DOWNLOAD PDF SOFTWARE AGENT-BASED APPLICATIONS, PLATFORMS, AND DEVELOPMENT KITS

Chapter 1 : Software Agent-Based Applications, Platforms and Development Kits | Ebook | Ellibs Ebooksto

Software agent based applications, platforms and, software agent based applications, platforms and development kits # whitestein series in software agent technology another focus lies on agent.

Provisional Patent Application Ser. Accordingly, this non-provisional patent application claims priority to U. In another aspect, one or more embodiments of the invention relate to a method for using servicing requests, comprising: In yet another aspect, one or more embodiments of the invention relate to a non-transitory computer readable medium comprising computer readable program code for: In the following detailed description of embodiments of the invention, numerous specific details are set forth in order to provide a more thorough understanding of the invention. However, it will be apparent to one of ordinary skill in the art that the invention may be practiced without these specific details. In other instances, well-known features have not been described in detail to avoid unnecessarily complicating the description. In the following description of FIGS. For brevity, descriptions of these components will not be repeated with regard to each figure. Thus, each and every embodiment of the components of each figure is incorporated by reference and assumed to be optionally present within every other figure having one or more like-named components. Additionally, in accordance with various embodiments of the invention, any description of the components of a figure is to be interpreted as an optional embodiment which may be implemented in addition to, in conjunction with, or in place of the embodiments described with regard to a corresponding like-named component in any other figure. In general, one or more embodiments of the invention relate to a SDK platform. The SDK platform includes: Further, in one or more embodiments of the invention, the SDK platform provides mechanisms that enable sharing data amongst other applications that implement SDK modules supported by the SDK platform. In one or more embodiments of the invention, the SDK platform provides a standardized way of communicating between applications such that a developer of one application can call a targeted action inside another application in a consistent manner. The system includes a computing system see e. In one embodiment of the invention, the backend services correspond to computing services that are remotely accessible to local SDK platform see e. The backend services may be implemented on one or more computing systems. The 3rd party services correspond to all other remote computing services excluding those provided by the backend services , where such 3rd party services are implemented using one or more computing systems described below in FIG. In one embodiment of the invention, the computing system may directly communicate only with the backend service. In such cases, if an application on the computing system is attempting to communicate with a 3rd party service, then such requests are sent to the backend service, which subsequently sends the requests to the 3rd party service. In other embodiments of the invention, the computing system may communicate directly with the 3rd services. In other embodiments of the invention, the computing system may implement both of the aforementioned embodiments. The SDK platform backend may provide SDK marketplace services, such that a software application developer may purchase or otherwise obtain the non-core SDK module instance and incorporate the non-core SDK module instance into a software application. A SDK module e. The SDK module may be the implementation of one or more application programming interfaces APIs in the form of libraries. In one or more embodiments of the invention, the SDK platform backend may also provide an application marketplace for software applications. An application market is a virtual network space, supported by hardware and software that connects consumers of software applications to developers of software applications and facilitates the exchange of software applications. In at least some embodiments of the invention, the application market is a public market that is accessible via the Internet. In one embodiment of the invention, the SDK platform backend includes a data repository not shown. The data repository includes functionality to store, for example, system configurations data and user data. Other data may be stored in the repository without departing from the invention. The system configuration data for a particular system may include identifying all a non-core SDK module instances see FIG. The system configuration data may include

an operating system identifier, a device manufacturer name and model name, a web browser name, and any other identifying information of a system configuration. The system configuration data for a particular system configuration may include additional information, such as the functionality provided by the system configuration, components of the system configuration, known defects of the particular system configuration, other such information, or a combination thereof. In one or more embodiments of the invention, user data corresponds to data for a particular user e. User data may include a user profile and system configuration identifier. A user profile includes information about a user. For example, the user profile may include administrative information e. The system configuration identifier is a reference to the corresponding system configurations data for a computing system that the user is using. Rather having a separate system configurations data and user data, the system configurations data may be a part of the user data. The use data may also include some or all of the following information: Further, the data in the user profile may be stored in an anonymized form. In one embodiment of the invention, an application corresponds to executable code that may be executing by the computing system Each non-core SDK module provides a particular type of functionality within the application in which it is implemented. The invention is not limited to the system architecture shown in FIG. In one or more embodiments of the invention, the local SDK platform enables applications A, B executing on the same computing system to share data. Further, the local SDK platform enables applications A, B on the computing system to invoke the functionality of other applications A, B on the computing system In one or more embodiments of the invention, the local SDK platform includes an application discovery module ADM that includes functionality to discovery which of the applications A, B on the computing system have one or more non-core SDK module instances A, B. Further, the ADM may include functionality to query each such application via the applications SDK platform interface A, B in order to determine which specific non-core modules A, B are present in each of the applications A, B. The local SDK platform may then use the information in the manner described herein. Further, the local SDK platform may provide information that it has gathered e. Applications that include non-code SDK modules may: Though not shown in FIG. In one embodiment of the invention, a deep link is a uniform resource identifier URI that links to a specific location within a target application. Further, the deep link may include other information such as the source of the deep link i. The following is an exemplary deep link: In this example, the target application is MyApp, the source application is App2 i. While the various steps in the flowchart are presented and described sequentially, one of ordinary skill will appreciate that some or all of the steps may be executed in different orders, may be combined or omitted, and some or all of the steps may be executed in parallel. The request may correspond to a request where the corresponding response includes a deep link. For example, the request may be for a native advertisement where the responding native advertisement includes a deep link to another application executing on the computing system. At the time of the request, the requesting application may not necessarily require a response with a deep link, e. In one or more embodiments of the invention, the request may be an explicit request for a deep link. However, if the application does not natively support the purchase action, the application can send a request to the local SDK platform in order to obtain a deep link for an another application on the computing system that can handle the purchase action or to obtain a deep link to an application that must be downloaded onto the computing system and then subsequently launched using the information in the deep link. These examples are not intended to limit the scope of the invention. In step , the local SDK platform determines whether another application B on the computing system can service the request. This information may be used to service the request. For example, if the request is for a purchase action, then the local SDK platform may include functionality to determine if any of the applications that include non-core SDK modules can service the request, i. In another example, if the request is for a native advertisement, then the request may need to be serviced by the backend service or a 3rd party service For example, the request may be sent to a backend service e. In another example, the request may be sent to the backend service, which subsequently sends the request to a 3rd party service e. The 3rd party advertisement exchange may then provide the native advertisement to the backend service, which may modify the native

advertisement to include a deep link e. The modified native advertisement may then be returned to the computing system Continuing with the discussion of FIG. In step , when the request cannot be serviced by another application B that is currently present on the computing system , then a request is sent to the backend service or a 3rd party service see example discussed above. In step , the backend service or a 3rd party service provides a response that may include a deep link. Consider the following first example, if the request is for a native advertisement, the backend service or a 3rd party service may select the native advertisement using any known selection method and the incorporate into the native advertisement a deep link. In this example, the deep link may correspond to an application that is present on the computing system and may specify that when the application launches what screen is to be initially shown to the user via the user interface on the computing system In the first example, the backend service or a 3rd party service includes information which may be obtained from the local SDK platform on which applications are present on the computing system and the functionality of each of those applications. Consider the following second example, if the request is for a native advertisement, the backend service or a 3rd party service may select the native advertisement using any known selection method and then incorporate into the native advertisement a deep link. In this example, the deep link may correspond to an application that is not present on the computing system and may also specify that when the application launches what screen is to be initially shown to the user via the user interface on the computing system. In the second example, the backend service or a 3rd party service includes information on what other applications are available for download on the computing system and the functionality of each of those applications. Consider the following third example, if the request is for a native advertisement, the backend service or a 3rd party service may select the native advertisement using any known selection method and then incorporate into the native advertisement a link to a website that supports, e. This may occur in scenarios in which there are no applications on the computing system or available for download on the computing system that can support the functionality required. In step , if the method reaches step via step , then the response which was generated in step is sent to the non-core SDK module A or, for example, the local SDK platform instance A in the first application A via the SDK platform interface A, B with or without modification ; alternatively, if the method reaches step via step , then the local SDK platform first generates the response the deep link, where the deep link may correspond to an application that is present on the computing system and may specify that when the application is launched what screen to initially show the user via the user interface on the computing system In step , in the scenario in which the second application B is already loaded on the computing system , the selecting of the deep link in the first application A launches the second application B to a specific screen of the application B , where the specific screen is specified in the deep link. In the scenario in which the second application B is not already loaded on the computing system , the selecting of the deep link in the first application A triggers the downloading of the second application B onto the computing system Once downloaded, the second application B is launched to a specific screen of the application B , where the specific screen is specified in the deep link. Finally, in the scenario in which the response includes a link to a website, the selecting of the link in the first application A launches a web browser application that displays the particular webpage as specified in the link. The system shown in FIG. Further, as shown in FIG. The invention is not limited to the system architecture shown in FIGS. The following describes various examples in accordance with one more embodiments of the invention. The invention is not limited by the following examples. In one or more embodiments of the invention, the local SDK platform provides generalized action identifiers which developers can elect to support in their application. The registration process may be managed by the application discovery module in the local SDK platform. Through the monetization SDK module i. The local SDK platform is also able to determine that Application B is installed, so Application A presents an advertisement for a San Francisco Giants hat available in Application B, where the local SDK platform generates a deep link that is included in the advertisement. The appropriate screen may be specified in the aforementioned deep link or may be determined using any other known mechanism.

Chapter 2 : Software Agent-Based Applications, Platforms and Development Kits - çˆ±ç¨‹åº•ç½‘

Software Agent-Based Applications, Platforms and Development Kits. Editors: Unland, Rainer, Klusch, Matthias, Calisti, Monique (Eds.).

 Abstract Application development platform is the most important environment in IT industry. There are a variety of platforms. Although the native development enables application to optimize, various languages and software development kits need to be acquired according to the device. The coexistence of smart devices and platforms has rendered the native development approach time and cost consuming. Cross-platform development emerged as a response to these issues. These platforms generate applications for multiple devices based on web languages. Nevertheless, development requires additional implementation based on a native language because of the coverage and functions of supported application programming interfaces APIs. Wearable devices have recently attracted considerable attention. These devices only support Bluetooth-based interdevice communication, thereby making communication and device control impossible beyond a certain range. We confirmed the efficiency of the proposed platform through experiments and a qualitative assessment of its implementation. Furthermore, wearable devices have attracted considerable attention as the next generation of mobile technology that will replace smartphones. Furthermore, even sports equipment companies such as Nike and Adidas have lately begun introducing innovative products and services in the wearable device market [2]. The Internet of Things IoT is a recent technology for collecting data and transferring data through sensors and adding a communication function to every object. In December , subscribers of smartphones exceeded Along with the popularity of smartphones and wearable devices, the market for mobile applications is steadily growing as well [5]. Due to the emergence of smartphones and the existence of multiple platforms, developers in the stage of application development have to build platform-specific environments, use multiple programming languages, and learn the relevant supporting application programming interfaces APIs. These constraints lead to wasted time and effort and increase the cost of application development. In order to solve such problems, mobile programming is in the process of standardization. Cross-platform development frameworks, such as Cordova [6] and Titanium [7], have garnered considerable attention as a solution. The Association for Computing Machinery ACM , the Institute of Electrical and Electronics Engineers IEEE , and related organizations are publishing an increasing number of studies on platform research that analyze the characteristics of cross-platform development frameworks. This suggests that application development on a single, unified platform will become possible in the near future [8 , 9]. Research Challenges and Contributions. Its greatest benefit is that it provides an application with a single source that is feasible on multiple mobile platforms, which increases development productivity. However, the range of APIs in Cordova and Titanium is not sufficiently wide, especially with regard to supporting wearable devices and IoT devices, since they are still in their developmental stages. Developers are thus inevitably required to learn the native language supported by the platform of the relevant device and initiate follow-up development because cross-platform development alone is insufficient for application development. The potential benefits of our proposed platform are as follows. First, it eases application development by supporting integrated development environments that supply essential APIs or facilitate device development. Second, it enables wearable devices that use Bluetooth to communicate with the outside by supporting IP communication based on WebSocket. The outline of the remainder of this paper is as follows. Section 2 examines the benefits and drawbacks of the existing development platforms. In Section 3 , we introduce NetApp-Agent, our proposed integrated development platform, together with its structure and features. Section 4 presents the results of our experiments involving NetApp-Agent as well as a qualitative assessment of its implementation. Finally, we offer our conclusions and recommendations for further research

in Section 5. Review of Existing Approaches and Issues Approaches to mobile application development can be divided into three major categories: Hybrid apps assume the form of native apps, but all or part of its internal configuration is developed in a web app environment [11 â€” 13]. In this section, we examine the development of native and hybrid apps together with the features and challenges of a network application development system that provides an IoT development platform and cloud service. Native Development Platform A native development platform involves developing applications on the platform of each device, such as the iPhone, the Android phone, and the Windows Phone, which operate in machine language code. It ensures optimized application performance. However, it has a few major disadvantages: We examine the native development environments for both the Sony and the Pebble SmartWatch, which are representative of wearable devices. The procedure for developing applications for the Sony SmartWatch is as follows [14]. A development environment must first be constructed. The Android development environment is then constructed [15]. Furthermore, the system structure and the API supporting the development need to be learned. The system architecture of Sony SmartWatch is shown in Figure 1 and can be divided into three major components: Smart Extension, Host Application, and Accessory. Smart Extension refers to the application to perform in wearable device. The Host Application is installed in smartphones and connected to the wearable device using Bluetooth technology. System architecture of Sony SmartWatch. Table 1 lists various supporting APIs. Control API in particular is crucial to controlling the display or the light-emitting diode LED of the device and to processing key events or touching events that require close attention. The Sensor API transmits data from the accelerometer and the illumination sensor of the device to the Smart Extension application. The Widget API affords content preview. The lengthy and complex procedure described above concludes the preparation for the development of an operating application for Sony SmartWatch. However, information regarding implemented classes, functions, and variables still needs to be checked, and extensive research needs to be conducted on the API implementation code and the sample code, along with code analysis, by using API reference documents for application development. Pebble supports different types of languages, such as JavaScript and Objective-C, where the latter is primarily used for application development. Moreover, a Pebble SDK is provided for development. The procedure for developing applications for Pebble SmartWatch can be divided into two parts. First, the application development environment must be constructed. We assume the construction of a development environment conducted on Mac OS X. Development environment construction is completed after building in Python library because Pebble SDK is based on Python. Pebble is compatible with smartphones that use Android and iOS platforms. Therefore, the development environment of a smartphone application should be built and an SDK called PebbleKit should then be installed to create an application that is in sync with the smartphone [17]. Second, a development support API must be learned. The scope of the supporting SDK is presented in Table 2. When developing a Pebble application, two issues need careful scrutiny in addition to the construction of the development environment and the examination of the supporting APIs. First, the Pebble SmartWatch does not support Korean characters. Hence, expressions in Korean need to be considered when developing an application that sends text messages or notifications for Social Network Services SNS to Pebble devices in Korean. Second, the image format of Pebble is problematic. PBI represents each pixel using one bit that contains image information in the header file. Thus, a developer must create a tool for image conversion to enable images in Pebble Watch. The relevant Pebble Watch application is then installed on a computer terminal by the way of inputting build and install command. Pebble Watch should be connected to a smartphone through Bluetooth for application installation, following which the computer and the smartphone that have already progressed in development should be connected to the same Wi-Fi network. Therefore, developers should pay particular attention to network configuration when installing the application. Thus, manufacturers produce smartphones on a variety of platforms. The coexistence of different platforms has led developers to establish suitable development environments for each platform when creating applications and learn the relevant development languages, the SDK, and the API. Table 3 shows diverse development

environments according to types of wearable device. As shown in the table, a developer needs to learn ten programming languages and seven APIs for an application adaptable to four devices. With the growing trend of wearable devices, more and more devices are expected to be introduced. Moreover, time spent on application development will increase in proportion to the number of devices. To solve these kinds of problems, an integrated development platform is required. Development environments for each wearable device. Cross-Platform Mobile Development Framework Table 4 shows that the development of mobile applications can be divided into three types: Features of each mobile application development method. As shown in Section 2. However, they have a few constraints given that they require building a different development environment for each platform and that the developer needs to learn the relevant development language and the SDK. The advantage of web apps is that they can attract and train developers, since learning a development language is relatively easy. On the other hand, difficulty in hardware control, slow speed of applications, and vulnerability on networks are the major weaknesses of web apps. Applications developed by hybrid apps assume the form of native apps, but all or part of their internal configuration is developed in web app. Hybrid apps improve development productivity because they can be operated in various mobile platform using a single source. Hardware control is also possible. Cross-platform is applied as a development tool for hybrid apps. In this section, we discuss Cordova and Titanium, two typical instances of cross-platform development. Cordova Cordova is an open-source framework that enables hybrid application development. Following the takeover, it reinforced the open-source policy with the development of the Apache license, and then he changed the name of the application to Cordova from version 1. Additional functions for Cordova are developed as plug-ins that are shared in open-source communities. The completed codes are then packaged through the Cordova library. The application in packaging is distributed to the device in which web-kit provided browser is equipped.

Chapter 3 : Developer Tools for Windows - Free downloads and reviews - CNET calendrierdelascience.com

Additional info for Software Agent-Based Applications, Platforms and Development Kits Example text The ConversationID should be viewed as a 'reference number'.

The usability criteria group is also easily understood. It is obvious that familiar mechanisms need less or no effort to learn and use while unfamiliar mechanisms could be time and effort consuming. Scalability refers to a potential critical issue; whether the platform can handle varying problem sizes or not. Standard compatibilities, next, are some of the most important features in an agent platform. Performance is about how efficient is the platform with respect to space and time operations. In this context, we studied how fast the agent communication is and the running speed of the platform itself. Values ranges from average meaning tolerable but not really fast to high meaning excellent platform performance. Stability is about how stable is the platform in case of long term execution whereas robustness is about how tolerant it is in case of breakdowns. In this study, it is a value representing a percentage based on all available data of platform crashes, ranging again from average to high many crashes. Next, programming languages and operating systems indicate the languages and the systems that are used in or by the platform, respectively. Finally, security management is an important aspect in multi-agent systems. The classic case of end-to-end security refers to those protocols and mechanisms that protect access to the platform while fairness refers to those special mechanisms that guarantee equal treatment. Additionally, platform security requires more mechanisms in order to ensure platform security and smooth operation. In this study, platform secure ranges from weak to good, high and strong in this order indicating the amount of security mechanisms that are available in the platform. Trust models help agents to decide who to trust, encouraging trustworthy behavior and deterring dishonest participation by providing the mean through which reputation and ultimately trust among the community can be quantified Resnick et al. In other words, in is not only important to have a secure agent platform but it is required the community acting in that platform to be able to use trust mechanisms. Hence, we do not include trust management as a separate criterion, although it is important, since it is not supported by the platforms yet except EMERALD. Web is moving towards a true global village, connecting people and knowledge. Eventually, the idea of building a network of content stored on the web making it possible for machines to understand meaning of data and to satisfy requests from people by using it, will came true to its full potential. The semantic wave embraces four stages of internet growth Mills The first, Web of Shared Information Web 1. The third, Semantic Web Web 3. Hence, the question is if agent platforms are or will be able to deal with that. In other words, the question is if agents build in these platforms will be able to traverse the web and integrate the available knowledge. So far, there is no survey, to the best of our knowledge, that takes into account this issue. To this end, we used all the available data to discover which platforms support technologies, semantic web technologies in particular, that can be used for this purpose. As a result, we present an additional classification subsection, where agent platforms are classified according to the degree of their support in semantic web technologies. Agent Platforms Overview 3. Hence, this section presents the platforms that are still available even if their development progress is stopped. An interesting question raises here; why platforms that are no longer developed should be included? The answer is that there are some important and popular agent platforms that are still used by the community though their developers have stopped their maintenance. However, these cases are just a minority. Almost all the presented platforms are active with stable releases. In this context, we have chosen twenty four cases including pure agent platforms, agent frameworks and agent-based simulators because all of them are or will be able to act like agent platforms. Hence, next, each of them is presented in alphabetic order while pivot tables complete the overview at the end of the section. The framework is broadly split into two parts: Currently, Agent Factory is used in a number of projects including mobile computing, robotics and other, e. It is a KQML-compliant integrated tool suite for constructing intelligent software agents, allowing software developers with no background in

intelligent systems or intelligent agent technologies to quickly and easily build intelligent agent-based applications. KQML is a language and protocol for communication among software agents and knowledge-based systems Finin et al. AgentBuilder is quite simple and easy to use even for unfamiliar users is its main advantage. AgentBuilder is an agent platform based on the notions of mental states, which comply with the agent language Agent-0 proposed by Shoham. Within AgentScape, agents are active entities that reside within locations, communicate with each other and access services. The AgentScape approach to management is targeted to scalability and autonomicity. The above allows AgentScape to be a promising agent platform for studies including large-scale distribution and heterogeneity. The platform provides functions for the residing agents, such as communication infrastructure, store, directory services, migration function, deploy service, etc. This interoperability is not necessary when developing closed systems, where no communication outside these systems is required e. The flexibility of its modeling language enables the user to capture the complexity and heterogeneity of business, economic and social systems to any desired level of detail. Additionally, the object-oriented model design paradigm supported by AnyLogic provides for modular, hierarchical, and incremental construction of large models. It is probably one the best solutions for simulation if there is no extensive coding background to start with. Cormas pre-defined entities are Smalltalk generic classes from which, by specialization and refining, users can create specific entities for their own model. It facilitates the construction of agent-based models and the design, monitoring and analyzing of agent-based simulation scenarios. Cormas was primarily oriented towards the representation of interactions between stakeholders about the use of natural renewable resources. Cormas is very popular and it provides some interesting features for real life applications. The platform is not FIPA-compliant. It facilitates the development of agent based applications that are complex, large scale and distributed. CybelePro is the commercial release of Intelligent Automation Inc. Cybele agent infrastructure has been used extensively by the government, industry and academia for applications such as military logistics, modeling, simulation and control of air and ground transportation, communication networks and a development of open systems. The advantage is that every agent can exchange its position justification arguments with any other agent, without the need for all agents to conform to the same kind of rule paradigm or logic. Additionally, EMERALD is the only agent platform that supports trust and reputation mechanisms in order to support trustworthiness and efficient decision making in the multi-agent system. It has been used so far in studying how agents act on behalf of their users in cases such as trading. It provides capabilities concerning the tight combination of 3D visualization, GIS data management, and multi-level modeling. G A M A supports capabilities for building large models written in the GAML agent-oriented language, with an optional graphical modeling tool to support rapid design and prototyping. INGENIAS addresses roundtrip engineering issues as well, by a concrete folder structure and a code-to-specification information migration tool. It is built on a sound logical foundation: BDI is an intuitive and powerful abstraction that allows developers to manage the complexity of the problem. It is entirely written in Java. JACK is the leading edge commercial BDI-agent toolkit that uses an intuitive language that extends the Java programming language with certain agent specific keywords. It is the first platform with support for capabilities and generic team structures, but does not yet support the FIPA-specifications. It simplifies the implementation of multi-agent systems through a middle-ware that claims to comply with the FIPA specifications. The agent platform can be distributed across machines which do not even need to share the same OS and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another one, as and when required. JADE is completely implemented in the Java language and the minimal system requirement is version 1. It is a free, open source and stable software distributed by Telecom Italia, the copyright holder. It allows for programming intelligent software agents in XML and Java. Jadex has been put into practice in the context of several research, teaching, and industrial application scenarios some of which are described in its website. It has been used to build applications in different domains such as simulation, scheduling, and mobile computing. For example, Jadex was used to develop a multi-agent application for negotiation of treatment

schedules in hospitals Braubach et al. In the latest version, the programming model of Jadex is based on the notion of active components that are conceptually based on SCA service component architecture. This allows for designing an application as hierarchical decomposition of components interacting via services and thus helps making complexity controllable. Active components extend SCA in several directions as it is intended to work in concurrent and dynamic distributed systems. The architecture allows any modeling and simulation technique to be integrated into the framework via plug-ins. Moreover, it provides a solid foundation of abstractions, algorithms, workflows and tools, focusing on efficiency. Yet, it does not enforce any reuse of its parts but allows its users to choose which functionality they want to apply. Additionally, JAMES II allows full control over experiment types and parameters, such as parameter scans, optimization, computation end policies and number of replications. JAS is a simulation toolkit specifically designed for agent-based simulation modeling. The core of the JAS toolkit is its simulation engine based on the standard discrete event simulation paradigm, which allows time to be managed with high precision and from a multi-scale perspective. Many features of JAS are based on open source third party libraries. The core of the JAS toolkit is represented by the simulation engine. It is based on the standard discrete-event simulation paradigm, which allows managing the time with high precision and multi-scale perspective. Thanks to its discrete-event engine, JAS represents a good compromise in simulating both discrete and continues agent-based models. This makes JAS a generic discrete-event simulation toolkit, also useful to realize process workflow simulation models. It implements the operational semantics of that language, and provides a platform for the development of multi-agent systems, with many user-customizable features. Using SACI, a multi-agent system can be distributed over a network effortlessly. Some of the features available in Jason are: The framework supports the design, implementation, and deployment of software agent systems. The entire software development process, from conception to deployment of full software systems, is supported by JIAC. It also allows for the possibility of reusing applications and services, and even modifying them during runtime. The focal points of JIAC are distribution, scalability, adaptability and autonomy. MaDKit agents play roles in groups and thus create artificial societies.

DOWNLOAD PDF SOFTWARE AGENT-BASED APPLICATIONS, PLATFORMS, AND DEVELOPMENT KITS

Chapter 4 : SDK for Device Software Development

While other agent based platforms (Unland et al.,), such as FIPA, focus on communication standards, ICARO-T focus on providing high level software components for easy development of complex.

Not all tools are supported on earlier operating systems Hardware requirements 1. To install through Windows Update, make sure you install the latest recommended updates and patches from Microsoft Update before you install the Windows SDK. The -mof parameter is deprecated. This parameter instructs mc. The only remaining difference between -km and -um is that the EventWrite[EventName] macros generated with -km have an Activity ID parameter while the EventWrite[EventName] macros generated with -um do not. The generated header now supports several customization macros. The manifest supports new attributes. If you use provider traits in the manifest, the EventRegister[ProviderName] macro automatically registers them. MC now report an error if a localized message file is missing a string. Previously MC would silently generate a corrupt message resource. MC can now generate Unicode utf-8 or utf output with the -cp utf-8 or -cp utf parameters. For more information, see the documentation. New debugger data model API A new object-oriented debugger data model interface to support debugger automation is now available by using the dbgmodel. Documentation will be available at: If you want the latest WPR to behave the same way as with previous versions, include the following attribute as part of the TraceMergeProperties element: To ensure proper use of WPA, install the latest version of. NET can be installed from https: You can browse the code on GitHub , clone a personal copy of the repository from Git, or download a zipped archive of all the samples. We welcome feedback, so feel free to open an issue within the repository if you have a problem or question. These samples are designed to run on desktop, mobile, and future devices that support the Universal Windows Platform UWP. An adapative app "lights up" with new features wherever the devices and Windows version supports them, but otherwise offers only the functionality available on the detected platform version. For implementation details, see Dynamically detecting features with API contracts 10 by For the latest release notes or issues with tools, see the Windows Developer Forum.

Chapter 5 : Embedded Systems Developer Kits & Modules | NVIDIA Jetson

"This book introduces major agent platforms, frameworks, systems, tools, and applications. Each system is described by their developers in sufficient detail so that the reader can get a good understanding of the architecture, functionality, and application areas of the system.

Chapter 6 : Rainer Unland (Author of Software Agent-Based Applications, Platforms and Development Kits

Software Agent-Based Applications, Platforms and Development Kits (Whitestein Series in Software Agent Technologies and.

Chapter 7 : Windows 10 SDK - Windows app development

This book introduces major agent platforms, frameworks, systems, tools, and applications. Each system is described by their developers in sufficient detail so that the reader can get a good understanding of the architecture, functionality, and application areas of the system.

Chapter 8 : Best IoT Development Kits | Overview Guide

Software Agent-Based Applications, Platforms and Development Kits Introducing major agent platforms, frameworks,

systems, tools, and applications, this text provides readers with a good understanding of the architecture, functionality, and application areas of the system.

Chapter 9 : Rainer Unland (Author of Software Agent-Based Applications, Platforms and Development Kits

A-globe: Agent Development Platform with Inaccessibility and Mobility Support David Å iÅ¡lÃ¡k, Martin RehÃ¡k, Michal PÄ›chouÄ•ek, Milan Rollo, DuÅ¡an PavlÃ-Ä•ek Pages