Finite-State Machine Based Distributed Framework DATA for Intelligent Ambience Systems

1MATEJ ROJC, 2IZIDOR MLAKAR
1Faculty of Electrical Engineering and Computer Science, University of Maribor, 2Roboti c.s. d.o.o
1Smetanova ulica 17, 2Tržaška cesta 23
SLOVENIA
1matej.rojc@uni-mb.si, 2izidor.mlakar@revolutionary-robotics.com

Abstract: - In this paper, a finite-state machine based distributed framework DATA used for development of intelligent ambience systems is presented. Event-based distributed framework DATA enables development of efficient, clear and flexible operation over several electronic devices, mobile platforms and mobile units that can be part of complex intelligent ambience. Framework DATA enables developers to specify, describe and implement several behaviours of the intelligent ambience and individual modules in a fast and flexible way. Adding new users, intelligent ambience system’s modules, mobile platforms, mobile units and electronic devices, can be performed without intervention into the code. Distributed framework DATA supports data flow between individual system’s modules that is easily understandable to the users. Framework DATA is java-based and supports several configuration possibilities (regarding behaviour and architecture), distributive operation for more intensive tasks, and flexible communication mechanism between intelligent ambience system’s modules. All these features are implemented by using finite-state machine formalism and XML markup language as will be presented in the paper.

Key-Words: intelligent environment, distributed systems, finite-state machines, framework DATA

1 Introduction

In the context of domestic environment and human-machine interaction, EU funded project Ambient Intelligence for the Networked Home (AMIGO) can be regarded as one of very important achievements in intelligent ambience research field. Amigo architecture enables easy and effective integration of services in today’s home through middleware that dynamically integrates several heterogeneous systems to achieve interoperability between services and devices. Its environment characteristics are: multiple-user aspect and distributed applications for sharing information within a location and between different locations. Intelligent ambience developers can work with classes provided in C, C# and java. The Amigo system provides middleware-layer mechanisms that allow integrating several heterogeneous services in the networked home independently of their underlying software and hardware technologies. The integration is achieved by bridging service protocols run by various devices [7][8][9][10]. On the other hand, the MIT intelligent room project uses three layered architecture for: sensor data interpretation, abstraction and application. Their project goal is to support users by interpretation of their actions, gestures, and speech [11][12][13]. The Aware Home from Atlanta uses generic service components, named widgets that can communicate over the HTTP and SMTP protocol by exchanging XML style messages [14]. At the University of Illinois, I-Living architecture is being developed for assisted living appliances. This system is centralized and uses a set of gateways for accessing proprietary hardware. A focus of their system lays on robustness through redundancy [15]. HomeLab from Philips and Domolab from Spain perform research in ambient assisted living. Three types of communication architectures are used: message-based, RPC and shared memory. Within these systems, communication or service-interaction aspects of pervasive systems are addressed mainly without specific mapping to appliances [16][17]. MCA framework that offers a distributed efficient shared memory mechanism over TCP/IP was presented in [18]. In this framework arbitrary data structures may be put into Blackboards, which give a powerful tool for data sharing. The Blackboard system is mainly used for larger and unstructured data e.g. images and data containers. A disadvantage of MCA framework approach is the lack of message event support. Events can only be emulated by state changes, on edges or by using proprietary messages.

In this paper, a novel event-based and distributed DATA framework is presented that is based completely on finite-state machine formalism. DATA framework integrates advanced networking technologies, which allow robust, ad-hoc networks to be formed by a broad range of electronic devices, personal mobile devices, and personal mobile robotic unit(s). DATA framework
approach enables efficient, flexible data flow throughout the system (control/configuration data, audio/video data, etc.), and behaviour specification of the system modules in XML style descriptions, following finite-state machine (FSM) formalism. Intelligent ambience architecture, based on DATA framework, can consist of one central server, several module servers and several clients. Java based structure enables uniform, XML style based communication between modules, using encryption mechanism for better security, efficient error recovery mechanism and system modules’ behaviour specification and configuration. Finite-state machine based architecture enables intelligent ambience system to implement efficient event-based mechanism, capable to process and perform several tasks simultaneously. DATA framework supports novel behaviour specification mechanism by using JavaCC [3] and UniMod [5][6] frameworks. In the DATA system architecture, devices are integrated through the so-called module servers in uniform way. In this way, DATA system can communicate with all devices in both ways all the time. In section 2, complex client/server architecture for intelligent ambience is presented. DATA framework and modules for construction of distributed DATA system are described into details in section 3. Proposed system’s behaviour implementation approach is presented in section 4. In section 5 we conclude with simple show case, representing flexibility of the proposed DATA framework.

2 Complex client/server architecture

Model for intelligent ambience system in general consists of several electronic devices, IP cameras, sensors, controllers for driving devices, various personal mobile units (acting as remote data, audio and video capturing and processing units), image/video and audio signal processing servers, database servers and one or more clients. All these devices and modules can be interconnected through complex client/server architecture, as presented in Figure 1. DATA system is suggested to be composed of the following building blocks:

(1) main server: the managing unit that is dedicated for heavy and intensive processing, and management of the whole system.

(2) remote client: these clients are dedicated to users and take care of multimodal human-machine interaction with the intelligent ambience system.

(3) personal mobile unit type 1: the managing units that take care of audio/video capturing, data processing and database management services. They capture, stream and play out audio and video data, but generally they are more limited with processing power than the main server.

(4) personal mobile unit type 2: the managing units that have very limited power sources, low processing power, and are suitable only for less intensive processing tasks. These modules support data and audio/video capturing, however without intensive processing. They should just take care for streaming audio and video data to the main server or other personal mobile units of type 1.

3 DATA framework and modules

Distributed DATA system’s architecture shown in Figure 1, consists of three different functional modules: client, main server and module server. All these modules are suggested to be implemented as finite-state machine engines (FSM) [1] by using DATA framework, presented in Figure 2.
FSM engines, interfacing modules with databases, parsing and generating XML documents, and audio/video capturing and transmission over the internet. Event-based finite-state machine package that can be used for continuous listening and detecting of events at very low processing power consumption is based on UniMod framework. DATA framework is additionally supported with Java CC compiler framework, and JMF 2.1.1e framework [2]. DATA framework can therefore be used for implementation of intelligent ambience system’s modules, we have in mind. In the rest of the paper, we have named clients as DATA clients, main server as DATA server, module servers as DATA module servers and the whole system as intelligent ambience system DATA. FSM based nature of all modules enables administrators of the system flexible and fast generation of several behaviour specifications (understandable to humans) not only for modules, but also for the whole system itself. DATA system’s modules are event-based machines, able to process random events and respond to them in clear and flexible manner. For configuration and upgrading tasks, it is important that modules’ behaviour is not hard coded within the system or within modules. All behaviour specifications are simply stored in the form of XML scenario files, which can be written off-line. Within the system they define not only the behaviour for each DATA module, but also define behaviour of the whole DATA system. DATA client is an event-based finite-state machine, capable to process random events and respond to them in a clear and flexible manner at very low processing power consumption. FSM-based architecture of the DATA client enables generation of several behaviour specifications. The connection established between DATA client and DATA system modules via DATA server, remains active until DATA client reaches the final state of the finite-state engine. Actions and time-scheduling tasks are specified with behaviour specification stored in XML style file. We name these XML files a behaviour specifications for DATA client. Additional audio/video capturing/processing modules can also be implemented in native languages (e.g. speech technology modules can be implemented in C/C++ language). Native modules are integrated into the DATA client through Java native interface mechanism by using JNI package of the DATA framework. Within DATA, client can be ”initiated” for a lot of events, e.g. after some key, or mouse button is pressed, or users’ speech commands are recognized. DATA client is also RTP receiver (accepts audio/video RTP data from the DATA server). This functionality is implemented by using RTP package from DATA module server and JMF 2.1.1e framework [2].

DATA server communicates with other DATA system’s modules by using proprietary XML based protocol. For generation and parsing of request/response packets, packages from the DATA framework are used. DATA server contains PooledThread object. This object is instantiated during the initialization step with beforehand defined number of threads. Each thread can then be used to serve specific DATA client. This kind of implementation approach has been selected because it is faster when managing several threads. For each request coming from specific DATA client, available thread is picked from the pool and used for managing established connection. DATA server is able to exchange data with clients and is also responsible for establishing all further connections with one or more DATA module servers. Namely, DATA server has to connect each DATA client with DATA module server(s), as specified in the desired system behaviour. DATA server has no knowledge how it should behave towards each DATA client. This knowledge is namely not defined in front. Within DATA system, only DATA clients have the role of defining desired behaviour of individual DATA system’s module and consecutively the whole DATA system. Therefore, DATA clients have to transport the desired behaviour specification (XML files) to DATA server and further to all involved DATA module servers. The transport is executed during DATA system configuration stage, performed right after connections between DATA modules are established. This approach is suggested, since each user in general can define different behaviour from the DATA system, and will simultaneously perform different tasks within intelligent ambience. Circumstantially, XML based behaviour specification for the DATA system as whole must vary for each user. Users are able to program DATA systems’ behaviour according to their needs and expectations, and DATA system can adopt different behaviours simultaneously. DATA system is able to run specific system architecture within dedicated threads for each user, and by using specific set of modules’ behaviour specifications for each of them. DATA server has the role of RTP data receiver, transmitter and/or post-processor. These roles can be used in combination or separately (with no relevance to each other). DATA module server architecture is very similar to the architecture of the DATA server and also contains the so-called PooledThread object. Threads are used to host connections with DATA server. When request from the DATA server is accepted, available thread is picked up from the pool and used. When no more threads are available, DATA module server simply rejects all further DATA server requests. In this case, DATA server has to wait until some thread is available again. DATA module server also implements RTP protocol. Its implementation is very similar to implementations already presented for DATA server and DATA client.
4 DATA system’s behavior

Complex client/server architecture of DATA system, as shown in Figure 1, has to be able to handle several events generated by users, individual system’s modules, electronic devices, mobile platforms etc., and perform several actions simultaneously. All these events must be detected and processed robustly, flexibly and efficiently and corresponding reactions have to be performed. Services must be available to any user and to more users simultaneously. Users in general use different profiles for the same service. Therefore, intelligent ambience system’s architecture and corresponding desired modules’ behaviour will be very colourful. Circumstantially, hard-coded intelligent ambience system’s behaviour implementation seems to be too inflexible solution for developers and users (e.g. bug problems, endless testing of the whole system etc.). When the individual system’s module fails, it is important that the system is robust and does not crash. Therefore, error recovery mechanism of the system must be able to recover failed system’s modules into initial state and enable them to continue with their work within the system. The system must also be able to serve different users simultaneously and be able to communicate with each of them in different ways. Within a complex system, a lot of data has to be exchanged and communication flow, based on beforehand mentioned protocols, has to be fast, efficient and fluent. The approach proposed in this paper enables robust, clear and flexible management and configuration of the intelligent ambience system, with simple and minimal additional work needed. How system’s behaviour fulfils our expectations and goals heavily depends on the approach used to specify behaviour of individual modules and the whole system. In general all system’s modules within intelligent ambience perform specific set of actions and process specific set of events in several combinations, depending on the task specified by users. Each task can be described by set of actions and events that have to be performed. Each task demands specific behaviour from the system and its individual modules. Behaviour of individual modules can be described in the form of behaviour specifications. All tasks can be described by a set of specifications, written separately for each module (for e.g. DATA client, DATA server and DATA module server(s)). These behaviour specifications have to follow protocols’ specifications and desired system’s architecture for specific users’ tasks. These specifications also have to contain sequences of actions and events that have to be performed by individual DATA system’s module. All these behaviour specifications can be efficiently and flexibly described in the form of graphs and performed by finite-state machine engines. As already mentioned, all DATA system’s modules are implemented as finite-state machine engines. For implementing DATA system’s modules as finite-state machine engines, FSM package from DATA framework is used. UniMod Java framework and FSM package from DATA framework, define all objects for construction and running of finite-state machine engines in Java language. In this way, each behaviour specification has to be presented in the form of finite-state machines and stored in files in a specific XML data format. When DATA system is used for complex intelligent ambience, there can be numerous XML files that have to be written by the administrator of the system. When XML files are created, they can be run by DATA system’s modules (by finite-state machine engines) in many combinations even simultaneously, without changing the code and without the need for re-compiling and updating of all system’s modules. By using such approach, the issue of endless compilation, deployment of new versions etc., is solved.

5 Application of DATA system

By using distributed DATA system for intelligent ambience any number of users is able to interact with the system and circumstantially with electronic devices and personal mobile units Users can use their own profile and experience specific behaviour from DATA system's modules, even simultaneously, with all other users. User actions are interpreted by DATA client as events that trigger corresponding finite-state machine engine. DATA client responds with actions, as it is specified in behaviour specifications defined by user. There are events that are generated within DATA system itself and have to be notified, processed and handled properly by all involved DATA system’s modules.

All DATA system’s modules can communicate via Wi-Fi/LAN and power-line networks. DATA server takes care of database management, processing and machine learning. IP or USB cameras and Java JMF framework integrated in the DATA system enable a real-time audio/video streaming within intelligent ambience and several audio/video processing tasks as a support to multimodal human-machine interaction between users and intelligent ambience. DATA server and DATA module servers can serve several simultaneous client connections, where number of connections is limited by processing power of hosted PC platforms. The show case, shown in Figure 4, describes DATA architecture, where user interacts with DATA client on a remote platform. DATA server represents central communication point, interfaced with one DATA client and two DATA module servers running on personal mobile platforms 1 and 2. This show case is a simple
example of general remote management of intelligent ambience when using DATA system. User’s location is not fixed and the only condition is that the platform for running DATA client module is available.

Figure 4: Show case: user interacting with DATA system from the remote client.

Figure 5: Show case: communication diagram flow.

Users can operate the entire distributed DATA system from a location that is not even necessarily within system’s LAN. Communication between user and the system, as shown in Figure 5, starts, when DATA client connects to the DATA server. **MainFrame communication section** presents communication possibilities between DATA server and remote DATA client. User performs actions by using available human-machine interface hosted by remote DATA client. DATA client forms suitable XML request packets according to detected events and used underlying protocols. DATA server takes care for parsing the content of accepted packet and performs specified actions. During interaction, user request can result in visual presentation of accepted XML packet stream from DATA server, or even audio/video stream. User can control/monitor one or even several electronic devices connected to DATA server, capture and preview audio/video data captured by one or several capturing devices that are connected to DATA server or just monitor status of the ambience and/or electronic devices within. Users can communicate also with DATA module servers, connected to the DATA server. The communication flow in Figure 5 presents two possibilities: **simultaneous control** and **simultaneous picture capturing task**. Since DATA client is connected to DATA module servers via DATA server, user’s XML request packets are first posted in the form of XML packet to the DATA server. DATA server then processes received XML packets and transmit them to the specified DATA module servers hosted by personal mobile platforms of type 1 and 2. **Simultaneous picture capturing task** shows that simultaneous requests from the user are not necessarily broadcasted/multicasted within the DATA system. When user wants to take pictures from the cameras located on main frame platform and personal mobile platform 2, DATA client has to post XML picture request packet to the DATA server that has to perform this task for specified cameras. DATA server has to return the picture, taken by a camera located on the main frame platform and also redirects the picture taken by a camera on personal mobile unit 2. Pictures can be transmitted simply in the form of TCP/IP packet stream.

5 Conclusion and future research

DATA framework has been proposed to be used for intelligent ambience, since it is very suitable for implementation of complex environments with many users interacting with the system in several ways, even simultaneously. Within DATA framework development, powerful Java features have been exploited (e.g. debugging, multithreading, exception handling capabilities, memory management etc.) as well as many powerful Java based frameworks (SAX support for analyzing and parsing XML files, JMF 2.1.1e framework for implementation of RTP protocol (audio/video transmission), UniMod for implementing finite-state engines, JavaCC for Java parser development etc.). By using presented approach for DATA system, users are able to specify several behaviours for DATA system and intelligent ambience itself, by writing suitable sets of behaviour specifications in a form of XML files. Behaviour of all DATA system’s modules can be defined in flexible and efficient way that is very importantly also readable and understandable to humans, and without any changes in the code needed. DATA system enables construction of several different architectures for intelligent ambience and with different behaviour from users’ point of view that can be
performed even simultaneously. DATA system is event-based system, enabling detection and processing of several events that can be triggered by users and the system itself. DATA system’s distributed nature and presented approach to behaviour implementation, enables further flexible and simple integration of new devices, equipment and personal mobile units, which is very important because of very dynamic nature of intelligent ambiences.

Acknowledgements
Operation is part financed by the European Union, European Social Fund.

References: