Electronic Communications of the EASST
Volume 17 (2009)

Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen 2009 (WowKiVS 2009)

Implementation of a User-Centric Context-Aware Playground

Sian Lun Lau, Niklas Klein, Andreas Pirali, Immanuel König and Klaus David

11 pages

Guest Editors: M. Wagner, D. Hogrefe, K. Geihs, K. David
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Implementation of a User-Centric Context-Aware Playground

Sian Lun Lau, Niklas Klein, Andreas Pirali, Immanuel König and Klaus David

comtec@uni-kassel.de, http://www.comtec.eecs.uni-kassel.de/
Chair for Communication Technology
University of Kassel

Abstract: Since the introduction of the ubiquitous computing vision by Mark Weiser, we observed quite some research from various R&D projects as well as groups that focused on different aspects in the area of context-awareness. However, there is yet no result which we can easily apply in a real environment. The context-aware service development and deployment process seldom focused on reusability and realistic implementation issues. We believe these issues can be tackled by proposing a complete context-aware system package. In this package both developers and users will be supported. This idea is realised at our department as a context-aware playground. This paper explains the observation on current work and the shortcomings, and proposes how the context-aware playground approach can be a solution.

Keywords: Context-Awareness, Implementation, User-centric, Service Creation

1 Introduction

In the past decade we witnessed the emergence of context-awareness related work. Initiated by the vision from Mark Weiser [Wei99], the different efforts aim to realise an environment where devices and applications assist users[1] in their everyday living by sensing and understanding available contexts. Dey defined context in his work as "any information that can be used to characterise the situation of entities" [Dey01]. Some of the prominent research results are Dey's Context Toolkit and Schilit's context-aware prototype using the ParcTAB infrastructure [SAW94]. There are also efforts in bringing context-awareness into the area of smart environment, such as offices and homes. The Gator Tech Smart House [HME+05], the Aware Home of the Georgia Tech [KOA+99] and the MavHome Project of the University of Texas at Arlington [DC06] researched on smart home where intelligent systems use different acquired contexts from respective environments to provide automation and adaptation.

As we analysed these work, we realised that most of the work focused on the demonstration of the ideology of context-awareness. Among them we observed an abundance of proof of concept results, but they worked only in a closed environment. It was partly due to the absence of suitable non-prototype devices, such as ready-to-deploy sensors and hardware. In the early stages many of the scenarios could only be tested in simulation and mock-ups. The various context-aware platforms and architectures were only usable in respective chosen scenarios. The software components are statically created according to the implementation design, hence making them hard-coded applications that are only applicable in the designed scenarios [DHB+04, ZBM04].

[1] In this paper the word user is synonymous with end-user

As far as we know, there is a lack of practical implementation of context-aware systems where reuse and reimplementation of tools and platform in real scenarios can be carried out. As an example, it is still not possible to select an existing context-aware system and deploy any desired sensor device and actuators to test or achieve context-aware adaptation for a new application scenario. This was due to the nature of the past and current realisation approaches - there was no focus on achieving a complete context-aware hardware and software package for all parties involved in this area. The word *complete* here does not mean that the solution can solve every single possible need. We define a complete package as a solution that considers the needs and requirements for a viable implementation that satisfies all parties involved in the development and deployment cycle.

We foresee that the users of the proposed system can decide individual application domains that suit their needs. For example, user John can select freely combination of three sensors to enable adaptation of his lighting systems, while another user Jane can select two other contexts on top of the three sensors used by John for a multi-modal news display service. In other words, context-aware system designers and developers do not make the decision on the potential applications - the users do. The former are responsible to focus on component and functionality implementation instead. Similar to the usage of current computing systems, hardware and software provider deliver different components that build a computer, while a user decides how the computer will serve him - it can be used as a modern typing machine, a multimedia device, a gateway to the World Wide Web, all three of them or something completely different.

The above mentioned vision is what we call a user-centric context-aware system. This system provides a complete and open solution that integrates needed devices and functionality for potential users. Users are free to use this system in various environments, such as their home, office, vehicle and etc. Now is a good time to combine past and current experiences and technologies to realise this vision. It would be a solution that considers both theoretical and practical requirements. The implementation of this vision is partly realised in our ongoing smart meeting room project, where we utilize various market available sensors and context processing approaches to perform adaptation and prediction of service behaviours common to a meeting room environment.

We see this implementation as a context-aware playground, where we wish to achieve two goals:

1. We would like to investigate how developers can be supported in the process of development. This includes the development of different actuators for a given environment, and integration of sensor devices.

2. We need to support users creating desired context-aware applications using components provided by the developers. Users should not be overwhelmed by the complexity of technology, since they do not necessarily posses sufficient technical expertise.

This paper will present our proposal of the solution. The structure of the paper is as following: Section 2 presents the concept and approach, where the requirements and the proposed architecture will be detailed. Section 3 elaborates our proposed implementation. In the Section 4 we will conclude the paper and present our next steps.

2 Our Concept/Approach

If we wish to offer a context-aware system as a playground for potential users to create their own applications, the underlying architecture needs to support the respective requirements. We use rooms as a general depiction of a possible scenario in order to highlight the requirements we have observed throughout our work. These requirements will be applied in the design of the context-aware system architecture. In this section we will first present the requirements using rooms as an implementation scenario. This is followed by a brief overview of the proposed architecture. We will also discuss shortly the rationale behind this idea.

2.1 Scenario and Requirements

Rooms normally serve a special purpose like office rooms, bath rooms or meeting rooms. Depending on the purpose of a room, there are unique situations in the room, like meetings or presentations in a meeting room. These situations have well-defined characteristics and the persons in the room have recurring tasks to fulfil. If a system is able to detect these situations through the use of sensors, it can then adapt the room to the situation. Adaptation in this case could be simple things such as the adjustment of the lights or the temperature. It is also possible to directly assist participants in the room, e.g. by automatically offering voice controlled meeting protocols or starting a projector and displaying a presentation, as soon as someone stands in front of the presentation screen.

Today, rooms, like office rooms or meetings rooms, and even living rooms, are equipped with a lot of electronic devices. There are television sets, phones, hi-fi systems, projectors, digital picture frames and many more. These electronic devices nowadays are often equipped with fixed or wireless LAN connectivity and have hard disk drives installed. The devices often have full-fledged operating systems like Microsoft Windows or Linux. Additionally, there are home automation devices that enable the control of light, heater and roller shutters. Therefore, there are already many potential devices to be controlled in an automated way.

On the other hand, there are various sensors available on the market. Light sensors, temperature sensors, acceleration sensors and humidity sensors as well as light barriers are cheap electronic components today. Video and audio can serve as well as rich sources of context data and are still active research areas. Nevertheless, there are still very few solutions on how to combine the available sensors and actuators as a complete package. There lacks a convenient package that is deployable as a product to enable context aware rooms and is easily manageable by the costumers.

To achieve this we have to overcome several obstacles. From a consistent architecture with well defined APIs for sensors and actuators to a convenient tool set for developers and users we have to provide well designed and concerted components.

Based on this basis, the architecture has to deal with five important requirements:

1. Support for heterogeneity
   The sensors in the room are connected with each others in various ways. Some have wireless network adapters or Ethernet adapters and are connected over network with a central server. Other uses technologies like ZigBee, Bluetooth or USB. Some of the sensors are directly connected to the central server, others to distributed devices, which are connected

to the central server over wireless or fixed network connections.
Therefore, the first requirement is to hide the heterogeneity to the developers in a way that allows the developer to concentrate on his specific problem without having to spend time on communication protocols.

2. High level context
   The raw sensor data normally is too technical for users to understand, e.g. a user may want to refer to a context like ”warm” instead of referring to ”between 20° Celsius and 28° Celsius”. Another example is the interpretation of GPS coordinates to location names, such as [51.311182,9.473184] can be represented as ”My Office”.
   Therefore, the second requirement is the support for high level context. High level context is context data calculated from raw sensors data to represent the context in more abstract way, which is more readable and understandable to the user.

3. Automatic intelligent distribution
   Depending on the number and complexity of calculations, it is possible that the consumption of time and energy is critical.
   Therefore, the third requirement is the support for automatic intelligent distribution (AID). AIDs goal is to distribute the calculation on the available devices to reduce the burden of the devices which are less capable in terms of processing power and energy resources.

4. Dynamic composition
   While many sensors are cheap components, when they are deployed there might be two potential issues. On one hand, similar sensors deployed in the same environment can be seen as redundant. On the one hand, some of these sensors may fail during operation. Furthermore, some sensors, such as sensors found on smart phones, can be introduced into the environment when users enter a given area.
   Therefore, the fourth requirement is the support for dynamic composition, which is the ability to change the sensors used by an application at runtime. E.g. if the rooms light sensors fail, it should be possible to detect the failure and use a light sensor of an entering smart phone.

5. User usage and composition support
   As mentioned earlier, the system developers do not fix the usage domain of the potential applications. They concentrate on the development of software components that provide specific functionalities. On the other hands, hardware developers provide compatible sensors and actuators that can be reused in the designated system. Users will use visual tools to compose different applications that will carry out the needed context-aware adaptation.
   Therefore, as the fifth requirement, the architecture needs to also provide tools for users to create desired context-aware applications.

One can foresee that such an architecture should consist of at least the following aspects. Different components such as sensors, actuators and context processing mechanisms can be found in the system. These components are the basic units that will be used to build context-aware applications. Service execution environments (SEEs) will act as supporting service platforms, where the different components would be deployed and executed. The development of these

components will be carried out by professional developers, and tools should be design to assist them to contribute new sensors, actuators and calculation components for the designated system. Last but not least, we also need to provide users the means to understand and to manage the behaviour of a context-aware service by selecting sensors, actuators and calculation components and combining them.

2.2 Architecture

2.2.1 Three Tier Architecture

The architecture is based on the FAME^2 middleware framework [WDD04]. This middleware framework is implemented in Java and has a very small footprint. It runs on small devices like PDAs and smart phones as well as on servers. FAME^2 follows the service oriented architecture principles and provides functionalities as independent components. This is exactly what is needed to support automatic intelligent distribution, dynamic composition and high level context.

The architecture is a three-tier architecture. The basis is built on the FAME^2 core, which has three functionalities. The core manages the life cycle of the components, which are loading, initiating, starting, stopping and updating components at runtime. The updating allows replacing a component with another one which provides the same interface at runtime in a transparent way. This is used to replace failed sensors with similar ones. Additionally the core enforces the security policy to control the access to components. The core handles also the communication between the components. Communication is implemented using the publish/subscribe pattern, which is very well suited for a context-aware environment. In this way it is not necessary to poll the sensors data continuously. Instead, the components within the SEE will be informed when context changes take place.

The second tier provides a set of basic functionality, which is needed for the management and control of the context-aware applications. In the playground we call them basic components. These basic components consist of a repository, a discovery tool, a rule engine, and a management interface. The repository contains all sensors, actuators and calculation components. Developers can search the repository to reuse existing components instead of implementing needed functionalities from scratch. The rule engine interprets rule languages like RuleML and thus can execute user defined rules. These rules are processing plans, which describe the data flow and data processing from sensors through calculation components to applications. In this way it is possible to define context-aware applications without writing code, but by combining predefined components [LKP+08].

On the top tier there are the components that build context-aware applications. In our approach we view context-aware applications as logical entities that consist of different components. Each component performs specific functionality. We define three types of components. Firstly, a context-provider component provides sensor or context data. Secondly, there are also actuator components that carried out a specific action. Last but not least, we also foresee the need of context processing components, where output from the context provider components are processed to produce inferred results. At the same time, if a composition of components produces interpreted contexts, it can be also reused in the platform as a context provider component. The communication between different components can be defined in program code or expressed in

logical rules. The latter will be interpreted via a rule engine in order to execute the context-aware application.

2.2.2 Context Processing in a Context-Aware Environment

In order to calculate high level context, we allow the composition of reusable components, as described in chapter 2.2. The composition can be seen as a directed acyclic graph, where the sensors are the sources and the applications are the sinks. The nodes on the path from source/sensor to sink/application are the calculation modules. Every component has a unified interface which provides the functions to process new events, when the data of the input components has changed and functions to publish the outcome of the calculation to the output components.

The processing of the graph is controlled by a central master device, which is responsible to distribute the graph through the devices in a way that saves energy and uses available processing power. To enable synchronised processing the graphs can be divided into processing rounds, where in every round only nodes of the same level are processed [KLPD08]. Node levels in a directed acyclic graph are defined as the length of the longest path of the node to a sink.

2.2.3 User's Role in the Environment

While current home automation solutions offer mainly time and temperature controlled behaviour, the user is not in control of the behaviour let alone able to create new applications on his own. Some solutions provide customization possibilities for users, but they are usually fixed schedule-based approaches. In our architecture, we want to enable the user to create, manage and control applications by manipulating the processing rules (see chapter 2.2) with a graphical tool. In this way users can reuse the provided components and can express their application needs. The applications are not executed based on a time table, but due to the changes of the contexts in the environment.

The graphical tool assists the user to select components offered in a platform, such as sensors, calculation components or actuators, from the repository and to "draw" an application by connecting the components to define the data flow. This gives the user the control and understanding to really be satisfied with the "product" context aware room.

2.3 Rationale of a complete context-aware system package

Through our proposal of the architecture we wish to promote a complete context-aware package development life cycle. In the near future we foresee the possibility to obtain both devices and application components that can be deployed directly in a context-aware system in a plug-and-play manner. Contrary to current products, where most of the intelligent systems are mostly static and closed, we wish to include users' involvement into the life cycle. The separation of concern approach applied allows different parties involved in the life cycle to concentrate on their own areas. The components in the platform can be reused for different purposes, as long as the needed functions are the same. At the same time, faulty components can also be replaced by similar components available at runtime, so that the execution of applications will not be disrupted.

Besides that, such proposal can also be used in prototyping innovative products for future scenarios. For developers and service providers the architecture enables flexible composition of applications, which one can easily build in order to test and evaluate newer ideas. The components offered in a system can be freely combined, and one can then execute the composed service for his desired purpose. We believe such system will encourage a shorter prototyping process, and useful products can be then delivered to potential users faster.

3 Proposed Implementation

To achieve the vision as proposed in chapter 2, we first will implement the playground in a meeting room, to act as a proof of concept and as a playground to test new ideas for context aware application. Therefore we will use sensors and actuators available on the market and incorporate them into our platform. Together with the tools already developed in previous projects [LKP+08] [s4a] [spi], we will provide a complete context-aware package including the tools for developers and users to realise and test new ideas and applications.

The focus of this playground is to realise a complete environment instead of concentrating on special aspects of the architecture or the calculation algorithms. Therefore we will use existing solutions where ever possible to prove the feasibility of realising a context aware room with cheap and easy available components and tools, which are well known from the research field of service creation.

Once such a complete playground is realised, it will be straightforward used to implement new features in the architecture and to test them in a real world scenario as well as to gather user feedback. New applications and algorithms can also be added to the playground and tested with real data obtained from the environment.

3.1 Context Providers Components

Context provider components are application components that provide high level context data to an application. This can be a sensor data as well as an aggregation of sensor data, processed in several steps. We use two different systems for the actual implementation, but the approach is not limited to this systems. The first system is the ”Phidgets” system [2]. ”Phidgets” offers a range of sensors connected through USB to a computer. The sensors available include temperature sensor, light sensor and accelerometer.

The second system currently in use is Sun SPOT (Sun Small Programmable Object Technology) [3]. It is a wireless sensor network (WSN) mote developed by Sun Microsystems. Each Sun SPOT has various sensors such as a three-axis accelerometer, a temperature sensor and a light sensor. Furthermore there are also five general purpose I/O pins for external input or output usage, e.g. control of a servo controller. The Sun SPOT can communicate with a receiver via the IEEE 802.15.4 standard or it can also be connected through USB directly with a computer.

The states of different actuators in the system can also be used as another source of context. This is useful in learning about actions executed and in informing both users and system what

[2] http://www.phidgets.com/, last visited on 25th October 2008
[3] http://www.sunspotworld.com/, last visited on 25th October 2008

the system has performed or is going to do. Further explanation on actuators will be presented in the next subsection.

There should also be context provider components that filter and interpret aggregated sensor data. For example, users' situational context can be processed and observed to be classified into high level contexts. Such context providers will implement suitable process algorithms in order to deliver accurate interpretations. The interpreted high level contexts can be used by the users to compose newer services that do not rely solely on sensor data.

3.2 Actuators Components

As actuators we use home automation components to control the light and the heater in the room. Additionally we use every device in the room with network capabilities. E.g. at the room doors are touch screen displays mounted to show information about the room context and a projector at the room can be switched on or off. It is also possible to play audio files on a hi-fi system in the room. Other actuators can control the aperture of the windows or the closing of the roller blinds. In our playground we will build actuators on different devices found in our meeting room. It is also foreseen that users can instruct the system to invoke the desired actuators based on different context provider components.

3.3 Tools

As mentioned above, there are also different tools offered by the platform. We developed the End-User Studio (EUS) to enable context-aware application creation using logic rules. Users express their needs via "if-then" relationships. We call this technique "Draw an Application" - the users basically draw applications that answer their needs. The drawn logic expressions are graphical representations of high level abstraction of service behaviours. These abstractions do not necessarily have direct relations to any of the components found in the SEE, since components are described with high level abstraction descriptions. The users can search for available components in the SEE and compose applications using these components. They can also start composition free-hand - by drawing rule expressions with desired keywords without searching for available components beforehand. Figure 1 is an example how a drawn rule may look like:

The EUS provides also the function of service deployment. After the user has created some applications, he can either test the composed applications or store them in the platform. Since the application wishes are expressed in logic rule policies, we have chosen RuleML [rul] as the representation language for the composition of applications. In this manner, any EUS that produces applications expressed in RuleML (with supported language versions) will ensure interoperability.

The different components in the playground will be developed by professional developers. We propose the use of a development tool plug-in for the well known IDE Eclipse to assist developers. Platform-specific requirements and routines can be assisted, simplified and automated through the use of the plug-in. Wizard dialogues can also be used for components creation, description and deployment processes. All these features will minimize the effort required for the component development on a given platform, where workload of developers in the development process can be reduced, and hence allowing developers to focus on the component functionality

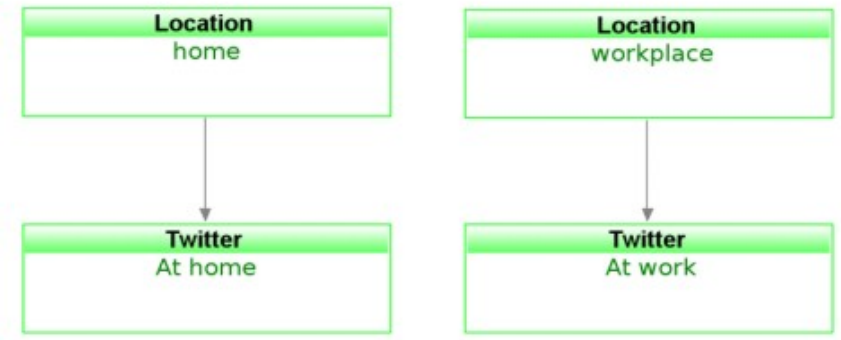

Figure 1: Example of a created application

instead.

We have also developed an agent called Business Rule Evaluator (BRE) to execute the application. As proposed in our past projects, context-aware applications can be defined using simple "*if-else*" logical rule expressions. As an example, a user can create a service that plays music on respective devices according to his situational contexts. A rule engine can be used to process the available contexts to decide whether which actuator component should be started. The BRE is responsible to load the rule-based application expressions, and finds out which context provider it should subscribe to. As soon as context changes take place, the BRE will be informed and the rule inference process will be triggered. In cases where context conditions are satisfied, the BRE will need to then invoke the respective actuator components.

4 Conclusion and Outlook

Based on our observation on existing research and our previous project results, we propose the implementation of a practical context-aware system. The system consists of a complete context-aware hardware and software package that supports both developers and users of the system. This idea resulted in a user-centric context-aware playground, which we have partly implemented in our smart meeting room project.

As next steps we wish to finalize the implementation of the various comsponents planned for this playground. Different sensors have been integrated into the system. Actuator components are also planned and will be developed. At the same time we will also add some context processing components, such as context reasoning and prediction components.

Once the components are ready, we plan to perform evaluation on the context-aware application creation. The expected result will help us to verify the ideas proposed in this paper. We wish to also exploit this playground in improving the various algorithms used in context processing components we currently have. The playground will be also our test bed for further context-aware related research.

5 Acknowledgement

This work has been partially performed in the MATRIX project, which is partly funded by the German "Bundesministerium für Bildung und Forschung" (BMBF). The authors would like to thank their project partners for their discussion.

Bibliography

[All] O. M. Alliance. OMA Enabler and Reference Releases. Available online at http://www.openmobilealliance.org/Technical/released_enablers.aspx visited on August 25th 2008.

[DC06] S. K. Das, D. J. Cook. Designing and Modeling Smart Environments (Invited Paper). In *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*. Pp. 490–494. IEEE Computer Society, Washington, DC, USA, 2006. doi:http://dx.doi.org/10.1109/WOWMOM.2006.35

[Dey01] A. K. Dey. Understanding and Using Context. *Personal Ubiquitous Comput.* 5(1):4–7, 2001. doi:http://dx.doi.org/10.1007/s007790170019

[DHB+04] A. K. Dey, R. Hamid, C. Beckmann, I. Li, D. Hsu. a CAPpella: Programming by demonstration of context-aware applications. In *in Proceedings of CHI 2004*. Pp. 33–40. 2004.

[HME+05] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, E. Jansen. The Gator Tech Smart House: A Programmable Pervasive Space. *Computer* 38(3):50–60, 2005. doi:http://dx.doi.org/10.1109/MC.2005.107

[KLPD08] B. N. Klein, S. L. Lau, A. Pirali, K. David. DAGR DAG based Context Reasoning: An Architecture for Context Aware Applications. In *Proceedings of the 8th International Workshop on Applications and Services in Wireless Networks (ASWN 2008)*. Kassel, Germany, October 2008.

[KOA+99] C. D. Kidd, R. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. D. Mynatt, T. Starner, W. Newstetter. The Aware Home: A Living Laboratory for Ubiquitous Computing Research. In *CoBuild '99: Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*. Pp. 191–198. Springer-Verlag, London, UK, 1999.

[LKP+08] S. L. Lau, N. Klein, A. Pirali, I. Koenig, O. Droegehorn, K. David. Making Service Creation for (Almost) Everyone. In *ICT-Mobile Summit 2008*. Stockholm, June 2008.

[LKP+09] S. L. Lau, N. Klein, A. Pirali, I. Knig, K. David. Implementation of a User-Centric Context-Aware Playground. In *Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen 2009 (WowKiVS 2009)*. Volume 17. Electronic Communications of the EASST, 2009.

[rul] The Rule Markup Initiative. Available online at http://www.ruleml.org visited on August 25th 2008.

[s4a] S4ALL-DE Project. Available online at http://www.s4all.eecs.uni-kassel.de visited on October 25th 2008.

[SAW94] B. Schilit, N. Adams, R. Want. Context-Aware Computing Applications. In *WMCSA '94: Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*. Pp. 85–90. IEEE Computer Society, Washington, DC, USA, 1994. doi:http://dx.doi.org/10.1109/WMCSA.1994.16

[spi] IST-SPICE Project. Available online at http://www.ist-spice.org visited on October 25th 2008.

[WDD04] B. Wuest, O. Droegehorn, K. David. The Fame2 Platform Concept: Moving Platforms to the Mobile. In *Proc. 5th Int. Conference on Internet Computing (IC'04)*. Pp. 423–430. LasVegas, USA, 2004.

[Wei99] M. Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.* 3(3):3–11, 1999. doi:http://doi.acm.org/10.1145/329124.329126

[ZBM04] T. Zhang, B. Brgge, T. U. Mnchen. Empowering the User to Build Smart Home Applications. In *Proceedings of 2nd International Conference on Smart homes and health Telematic (ICOST2004), Singapore, 2004. Palviainen, Marko Series*. Marko Palviainen, 2004.