling [8]. Even if there are approaches supported the dynamic and the rich context modeling, and taking into account its structural and behavioral aspects, they propose an architecture which not supports all the components that are essentials for context-aware building or they have only proposed methodologies for designing context-aware applications [3] [11] [12].

On the other hand, **Table 1** shows that is only CEManTIKA approach [12] among these frameworks supporting all the necessary features for supporting context-aware application (collecting, interpreting, storage, controlling, triggering and adapting). However, this approach proposes only guidelines to support the design of context-aware system, they do not propose any implementation or code for development. Hence,

**Table 1** shows that there are not many solutions that address to implementation of our approach [1] [4]. In addition, there is lack of general techniques and tools to simplify the design and the development of Con text-

**Table 1.** Comparison between current context-aware frameworks.

| **Requirements** | | Context Tolkit [6] | SOCAM [7] | Wildcat [4] | JCAF[1] | Henricksen *et al.* [8] | Costa [3] | CEManTIKA [12] | TriPlet [11] |
|---|---|---|---|---|---|---|---|---|---|
| Context model | *D omain independent* | + | - | + | + | + | + | + | + |
| | *Rich* | - | - | - | - | + | - | + | + |
| | *Dynamics* | - | - | + | - | + | + | + | + |
| | *Behaviora and structural* | - | - | - | - | - | + | + | + |
| Architecture | *Context collecting* | + | + | + | + | + | + | + | - |
| | *Context interpreting* | + | + | - | - | + | - | + | - |
| | *Behavior specification* | + | + | - | - | - | - | + | - |
| | *Context storage* | + | + | - | - | - | - | + | - |
| | *Context controlling* | - | + | + | + | - | + | + | - |
| | *Application adaptation* | - | - | + | + | + | + | + | - |
| | *Separation context To adaptation* | + | + | + | - | + | + | + | + |
| Detailed Development process | | - | - | + | - | - | + | + | - |
| Support for implementation | | - | - | + | + | - | - | - | - |
| Rapid development of applications | | - | - | - | - | - | + | - | - |

aware application. We can observe also that only few solutions have been reported on the development of context-aware application that combines an integrated support on context modeling and context-aware system development. Finally, we can conclude that is no one approach who offers support on developing context aware application that combine an extensible complete architecture, a context metamodel and a process design and a support for implementation.

The concerns and shortcomings observed for CAA frameworks delineate the problem space of our work. We propose an approach that is, firstly, reposed on generic, rich a dynamic context model. Secondly, it will simplify the process of development and enables the rapid creation of application at design time. Thirdly, our approach will allow capture, interpret and storage of the context in runtime. As well as our approach will permit the observation of the context and the adaptation of the application to the change of the execution context. Finally, we will implement and test the feasibility of our approach. CAISDA approach presented in the next section intends to fulfill these requirements.

## 4. Overview of CAISDA Approach

The main goal in this paper is to satisfy the requirements posed in the previous section. Our approach CAISDA responds to these requirements. Its architecture is formed of two parties. It describes the development process of a context-aware application by the developer in design time for it is consistent with our framework in run time. In the other hand, it describes the architecture of this framework in run time.

### 4.1. In Design Time

**Figure 1** shows an overview of CAISDA approach in design time.

In design time, the developer must build a context model of its application to be consistent with metamodel proposed for CAISDA [10] and follow a set of steps guided by mechanisms and tools to generate at the end a context-aware application integrated in the CAISDA framework. Firstly, the programmer should build its appli-

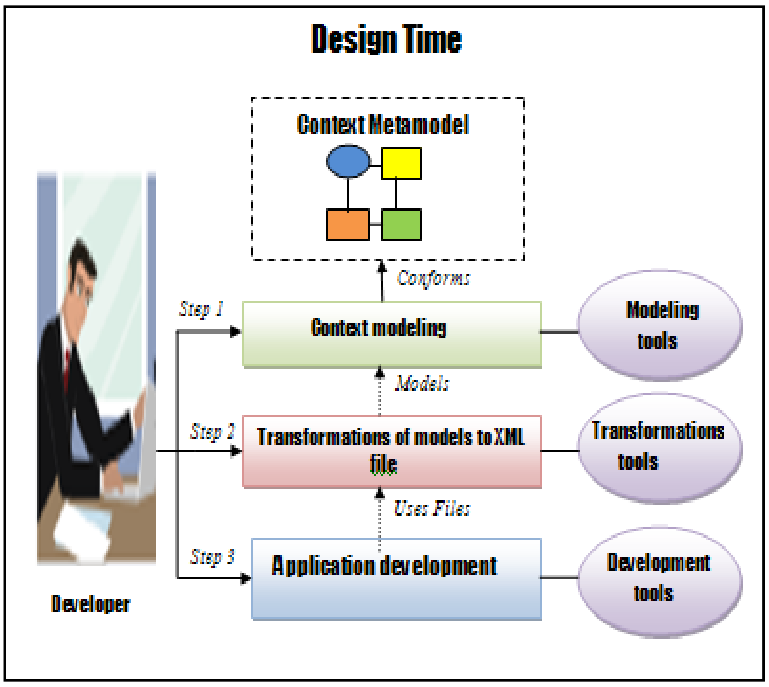

**Figure 1.** Overview of CAISDA in design time.

cation context model using modeling tools that assist the developer in the specification of models. These tools are very important to simplify the use of his approach and to support the validity of context model and its conformity to context metamodel. Secondly, this context model will be transformed to data simply manipulated by the application. For example in our approach we will choose the XML file. This step will be guided by the transformation tools used in the model driven engineering (MDE) [2]. Finally, the programmers should develop their applications and integrate it to CAISDA framework. In the next section we will describe the architecture of this framework.

### 4.2. In Run Time

The architecture of CAISDA in run time is described in the **Figure 2**.

In runtime, when the user launches the application, he must be able to capture and to interpret the context elements possible to change during the application execution. For this raison, we propose to use software modules called Context Providers which collect the context from different heterogeneous sources and interpret it to understandable structure by the system. In addition, our framework must permit the system to observe the dynamic elements of the context and detect the change in the context. To do this, we use in our solution software components called Context controllers, which monitor the context information sent by the context provider. For detecting the change of the context and notify the system for this change, our approach will be based on the paradigm ECA (on Event if Condition do Action): when the event occurs, the condition is evaluated and, if verified, the action is triggered. At the moment, the Context Analyzer identifies the change of the context and provides the necessary reactions to respond to these changes to Adaptation Producer. This software component allows, on the one hand, to identify the type of adaptation required (functional or interface) according to the reactions defined by the Context Analyzer and, on the other hand, it realize this adaptation based on adaptation technical. The Adaptation aims to change one or more aspects of an interactive system according to the context in which end users are located.

## 5. Methodology of Our Work

In this section, we present a methodology to follow in our work to implement CAISDA approach. Our goals are organized in three phases.

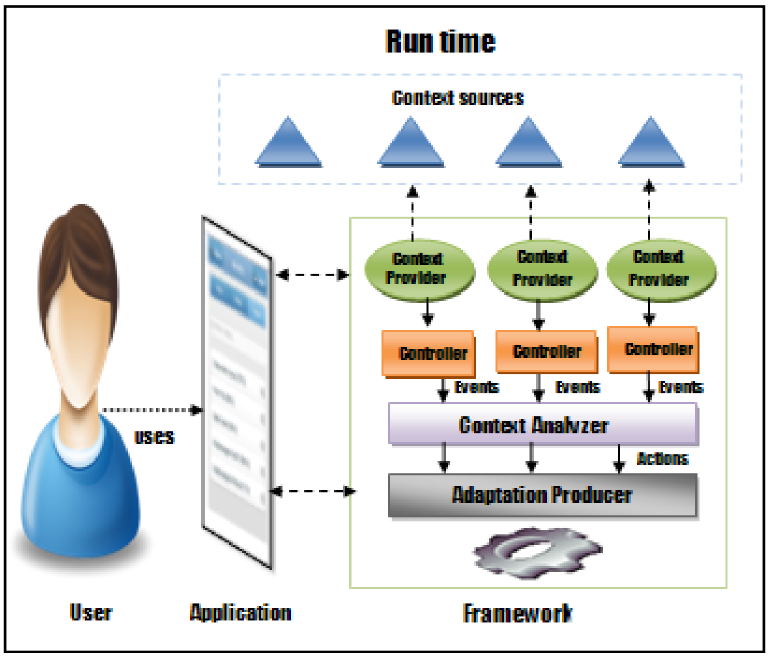

**Figure 2.** Overview of CAISDA in run time.

- First phase aims to build a generic and rich context metamodel that representing the static and dynamic aspect of the context. The works reviewed in the literature has allowed us to identify the concept and the main shortcomings in the context modeling. For more information the reader can refer to our work [10].
- The second phase intends to develop a software framework that observing the execution context of an application in runtime, reporting the system events that occur there and the components of this framework based on the analysis of the results of the work reviewed in the Section 2. Whereas the techniques identified of adaptation approaches reviewed in [9] used to define the adaptation process of our framework.
- The third phase consists in defining a development process to help and guide the programmer to develop their application by providing mechanisms for configuration and deployment.

## 6. Conclusion and Futur Work

A context-aware application can dynamically capture a set of information from its environment, this information represents a context, and the application adapts its execution to this context. Thereby, the context-aware design poses new challenges. The system can be implemented in different ways. In order to improve the scalability and reusability of these systems, several approaches proposed architectures and frameworks in their works.

In this paper we reviewed some frameworks for context- aware development. Generally studied approaches are considered only a few context elements. However, it is important that an approach reposes on rich context model which support the dynamic and static aspect of the context to respond to changes of the context in run time. In addition, most approaches interest in offering methodologies for the development of context-aware application more than complete implementation support. Therefore, we presented in this paper a CAISDA, approach for context-aware development. Our approach will be original in the sense that it combines an extensible architecture that meets all the requirements of a context-aware application, a rich and generic context model, and a rapid and complete development process.

In our future work we will implement our approach. Firstly, we develop our development process to guide the programmer to use our approach providing mechanisms for design and configuration. Secondly, we will develop our software framework. Finally, we will show the validity of our approach by applying it to cases study.

## References

[1] Bardram, J.E. (2005) The Java Context Awareness Framework (JCAF)—A Service Infrastructure and Programming

Framework for Context-Aware Applications. *Proceedings of the Third International Conference on Pervasive Computing*, Munich, May 2005, 98-115.

[2] Bezivin, J. (2001) Towards a Precise Definition of the OMG/MDA Framework. *Proceedings of the* 16*th Annual International Conference on Automated Software Engineering ASE*, San Diego, 26-29 November 2001, 273-280. http://dx.doi.org/10.1109/ASE.2001.989813

[3] Costa, S. (2007) Architectural Support for Context-Aware Applications—From Context Models to Services Platforms. Ph.D. Thesis, The Netherlands University, Enschede.

[4] David, P.C. and Ledoux, T. (2005) WildCAT: A Generic Framework for Context-Aware Applications. *Proceedings of the* 3*rd International Workshop on Middleware for Pervasive and Ad-Hoc Computing*, ACM, New York, 1-7. http://dx.doi.org/10.1145/1101480.1101483

[5] Dey, A.K., Salber, D., Futakawa, M. and Abowd, G.D. (1999) An Architecture to Support Context-Aware Applications. *The* 12*th Annual ACM Symposium on User Interface Software and Technology* (*UIST*'99), GVU Technical Report GIT-GVU-99-23, June 1999.

[6] Dey, A.K., Abowd, G.D. and Salber, D. (2001) A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human Computer Interaction*, **16**, 97-166. http://dx.doi.org/10.1207/S15327051HCI16234_02

[7] Gu, T., Pung, H.K. and Zhang, D.Q. (2005) A Service-Oriented Middleware for Building Context-Aware Services. *Network and Computer Applications*, **28**, 1-18. http://dx.doi.org/10.1016/j.jnca.2004.06.002

[8] Henricksen, K. and Indulska, J. (2006) Developing Context-Aware Pervasive Computing Applications: Models and Approach. *Pervasive and Mobile Computing*, **2**, 37-64. http://dx.doi.org/10.1016/j.pmcj.2005.07.003

[9] Jaouadi, I., Ben Djemaa, R. and Ben-Abdallah, H. (2014) Interactive Systems Adaptation Approaches: A Survey. *Proceedings of the* 7*th International Conference on Advances in Computer-Human Interactions ACHI*, Spain, 127-131.

[10] Jaouadi, I., Ben Djemaa, R. and BenAbdallah, H. (2014) A Generic Metamodel for Context-Aware Applications. *Proceedings of the* 23*rd International Conference on Systems Engineering ICSEng*, Las Vegas, 19-21 August 2014, 587-594.

[11] Motti, V.G. and Vanderdonckt, J.A. (2013) Computational Framework for Context-Aware Adaptation of User Interfaces. *Proceedings of the Seventh International Conference on research Challenges in Information Science* (*RCIS*), France, 29-31 May 2013, 1-12.

[12] Vieira, V., Tedesco, P. and Salgado, A.C. (2011) Designing Context-Sensitive Systems: An Integrated Approach. *Expert Systems with Applications Intelligent Collaboration Design*, **38**, 1119-1138.