Extending a Middleware for Pervasive Computing to Programmable Task Management in an Environment of Personalized Clinical Activities

Giuliano Ferreira, Iara Augustin,
Giovani Rubert Librelotto,
Fábio L. da Silva, Alencar Machado
Federal University of Santa Maria
Technology Center
Avenida Roraima 1000, 97105-900
Santa Maria, RS, Brazil
Email: {august,librelotto}@inf.ufsm.br,
giuliano@cpd.ufsm.br, alencar.ufsm@gmail.com

Adenauer Correa Yamin
Catholic University of Pelotas
Rua Felix da Cunha, 412, 96010-100
Pelotas,RS, Brazil
Email: adenauer@ucpel.tche.br

***Abstract*—Currently, Pervasive Computing has focused on the development of programmable and interactive environments, which are intended to help the user in daily activities. The health system of the future envisages the use of Pervasive Computing as a way of optimizing and automating clinical activities. Under such perspective, the present study has tried to adapt a middleware for pervasive environment management to support and manage the accomplishment of clinical tasks (pervasive applications that help physicians perform their activities), fulfilling some requirements of activities-oriented computing, and creating a tool that will help physicians in their daily tasks. So, ClinicSpace can be seen as a system context aware pervasive oriented clinical tasks.**

***Keywords*-Ubiquitous Computing; middleware; daily activities oriented computing; end-user programming; clinical activities.**

I. INTRODUCTION

The first systems of pervasive computing have concentrated on the creation of middleware for management and availability of the ubiquitous environment, aiming at understanding its requirements and need [6][3][14]. Systems available at that time made possible to create some concepts inherent to the nature of the pervasive space: communication, mobility, context-awareness and daily activities. Communication and mobility have been approached for a long time; context is currently being handled; but daily activities require further efforts, because they are in an early phase, where what is desirable has still to be defined [10].

The current focus of Pervasive Computing is turned towards processing daily human activities in the most integrated way as possible to the real environment known by the final user (user-centric computing). Under such perspective, one of the major application areas is Health Care, it already faces situations where information from the physical world is proactively acquired and automatically integrated to applications (virtual world) [15].

Then, we can consider that Pervasive Health (or Ubi-Health) is in its first generation, trying to understand the needs, features and technologies required to design systems that will create the hospital of the future [9]. The project Hospitals of the Future envisages the use of technologies that will make an intelligent space, reactive and proactive, where information management systems will take decisions and will adapt to the situations they detect [11].

However, one argues that the system proactivity should not be too strict, once it is designed in a generalized way and not customized. If the physician wants to make things his way, he must be able to do so and interact, command and interfere in the tasks/activities managed by the pervasive system. Therefore, the pervasive system is required to focus on the end-user (physicians) and provide support to his daily tasks, balancing between proactivity (act in the place of the user) and customization (individual way of performing a task).

Based on such premises, the proposal of the project "ClinicSpace: Support to clinical tasks in the hospital environment of the future based on Ubiquitous/Pervasive Computing technologies" is to develop a pilot-tool that makes it possible to physicians to customize his tasks, which are managed and performed by a middleware in a pervasive environment. The main goal in the customization of tasks is to reduce the impact of interference of the automated system in a clinical environment and, therefore, minimize the high rejection to computational systems that such interference may cause. So, the middleware must provide mechanisms for users to run, stop, resume and schedule their tasks. Moreover, it should also control the trigger of tasks in response to context changes.

This paper is organized in the following sections: Section 2 discusses how clinical activities/tasks were modeled; Section 3 discusses the architecture for programming and running tasks, briefly discussing target middleware for

 ISBN: 978-1-61208-100-7

pervasive environments upon which the current study was based, as well as changes made to adapt the middleware to the activities-oriented computing and clinical environment requirements; Section 4 introduces a use case about the tasks management system; Section 5 discusses the prototyping and evaluation of developed architecture; Section 6 describes some related works and eventually Section 7 makes final considerations about the study.

II. Task-Oriented Computing

Clinical activities, such as outpatient care, are processes that take place collaboratively, in a coordinated and distributed way, in a given space and can be aided by computational applications [1]. According to the Theory of Human Activity [13], human activities can be modeled with a subject (who practices an activity), an objective (what to do), an action (process to accomplish the objective) and operations (how the action is performed). From that theory we derive the concepts adopted to model tasks for the construction of the ClinicSpaces software architecture.

Thus, clinical activities were modeled in tasks (actions). For example, the activity "outpatient care" can be modeled as a mixed task that identifies the patient automatically, looks for his electronic medical records, reviews pending tests, etc., displays them to the physician, and makes it possible that the physician adds information during the visit. Thus, a task that will proactively help the physician during patient's visits is created. This task can also be linked to other tasks, so that the activity can be modeled in the most complete way as possible. For example, it could be connected to a task of tests request and/or treatment prescription [12].

Tasks can be customized according to how the user (physician) can perform his activities. The goal is to create abstractions so that the computational system can fit the real features of the clinical environment. Thus, in the same way as actions are composed of operations, tasks are composed of sub-tasks. Sub-tasks are pervasive applications that maintain a direct relation with computational artifacts (applications, services and resources) and make available the computational support to customized tasks, which together model the clinical activity to be performed. Sub-tasks are concerned to the applications of the pervasive health care electronic system (pEHS) [8], such as finding the electronic records, viewing tests and management systems of the pervasive environment, extended to tasks handling (Section 3) Both tasks and subtasks were modeled through specific ontologies [5], containing information regarding identification (user that created the task, clinical expertise, description of functionalities) and management (status, required resources, contexts supported).

So that, through the Tasks Edition Interface (a programming tool oriented to the end-user, see Figure 1), a task is programmed intuitively by the physician through clustering of sub-tasks and tasks, in a sequence that reflects the way how the user performs his activity (customization). At the same time, customization can also be made at the sub-task level, with the configuration of specific parameters, which change the way how the sub-task processes its functionality.

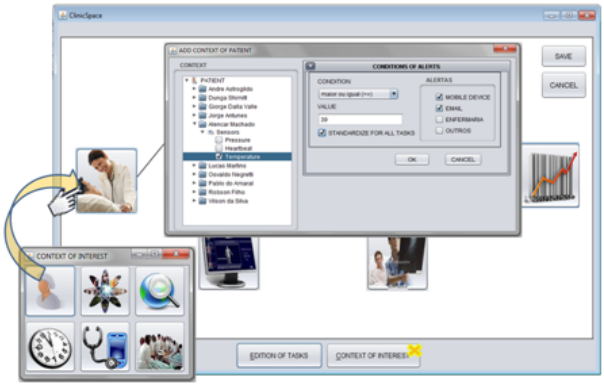

Figure 1. Task Edition Interface

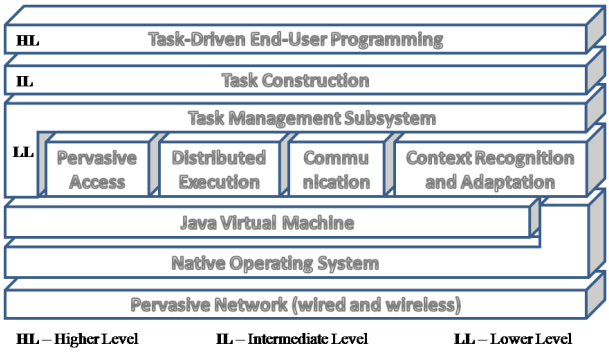

Figure 2. System architecture for tasks managing and programming

III. Task Management

As we see, the ubiquitous systems bring new challenges to different areas. Among them is the development of a new middleware that, besides managing the pervasive environment, must manage users' daily tasks. Middleware must allow users to customize (program), run, stop, resume, schedule their tasks and control how they respond to context changes. Our challenge in this work, within the scope of the ClinicSpaces project, was to adapt a pervasive environment management middleware to manage the users' tasks as well (physicians), meeting the requirements of user-centric and daily activities support in the clinical environment [9].

The architecture designed for programming and management of tasks (see Figure 2), is organized in levels that reflect the system views:

- Higher level, composed of end-user (physician) who interacts with the tool to build and edit his tasks that would be triggered by changes in context (reactive). The ClinicSpaces interface is based on principles of the

 ISBN: 978-1-61208-100-7

Visual Programming paradigm, with a hybrid solution that uses graphic elements (icons and diagrams).

- Intermediate level, composed of mapping and conversion of tasks defined by the user in pervasive applications and by the management of tasks programmed by the middleware;
- Lower level, composed of the set of the middleware services for pervasive environment management and supports the execution of pervasive applications.

A. EXEHDA - middleware to pervasive environment management

Middleware EXEHDA - Execution Environment for Highly Distributed Applications [2] is used to manage the pervasive environment where tasks will be performed. This middleware aims at creating and managing a pervasive environment providing a virtual environment to the user, where applications have the style "follow-me" [7]. Thus, EXEHDA makes it possible to the user to access its computational environment regardless of place and time.

EXEHDA was designed to easily add new components. It is structured in a minimum kernel (required to boot) and services loaded on demand, which are organized in subsystems that manage: (a) distributed execution; (b) communication; (c) context recognition and adaptation; (d) pervasive access to resources and services. EXEHDA sub-systems are formed by services, and each service defines an interface and may be implemented in different ways, suitable to the types of devices that will be supported. Services provided by EXEHDA are customizable at the host level, and are determined by the execution profile, which defines the set of services to be activated and the parameters for their execution, besides associating each service to a specific implementation. This way, services can be "plugged" with no need to have the middleware's kernel modified [4].

A pervasive scenario, as a hospital, defines a personal virtual environment that should follow the user in his movement. This movement can involve both logical mobility (data, code and computation) and physical mobility (resources and devices in use). We refer to this feature of environment and applications as follow-me semantics.

The Pervasive Access Subsystem of EXEHDA is responsible for implementing the resources availability at anywhere, at anytime. To manage these functionalities, the middleware maintains information about user, applications, resources and services. Applications are not installed in the traditional way, meaning that the application's executable code is neither stored in nor managed by the user's virtual environment service. In fact, application installation consists only of copying the application's launching descriptor to the users virtual environment. The executable code for the application is still provided on-demand by the Application DataBase (ABD) service by the time the application starts to execute in a given device. Application profiles (resource descriptors and shared data) are stored in the Application Virtual Environment (AVE), which disappears when the application has finished its execution.

As a user physically moves (by carrying its current device - user mobility - or changing the device being used - terminal mobility), his currently running applications, in addition to the user's virtual environment, need to be continuously available to him, following the user's movement in the pervasive space.It is desirable that, when the system state changes, the middleware dynamically reallocate, reschedule and restructure the (logical and physical) resources available for the application. Application is managed by Distribution Execution subsystem which is responsible by lauching, migration and controling of execution through interaction with other subsystems. Application is on-demand installed by the ABD service in the target device. The executable code for the application is continually provided on-demand according to the Executor service.

The assembling of the context state information, which guides many of the middleware operations and also the application's adaptive behavior, is accomplished by the Context Recognition and Adaptation subsystem, through the cooperative operation of the services Monitor, Collector and Context Manager. The produced context state information feeds both functional (that modifies the code being executed) and non-functional (related to scheduling and resource allocation) adaptation processes, which are managed by the AdaptEngine and Scheduler services respectively. The adaptation model adopted is collaborative and it is reached by two forms: (i) adaptation commands, by explicit calls to some of the middleware's services, and (ii) adaptation policies implicitly guide middleware's operations. Adaptation policies are in the form of XML documents, deployed together with the application's code when it is installed in the BDA pervasive repository. Typically, adaptation policies are defined at development-time by the application designer.

With respect to communications, EXEHDA currently provides, through the services Dispatcher, WORB, and CCManager, three types of communication primitives, each addressing a distinct abstraction level. The Dispatcher service corresponds to the lower abstraction level, providing message-based communications. Message delivering is done through per-application channels, which may be configured to ensure several levels of protection for the data being transmitted. Protection levels range from data integrity, using digital signatures, to privacy through encryption. Additionally, the Dispatcher uses a checkpointing/recovery mechanism for the channels, which is activated when a planned disconnection is in course. This feature may or may not be activated by the upper communication layers depending on its particular demands. In order to make easier the development of distributed services, EXEHDA's also provides an intermediary solution for communications, based on remote method invocations, through the WORB service.

 ISBN: 978-1-61208-100-7

The programming model supported by WORB is similar to Java RMI, though it is turned to the pervasive environment while RMI is not. Specifically, WORB remote method invocations, differently from Java RMI, do not require the device to keep connected during the entire execution of the method in the remote node. At a higher level, the EXEHDA's CCManager provides tuples-space based communications. It builds on the WORB service, which already handles planned disconnections, providing applications with an anonymous and asynchronous communication model.

B. Adapting the EXEHDA to manage the personalized clinical task

EXEHDA was projected in modular way to easy the future adaptation. Due to the flexible features of EXEHDA, as the integration of new services, the adaptation of the middleware to the new tasks management was modeled as a new subsystem, named Sub-system of distributed task management (SDTM). The SDTM services shown in Figure 2 are: (i) Task Management Service, responsible for managing tasks execution; (ii) Tasks Access Service, responsible for the pervasive access to the user's task repository; (iii) Tasks Context Service, responsible to make available the context information relevant to tasks; (iv) Inference Service, responsible for the identification and activation of tasks based on context or schedule; (v) Active Tasks Service, responsible for the pervasive access to user's active tasks; (vi) Interception Service, responsible for connecting SDTM to a system of electronic records, which makes the access available to clinical applications.

The *Tasks Access Service* (TAS) role is to provide pervasive access to the tasks repository of each user, as well as to the repository of sub-tasks, which is unique for all the system, once they are not changeable. Through this service, the Task Management Service searches the list of tasks available to the user when his session starts, and the customizable ontological description [5] of tasks to be instantiated in pervasive applications and start to run. Besides, this service makes available access to information about tasks execution, which support the user's preference processing. The Tasks Access Service use the Pervasive Access Sub-system from EXEHDA to access the user's virtual environment.

The *Tasks Context Service* encapsulates, within objects used by tasks and sub-tasks, context information obtained by the Context Recognition Sub-system from EXEHDA. In our perspective, context is related to users, location, time and resources. Thus, context can be handled in a simpler way by the sub-tasks programmer and becomes more understandable from the user's point of view, who will access contexts that he considers useful for the accomplishment of tasks. This is necessary because the context provided by EXEHDA is in form of raw data. The goal of this service is to gather this information in form of objects with a simple API.

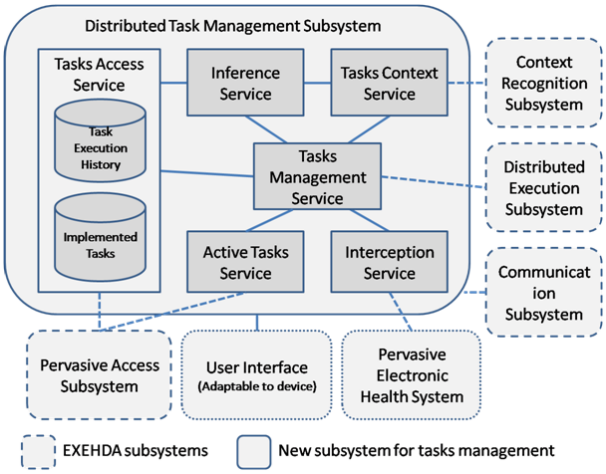

Figure 3. Architecture of the Tasks Distributed Management Sub-system

The *Inference service* processes historic information about the execution of tasks to infer upon the tasks activation, based on context change, in the user's preference in some contexts and scheduling. This way, the system proactivity is improved. This service is still under definition, and was not included in the tool pilot prototype.

The *Tasks Management Service* (TMS) role is to control the execution of user's tasks, enabling him to (i) trigger or schedule the execution of a given task; (ii) stop and later resume a task that was under execution; (iii) cancel a task that does not need to be continued. For that end, TMS makes an API available so that the user can control his tasks. Such API is encapsulated in the graphic interface adequate to the type of device and activity that is being modelled. Besides, TMS manages the migration of tasks, so that they can "follow" the user when he changes device.

The *Active Tasks Service* was created with the goal of keeping the active tasks of each user under a centralized control. For if such control was distributed (carried out by each TMS instance), it would require too much communication and processing to maintain tasks synchronized, making management complex and even unfeasible for some portable devices. Besides, a host is required where to the tasks should be migrated if the user is not using any device.

As a consequence, the Active Tasks Service works in an architecture client/server through HTTP requirements. A host (server) keeps information about tasks active (for each user) and clients make requests to obtain, add, exclude and update tasks. The Active Tasks Service was based on HTTP requests because there is already an infrastructure in middleware for such type of communication. Thus, the Active Tasks Service use the HTTP Service functionality from EXEHDA in the same way as services of the Pervasive Access Sub-system. A specific handler is used to convert service requests into HTTP requests. Therefore, the use of

 ISBN: 978-1-61208-100-7

HTTP communication is transparent from the perspective of other services that use the Active Tasks Service, which makes possible to change the implementation without changing coding of clients from such service.

The *Interception Service* is responsible for handling events generated by the Pervasive Health Care Electronic Register (under definition). In order to fulfill the objective of the project ClinicSpaces, i.e., to include the computational system in the clinical environment reducing its interference in the work of professionals, this service is aimed at intercepting the events of pEHS and process them, triggering applications that will help the physician handle the event, requiring the least possible interaction with the computational system. For example, if an event requires the immediate attention from the professional, the Interception Service may trigger the interruption of running tasks and start a task that help him to meet the event, for example, searching the patient's records (if this is the reason for the event).

The Interception Service can implement abstractions to make the use of pEHS applications easier, which makes possible to easily implement sub-tasks, considering that (i) the ClinicSpaces architecture can be seen as a management agent that facilitates the input of data in the healthcare system used; (ii) each sub-task is associated to at least one application of pEHS, which can be called to interact with the user or can be used only in second plan; and (iii) the system, through passing of parameters, can inform previously data for the application, minimizing the data input from the user.

IV. USE CASE

This case study comprised patients seen at ophthalmological emergency service of the University Hospital of Santa Maria (HUSM). Santa Maria is a tertiary and most important general hospital of the center region of the state of Rio Grande do Sul (Brazil) providing medical care to a population of 1.1 millions inhabitants from 47 cities. It is the biggest hospital, which provides ophthalmological emergency in the region.

In the same way, their patients attend the ophthalmological emergency service mainly being referred by general physicians, nurses or an ophthalmologist, or also by their own decision. There they are examined by an ophthalmologist, treated if necessary and sometimes referred for other center.

Data about patients were collected during interview and ophthalmological examination. When a patient had more than one diagnosis, only the most serious one was listed. Veracity of the emergency was categorized as *true* or *not-true* emergency. A true ocular emergency was considered if there were risk of decreasing or loss vision, as well as cases requiring immediate (same day) evaluation in either an emergency department or an ophthalmology outpatient department due the intensity of symptoms.

In this case we applied the middleware EXEHDA to control daily tasks of an ophthalmologist. This way, he will be able to create (personalize) a task that will help him in this activity, through of Task Edition Interface (see Figure 1). For example, if the ophthalmologist usually first check the latest information about the patient's medical records, he starts the task's construction with a sub-task to search the patient's medical records. The tasks construction tool adds, automatically, a sub-task to identify the patient using RFID for example, since it is necessary to identify which medical records should be accessed. Then, the physician adds a sub-task to view the information in his preferable form.

Continuing the task's composition, the ophthalmologist may add a sub-task to register the information about the patient's care. In addition, a task to request examinations could be added. It will register the request in patient's medical records and forward the request to the laboratory.

Finally, the ophthalmologist could add a task to prescribe the medication or treatment. This task will register the prescription in patient's medical records and will print it. When finished the task construction (personalization), it will be stored in the User Virtual Environment.

A. Task Execution

When the ophthalmologist examines a patient (human activity modeled as a set of tasks), the system interface is accessible and the tasks are available [2]. The patient's examination task is programmed (customized) in advance and triggered by the physician or by changes in context (patient's arrival detection). The Tasks Management Service (TMS) accesses the Tasks Context Service (TCS) to obtain the physician's identification, device configuration and other information required to instantiate the task. Processing the ontological description of the task, TMS finds sub-tasks that compose it and finds their code.

The first sub-task to be performed is the patient's identification. For that end, TCS uses sensors managed by the EXEHDA middleware, through of the context recognition subsystem [2][6], to identify the patient that will be examined. The information that returns is sent to other sub-tasks. Then, the sub-task that searches for the patient's information is triggered in the pEHS linked to the ClinicSpaces architecture. The information is then used as a parameter for the next sub-task, which displays the data from the records in the format the user selected. It is only from this moment that the interaction between ophthalmologist and task starts, because the execution of previous sub-tasks is transparent.

The system's interface checks which tasks are available to the user. These tasks – composed of other tasks and sub-tasks – were validated at the time of creation and customization of these tasks using TMS.

After the doctor reviews the information, he terminates the application (sub-task) and TMS triggers the next sub-task, which will store the ophthalmologist's notes in the electronic

 ISBN: 978-1-61208-100-7

records (pEHS). This sub-task calls a specific application of pEHS, and parameterizes it with the ophthalmologist's and patient's identification. At the end of this sub-task, the next – tests request – is triggered. This will be a task created by the user or obtained from the system. This task is independent, and can be performed in isolation.

The tests request task stores the ophthalmologist's request in the patient's records and send it to the lab, which is integrated to pEHS. At the end of this task, TMS triggers the treatment prescription task, which is stored in the patient's records and sends it to the closest printer, obtained through TCS.

The module decomposes and recomposes the tasks flow at the execution point. The TMS decomposes a task into sub-tasks. A task is represented by an XML file that links to the XML files of the sub-task. The decomposition is done when the system reads the XML files, instantiates objects (sub-tasks), sets the parameters, and executes them.

V. Prototyping and Evaluation

In order to validate the concepts discussed in this paper and evaluate the impact of task management in middleware and in execution of pervasive applications (tasks), a pilot prototype of the Sub-system of Distributed Task Management was developed. This prototype is composed by the Task Management Service, Tasks Access Service, and Active Tasks Service. All these services were were developed in J2EE, as well EXEHDA middleware and its applications. These services were integrated with the EXEHDA middleware and instantiated in two nodes: one base node, that is responsible for the cell's management; and one common node (client), representing a user's device.

The user's application, through API of the SDTM, searches the tasks available for user and the tasks that were initiated and do not terminated in previous session. As this information is located in base node, the local instances of the Tasks Access Service and Active Tasks Service generate requests for their remote instances. Therefore, the operations of these two services were monitored in experiments to determine the impact of them on the system, in terms of startup time of tasks and number of remote requests. On the other hand, the tasks execution management is done, locally, by the Tasks Management Service. Thus, this service was monitored in terms of extra processing to control the tasks.

The experimentations showed that the impact of new services, necessary for the tasks management, was minimal, both in terms of middleware as user's applications. Moreover, the evaluation of services indicated points in the prototype that can be improved, as the number of remote requests, that although they are acceptable, they can be reduced.

Therefore, in general, the evaluation of the experimentations showed that the proposal to extend a middleware for pervasive computing to manage the daily tasks of health

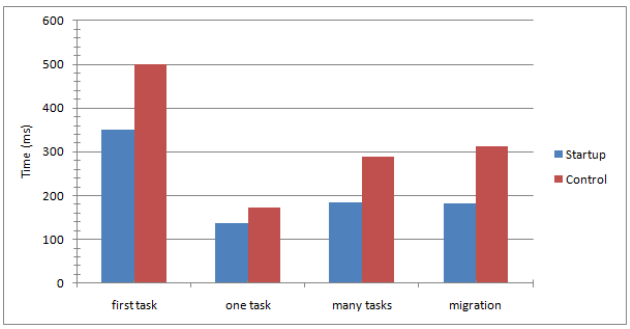

Figure 4. Startup and control time for different situations in tasks management

professionals is feasible and promising. The results were very satisfactory, since the prototype worked together with EXEHDA, performing their functions without generating overhead in the middleware.

As can be seen in Figure 4, the average time for initialization and control of tasks are below a millisecond. In the graph can be noted that the migration of tasks do not have significant influence on time for management. However, in the execution of the first task (after system boot), the management time is influenced for the startup of the middleware services. Moreover, the graph shown that even in the management of concurrent tasks (5 to 10 tasks), the overload of the system is minimal.

VI. Related Works

The ISAM [6] project, as well as Gaia [14], originated a middleware for pervasive environment management that integrates the premises of grid computing, mobile computing and context-aware computing. EXEHDA [2][7] middleware creates and manages a virtual environment for the user, in which applications are executed in a distributed and context-adaptable way. However, it does not approach the concepts of daily activities.

The concept of Task-based Computing was introduced by the Aura project [3] so that the middleware can manage the pervasive environment proactively in a way that the user can keep the continuity of his activities, at the same time as he moves from one place to the other. In this project, tasks are modeled as collections of services; the service description is used to find the necessary resources or reconfigure the middleware to run the task. The Aura system is automated and proactive, and it urges the user to perform his tasks according to predefined and preprogrammed settings, that is, it does not allow for the customization of tasks (services), which increases the interference of the system in the environment.

The Activity-based Computing [10] project brings a proposal for the use of Task Based Computing in health care environments. A framework was developed in this project that provides the required infrastructure to perform services

 ISBN: 978-1-61208-100-7

that will support the features inherent to the health care professionals activities. Thus, services can be initialized, suspended, stored, resumed in any device and at any time, as well as they can be sent to other users or shared among different users. The project aims also to allow developers of clinical applications to incorporate in its programs support to mobility, interruptions, concurrent activities and cooperation.

Some ideas taken from such projects had some influence in our work. We highlight the use of EXEHDA middleware and the Activity-based Computing project, which guided the definition of concepts related to tasks in the health care area. However, as one can observe in Figure 5, none of the projects focuses on the final user as the ClinicSpaces does. This project differential is that it enables ophthalmologist to customize tasks and balance the proactive execution with the control over the execution of tasks.

	Pervasive Computing		Task-based Computing		
	Mobility	Context	Automation	Customization	Control
Gaia	X	X			
ISAM	X	X			
Aura	X	X	X		X
ClinicSpaces	X	X	X	X	X

Figure 5. A comparison among ClinicSpaces' related works

VII. Conclusion

Pervasive computing promises ubiquitously support to users in accomplishing their tasks. Hardware and networking infrastructure to make the pervasive computing come true are increasingly becoming a reality. While this scenario represents an attractive computational environment, it poses three major challenges in the form of heterogeneity, dynamism, and execution context changes, all these are intensified because of user's mobility. Our project, named ClinicSpaces develops a pilot tool that will allow the physician to configure/program computational tasks that will help him in daily professional activities. Thus, we intend to minimize the avoidance of these professionals with relation to the use of computational systems which have a pre-established behavior.

ClinicSpaces' architecture is innovative in that the system can be customized according to the user's characteristics, in a way that professionals can use the computational system as if it were an assistant that would help him in his activities, instead of imposing the way how he should perform his work. In order to make such ideas real, the present study aimed at adapting a pervasive space management middleware to support tasks execution using EXEHDA, because it employs a lot of strategies in its services to allow the adaptation to the current state of the execution context, such as on-demand adaptive service loading and dynamic discovery and configuration.

The project is under development, it was implemented but not yet tested by physicians. The actual utility will be verified in a next field research. One of the contributions of this work is the service architecture required to adapt a pervasive environment management middleware to the management of tasks in this environment, fulfilling some requirements of activities-oriented computing to the final user. Another outcome from this work is the pilot prototype of an assistant (implemented as a middleware for pervasive environment management) that helps physicians in tasks management (daily activities). The next step of our research is to assess the usability of this solution in a real environment, after the insertion of the Pervasive Electronic Health Care System (pEHS) and this architecture.

Acknowledgements – This work has been partially supported by Brazilians Agencies FINEP/CNPq/MCT.

References

[1] A. Ranganathan and R.H. Campbell, *Supporting Tasks in a Programmable Smart Home*. In From Smart Homes to Smart Care, v. 15. Amsterdam: IOS Press, 3-10, 2005.

[2] A. Yamin, I. Augustin, L.C. Silva, R.A. Real, and C.F.R. Geyer, *EXEHDA: adaptive middleware for building a pervasive grid environment*. In Frontiers in Artificial Intelligence and Applications: Self-Organization and Autonomic Informatics (I), vol. 135, pp. 203-219. IOS Press, 2005.

[3] D. Garlan, P. Steenkiste, and B. Schmerl, *Project Aura: Toward Distraction-free Pervasive Computing*. In IEEE Pervasive Computing. New York, NY, 2002. pp. 22-31.

[4] G. Ferreira, I. Augustin, G.R. Librelotto, F.L. Silva, and A.C. Yamin, *Middleware for management of end-user programming of clinical activities in a pervasive environment*. In Proceedings of the 2009 Workshop on Middleware for Ubiquitous and Pervasive Systems. ACM New York, USA. 2009. pp 7-12.

[5] G.R. Librelotto, J.B. Gassen, M.C. Silveira, and L.O. Freitas, *OntoHealth - Um framework para o gerenciamento de ontologias em ambientes hospitalares pervasivos*. In: II Workshop on Pervasive and Ubiquitous Computing, 2008. pp. 31-42.

[6] I. Augustin, A.C. Yamin, L.C. Silva, R. Real, G. Frainer, G. Cavalheiro, and C. Geyer *ISAM: joing context-awareness and mobility to building pervasive applications*. In Mobile Computing Handbook. I. CRC Press, New York, NY, 2004.

[7] I. Augustin, A.C. Yamin, and L.C. Silva, *Building a Smart Environment at Large-Scale with a Pervasive Grid Middleware*. In Grid Computing Research Progress. Nova Science Publishers, Inc, 2008. pp. 182-186.

[8] J.B. Jorgensen and C. Bossen, *Executable use cases: requirements for a pervasive health care system*. IEEE Software. 21(2):34-41, 2004.

[9] J.E. Bardram, *Hospitals of the Future: Ubiquitous Computing support for Medical Work in Hospitals*. In Proceedings of 5th International Conference on Ubiquitous Computing, pp. 1-7. 2003.

 ISBN: 978-1-61208-100-7

[10] J.E. Bardram and H.B. Christensen, *Pervasive Computing Support for Hospitals: An overview of the Activity-Based Computing Project*. In IEEE Pervasive Computing, v. 6, issue 1, 44-51, 2007.

[11] J.E. Bardram and T.R. Hansen, *Context-Based Workplace Awareness*. In Computer Supported Cooperative Work. v. 19, issue 2, 105-138. Kluwer Academic Publishers. 2010.

[12] K.T. Unruh, M. Skeels, A. Civan-Hartzler, and W. Pratt, *Transforming clinic environments into information workspaces for patients*. In Conference on Human Factors in Computing Systems. SIGCHI: ACM Special Interest Group on Computer-Human Interaction. 2010. pp. 183-192.

[13] M. Kaenampornpan and E. O'Neill, *Integrating History and Activity Theory in Context Aware System Design*. In Proceedings of 1st International Workshop on Exploiting Context Histories in Smart Environments, Munich, 2005.

[14] M. Roman, M. Román, C. Hess, R. Cerqueira, R.H. Campbell, and K. Nahrstedt, *Gaia: a Middleware Infrastructure to Enable Active Spaces*. In IEEE Pervasive Computing. New York. 2002. pp. 74-83.

[15] U. Varshney, *Pervasive Healthcare*. IEEE Computer, 36(12): 138-140, 2003.

 ISBN: 978-1-61208-100-7