International Journal of Computer Integrated Manufacturing
Vol. 22, No. 7, July 2009, 671–685

A conceptual framework for computer-aided ubiquitous system engineering: architecture and prototype

Suho Jeong[a], Seung Min Hur[a,b] and Suk-Hwan Suh[b]*

[a]*Department of Industrial and Management Engineering, POSTECH, Pohang, South Korea;* [b]*Center for Ubiquitous Manufacturing, POSTECH, Pohang, South Korea*

(*Received 20 April 2009; final version received 10 May 2009*)

Ubiquitous computing technology has been rapidly advanced and applied to diverse domains. This paper proposes a new approach, called CAUSE (Computer-Aided Ubiquitous System Engineering), for implementing ubiquitous manufacturing systems. This method has great potential compared with the conventional ones that rely on physical implementation and human experience through trial-and-error correction. Specifically, this paper develops a framework for the software supporting the ubiquitous system engineering based on requirement analysis, functional details, and implementation issues including enabling technology. The usefulness of the proposed method is shown by developing a prototype system for a ubiquitous manufacturing system with radio frequency identification (RFID).

Keywords: ubiquitous computing technology; UbiDMR; ubiquitous convergence technology; ubiquitous system engineering; ubiquitous manufacturing system design; RFID application

1. Introduction

Since the concept of ubiquitous computing was proposed in early 1980s, the hardware/software technology for realising the ubiquitous system has been greatly advanced. In a ubiquitous system, required information is collected from a variety of sensors and distributed via communication networks anytime and anywhere. There have been a number of attempts to apply such ubiquitous technology to diverse domains such as home, hospital, factory, traffic system, surveillance/monitoring system, battlefield, logistics, etc. As far as applied technology is concerned, almost all the applications use RFID (Radio Frequency Identification) technology for identification of the PPR (Product, Process, and Resource) (Huang *et al*. 2007, Huang *et al*. 2008a, Huang *et al*. 2008b).

Our research group proposed a new product lifecycle engineering paradigm called UbiDMR (Design, Manufacture, and Recycling via Ubiquitous Computing Technology), whose main issue is the utilisation of the entire product lifecycle information via ubiquitous computing technology for product design, manufacturing, and recycling (Suh *et al.* 2008). Realising such a paradigm requires integration of several domain technologies, such as manufacturing technology, information technology, and ubiquitous computing technology. In our publication, we proposed a technology roadmap, as shown in Figure 1. The enabling technology is largely composed of: 1) horizontal integration technology to build up an integration highway, called the UPLI (Ubiquitous Product Lifecycle Information highway), for seamless information exchange among stakeholders associated with the product life cycle, 2) vertical integration technology dealing with sensors, devices, and communication used to extract information from the PPR in the physical world, and 3) system engineering technology to establish the ubiquitous system through the horizontal and vertical technology, including design, simulation, analysis, and evaluation of the ubiquitous system on the computer.

Following the UbiDMR paradigm, innovative services can be derived in the line of product lifecycle engineering spectrum. Our previous publications have developed a conceptual framework and architecture for each service, and showed the following u-Services were effective in usage scenarios: u-Factory (Shin *et al.* 2007), u-PRMS (Um *et al.* 2008), and UPLS (Lee and Suh 2008). In order to realise a new u-Service and take full benefit of the UbiDMR paradigm, system engineering is a key technology. System engineering technology includes design, analysis, and optimisation of a specific ubiquitous system as characterised by the product type and environment where the product is manufactured, used, and recycled. This paper is concerned with system engineering for the ubiquitous

*Corresponding author. Email: shs@postech.ac.kr

ISSN 0951-192X print/ISSN 1362-3052 online
© 2009 Taylor & Francis
DOI: 10.1080/09511920903030387
http://www.informaworld.com

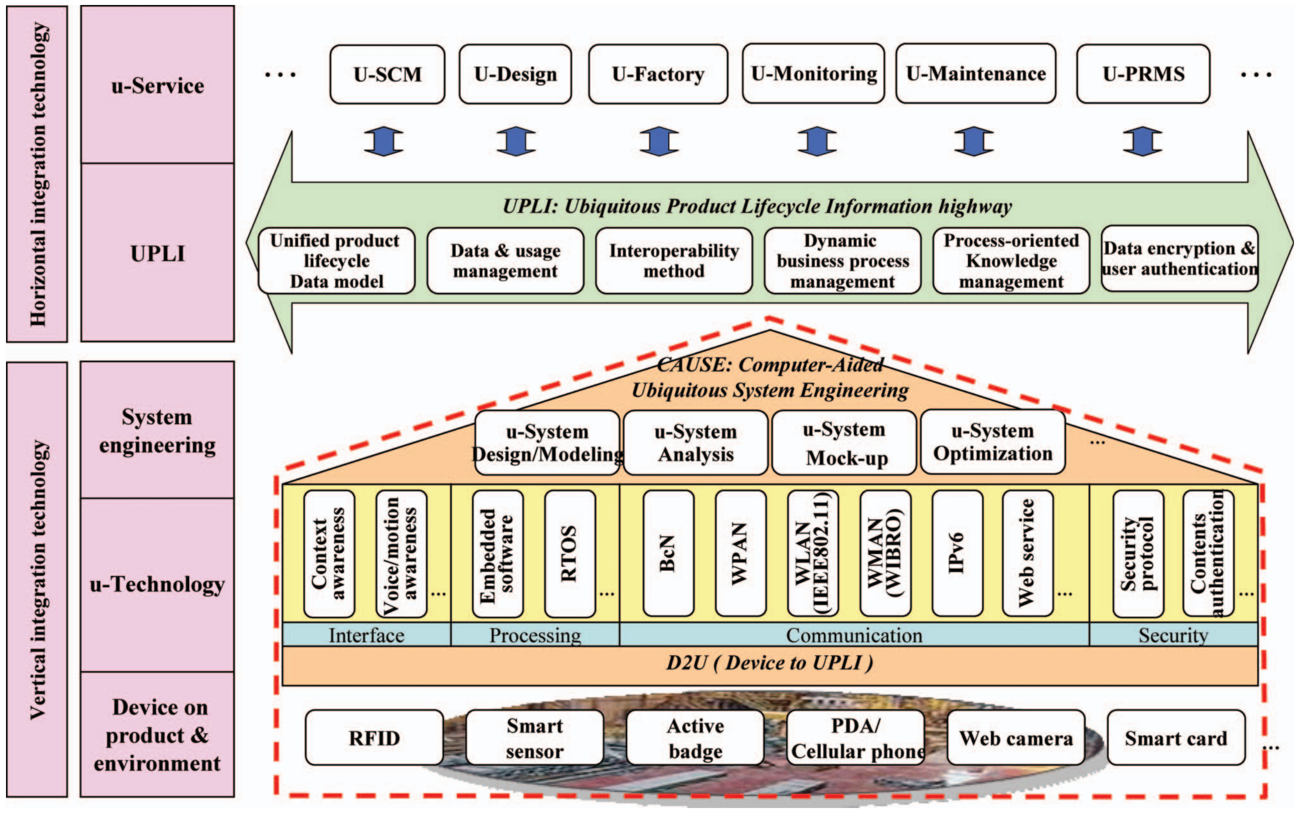


Figure 1. Enabling technology of UbiDMR (Suh *et al.* 2008).

system called CAUSE (Computer-Aided Ubiquitous System Engineering) indicated by the dotted box in Figure 1.

At present, most system engineers on the ground develop ubiquitous systems relying on their experience, knowledge, and know-how in a trial-and-error manner, as shown in Figure 2(a). However, as the complexity as well as the size of the applied ubiquitous systems increase, system performance becomes harder to predict and verify without physical implementation. For instance, in a simple ubiquitous system composed of only RFID components, even an experienced engineer reported difficulty when the number of readers was more than three (Nishikawa *et al.* 2006). Further, the difficulty will increase as the number and type of ubiquitous system components (such as sensors, network, etc.) increase, or as the operational scenario gets complicated. Developing a system in this way can incur significant costs to fix errors, especially when the components are expensive. Moreover, fixing errors after implementation seems impossible in ubiquitous systems such as battlefield. Therefore, a computer-aided system engineering, namely CAUSE is needed before the physical implementation of a ubiquitous system. CAUSE helps the ubiquitous

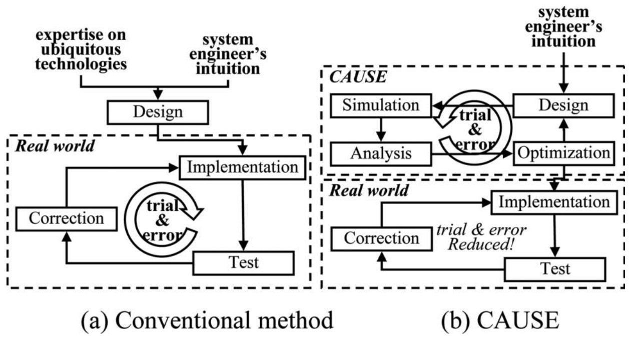


Figure 2. Comparison between the conventional method and CAUSE.

system engineer by providing tools for design, simulation, and analysis in a unified fashion as shown in Figure 2(b), where the trial-and-error is done via computer.

Formally, CAUSE is defined as a methodology for the systematic design of ubiquitous systems through simulation and analysis (Suh *et al.* 2008), and the CAUSE *system* as a software system supporting CAUSE. For the sake of convenience, however, this paper will not differentiate between them. This means that CAUSE often stands for the CAUSE *system*. CAUSE ultimately aims to derive, verify, and validate the optimal design alternative for the target ubiquitous system before the physical implementation. Thus, CAUSE can be regarded as a V&V (Verification and Validation) process for the designed ubiquitous systems. In other words, CAUSE system *verifies* whether or not the designed system works well through simulation and analysis, and *validates* whether or not the designed system is optimal in a certain sense. From another viewpoint, CAUSE can be considered as a CAx (where x means application area, like CAD for design, CAM for manufacturing, and CAE for engineering analysis) tool for ubiquitous system engineering.

It is worth mentioning that the target system for CAUSE can be either: 1) A product manufacturing system, or (2) An information engineering system. In the case of product manufacturing systems, like the contemporary CAD/CAM/CAE, CAUSE should be designed to support design, simulation, and analysis of ubiquitous manufacturing systems. On the other hand, in the case of information engineering systems, CAUSE should support the selection of the sensors and networks used to collect, aggregate, and transmit data to the UPLI, perform the verification and validation process, and conduct simulations. In this paper, for the clarity of presentation, the target system is limited to the product manufacturing system.

The remainder of this paper is organised as follows: Section 2 reviews the previous work related to the CAUSE system; Section 3 discusses the requirement analysis from system-level and functional-level aspects, and Section 4 derives the proposed architecture of CAUSE system; Section 5 presents the implementation strategy including enabling technology; and Section 6 shows the usefulness of CAUSE by introducing a prototype called 'CAUSE for RFID' for a ubiquitous manufacturing system with RFID. Finally, Section 7 concludes the paper.

2. Related work

This section reviews existing tools related to the CAUSE system. Table 1 compares the related work from the viewpoint of the functions which the CAUSE system should have. Specifically, the purpose of the tools is described, and the support of the system engineering and the accommodated technology is compared item by item. Items from the perspective of system engineering include design, simulation, analysis, and evaluation, while accommodated technology means the technology accommodated by the system such as WLAN (Wireless Local Area Network), WPAN (Wireless Personal Area Network), RFID (Radio Frequency Identification), security, and context awareness. These items are marked differently depending on the degree of satisfaction.

System engineering includes previous work in CAx, (for instance CAD/CAM, CAD/CAE, etc.). For instance, ARENA (Arena [Online]), Promodel (Promodel [Online]), etc., provide manufacturing simulation/animation for shop floor system with performance results such as throughput, cycle time, etc. By providing various statistical tools to analyse the simulation results, these tools support the system engineer who intends to derive the optimal design alternative systematically. As shown in Table 1, however, they do not cover the technical analysis for ubiquitous technology, which is a crucial factor in ubiquitous system engineering.

Ubiquitous network simulators such as ns-2 (ns-2 simulator [Online]) and QualNet(QualNet [Online]), etc., configure networks such as WLAN, WPAN, etc., by defining the behaviour of the network components, and then simulate their interaction. The simulation result can be analysed with the help of tools such as event tracers, trend graphs, etc. Tools in this category were developed to evaluate the performance of a certain protocol, and they are mainly utilised for that purpose.

Several ubiquitous system simulation systems were developed from various points of view. UbiWISE (Barton and Vijayaraghavan 2002) is a kind of ubiquitous computing simulator accommodating context-aware technology, which may be used for researchers to develop systems that combine spontaneous interaction and physical-virtual connection. iCAP (Sohn and Dey 2003) and a CAPpella (Dey *et al.* 2004) allow users to create their desired behaviour without writing any code. UbiREAL (Nishikawa *et al.* 2006) is a comprehensive simulator accommodating the ubiquitous technologies of WLAN, WPAN, RFID, and context-awareness. TATUS (O'Neil *et al.* 2005) is a ubiquitous computing simulator supporting the interface with other simulators such as a wireless network simulator. Finally, Rifidi (Rifidi 2006 [Online]) is a dedicated simulator for RFID applications providing design and simulation. Although existing tools partially support system engineering for ubiquitous systems, they are not suitable for comprehensive development. In other words, a new system is needed for developing the ubiquitous system.

3. Design considerations for CAUSE

The CAUSE system is a computer-aided tool, to some extent, a kind of expert system to support design, simulation, and analysis of ubiquitous systems, aiming

Table 1. Comparison of existing works.

		System engineering functions				Accommodated technology				
Tools	Usage/Functionality	Design	Simulation	Analysis	Evaluation	WLAN	WPAN	RFID	Security	Context awareness
ARENA (Arena)	A software to model and analyse business, service, or (non-material handling intensive) manufacturing processes or flows	●	●	●	●	×	×	×	×	×
Promodel (Promodel)	A discrete event simulation software, used for evaluating, planning, or designing manufacturing, warehousing, logistics, and other operational and strategic applications	●	●	●	●	×	×	×	×	×
Ns-2 (ns-2 simulator)	A discrete event simulator targeted at networking research with substantial support for simulation of TCP, routing, multicast protocols, etc. over wired and wireless (local and satellite) networks	○	●	●	×	●	●	●	×	×
QualNet (QualNet)	An ultra high-fidelity network evaluation software that predicts wireless, wired and mixed-platform network and networking device	○	●	●	×	●	●	●	●	●
UbiWise (Barton and Vijayaraghavan 2002)	A software to test the user interfaces of wireless devices and protocols between mobile devices	Δ	●	×	×	N/A	×	×	×	●
UbiREAL (Nishikawa *et al.* 2006)	A software to simulate and test ubiquitous applications with various context	Δ	●	●	×	●	●	●	×	●
TATUS (O'Neil *et al.* 2005)	A virtual ubiquitous computing environment to support SUT (system under test)	Δ	●	●	×	×	×	×	×	●
iCAP (Sohn and Dey 2003)	A system which assists users in prototyping context-aware applications without writing any code	♢	♢	×	×	×	×	×	×	●
a CAPpella (Dey *et al.* 2004)	A context-aware application which supports end users in programming by demonstration recognition-based context-aware behaviours	♢	♢	×	×	×	×	×	×	●
Rifidi (Rifidi 2006)	A simulator to support implementation and test for RFID application by selecting and configuring RFID readers and tags	●	●	○	×	N/A	N/A	●	N/A	N/A

●: strongly satisfied, ○: weakly satisfied, Δ: partly satisfied, ♢: satisfied in a different manner, ×: unsatisfied, N/A: not available.

to derive an optimal design alternative before the physical implementation. Considering the state-of-the-art of the technology, this is a very challenging mission. This section discusses some of the design considerations for CAUSE from the perspective of various stakeholders including CAUSE developers and CAUSE users, respectively categorised into system-level and functional-level aspects.

3.1. System-level aspect

System-level aspects are some of the considerations related to architecture design taken into account by the developer of the CAUSE system.

[SR #1] *Target application (domain)*: Different application domains have totally different architectures and details, which determine the configuration of design, analysis, and simulation. Thus, the first step in designing the CAUSE system is defining the target domain.

[SR #2] *System accuracy (fidelity)*: As one of the primary missions of CAUSE is to provide a means to check the performance of the target system before the physical implementation, the simulation and analysis must be accurate. Thus, a module to enhance the accuracy is required.

[SR #3] *Scalability*: As the number of components used in the ubiquitous system increases, the simulation speed may slow drastically, hampering practical usage. This should be taken into consideration in some fashion.

[SR #4] *Customisation and compromise (CC) mechanism*: Unlike a conventional CAx system, a ubiquitous system runs on various scenarios. Thus, the architecture of CAUSE should be generic to cover scenarios comprehensively, but should be equipped with a means for customisation and compromise (CC) for specific application.

[SR #5] *One-centre concept*: CAUSE should be developed based on the one-centre-concept, so that CAUSE users could perform the entire system engineering task without leaving the CAUSE system. Thus, CAUSE needs to interface with other systems for dedicated functionality and databases.

[SR #6] *Interface with external package*: To provide various requirements, CAUSE needs to have so many algorithms that it would be impractical to develop all of them. To use those that are already implemented and available, e.g., network simulators (ns-2simulator [Online], QualNet [Online]), CAUSE needs to have bi-directional interface for external packages/modules.

[SR #7] *External DB*: System components like sensors and network are used in designing a target ubiquitous system. The catalogue information including technical details and prices often change in the fast growing market. To reflect the timely information, CAUSE needs to access DBs from various sources.

[SR #8] *Collaborative environment*: The CAUSE system can be used simultaneously by multiple system engineers. In order to support the collaborative use of CAUSE, an environment for collaborative design of ubiquitous systems via the Internet should be considered.

[SR #9] *Dealing with a variety of HCI devices*: Human operators may access a ubiquitous system via various means (e.g., PDA, smart phone, desktop) in various situations (e.g., office, workplace, etc.). Thus, CAUSE should be designed to provide customised services.

3.2. Functional-level aspect

Functional-level aspects are some of the functions that the users possibly want to incorporate in the CAUSE system.

[FR #1] *Basic function*: CAUSE should support ubiquitous system development from the design stage to the implementation stage consistently. In order to do this, CAUSE should provide basic system engineering functions such as design, simulation, analysis, evaluation, and documentation.

[FR #2] *Ubiquitous system design*: The target system is defined in terms of system layout, component design, operation scenario, etc. Systematic description of such a system is a big and difficult task. CAUSE should have a module to support this function in some fashion. For such a purpose, a structured method or language including syntax and semantics for describing the system scenario is necessary.

[FR#3] *Knowledge-based system*: All system engineers do not have interdisciplinary expertise including ubiquitous computing technology, information technology, and manufacturing technology. Thus, a knowledge-based design advisor is required, which provides some technical information to help choose the adequate components and install them.

[FR #4] *Interoperability between devices and application systems*: In the ubiquitous environment, various devices and application systems used at the different life cycle stages communicate with each other via wired or wireless networks. As these systems may run on various platforms, CAUSE must guarantee interoperability between the various devices, platforms, and application systems.

[FR #5] *Use of standardised data*: The ubiquitous system aims to collect required information according to the purpose of the system. The data must be standardised for the seamless information flow throughout the entire product life cycle.

[FR #6] *Analysis of system performance*: This function is the most fundamental part of the CAUSE system. System performance must be analysed from various aspects such as productivity (for example, throughput, goodput, delay, loss, etc.), development/maintenance cost, etc.

[FR #7] *System optimisation*: Another key feature to be addressed in CAUSE is optimisation. If the performance of designed systems is acceptable, then the system engineer can select the one that would cost the minimum among the feasible alternatives.

[FR #8] *3D simulation*: Like other CAx systems, CAUSE should show the virtual operation of the designed ubiquitous system realistically. This will help the system engineer recognise undesirable behaviour and invoke better design.

[FR #9] *Verification and validation (V&V)*: One of major functions of CAUSE is V&V for the ubiquitous system. V&V is the process of checking that a system meets specification and fulfils its intended purpose (Sargent 2005). In the case of CAUSE, it means that CAUSE system *verifies* whether or not the designed system works well, complying with the designer's intent through simulation and analysis, and *validates* whether or not the designed system is optimal.

4. Generic architecture of CAUSE system

The various considerations and requirements given in section 3 must be incorporated into the CAUSE system in some fashion. This section derives an architecture by a functional model of CAUSE via IDEF0 representation, followed by an implementation architecture via the SPL (Software Product Line) approach.

4.1. Functional design using IDEF0 diagram

An IDEF0 diagram is a useful tool to model a system in terms of its functional description, inputs, outputs, controls, and mechanisms. Here, it is applied in deriving the functional architecture for CAUSE. Figure 3 and Figure 4 show the topmost and second-level diagrams for the CAUSE system based on IDEF0 notation, respectively. The topmost function of the CAUSE system is developing a ubiquitous system. To develop a ubiquitous system, the information about the target ubiquitous system is required. Thus, system layout, operation scenario, and component specifications are the *inputs* for design. Besides, simulation setup, optimisation objective, and measurement data are required as *inputs* for simulation, optimisation, and validation of the designed system, respectively. As the ultimate goal of the CAUSE system is to derive the optimal design alternative, the optimal design alternative is the ultimate deliverable of the CAUSE system, and it is the *output* of the topmost function of the CAUSE system. In addition, documents for implementation, performance analysis result, and estimated cost are provided.

During the development of a ubiquitous system, domain knowledge as well as some standard, model, language, criteria, and formats are required as the *controls* of the topmost function of the CAUSE system. On the other hand, the *mechanism* involves knowledge base and the inference engine for the knowledge-based system, graphics library and the simulation engine for three-dimensional (3D) simulation, and several supporting packages (modelling, simulation, analysis, drafting).

The topmost function of the CAUSE system consists of the following six second-level functions: (1) design a ubiquitous system, (2) simulate the design alternatives, (3) enhance the accuracy of the CAUSE system, (4) analyse the simulation result, (5) select the optimal design alternative, and (6) document the design alternative. The design, simulation, analysis, evaluation, and documentation functions are derived from the basic functions. In addition, the accuracy enhancement function is added for the verification and validation. Table 2 shows the relationships between the design considerations in functional-level aspect and IDFF0 functions of the CAUSE system.

4.2. Generic architecture of CAUSE system

4.2.1. SPL approach

CAUSE supports a variety of ubiquitous systems with different scenarios, components, and environments. In this paper, we adopt an advanced software engineering approach called SPL (Software Product Line) approach. SPL is powerful in terms of software reuse, variations, and extensibility (Gorton 2006).

From the perspective of the design consideration, the SPL approach is very powerful in the sense that domain-specific CAUSE systems can be developed with common *core assets* and application specific *custom assets*. The core assets are designed to include as much functionality as required for design, analysis, and simulation in various domains, while the custom assets to contain functions individualised to a certain

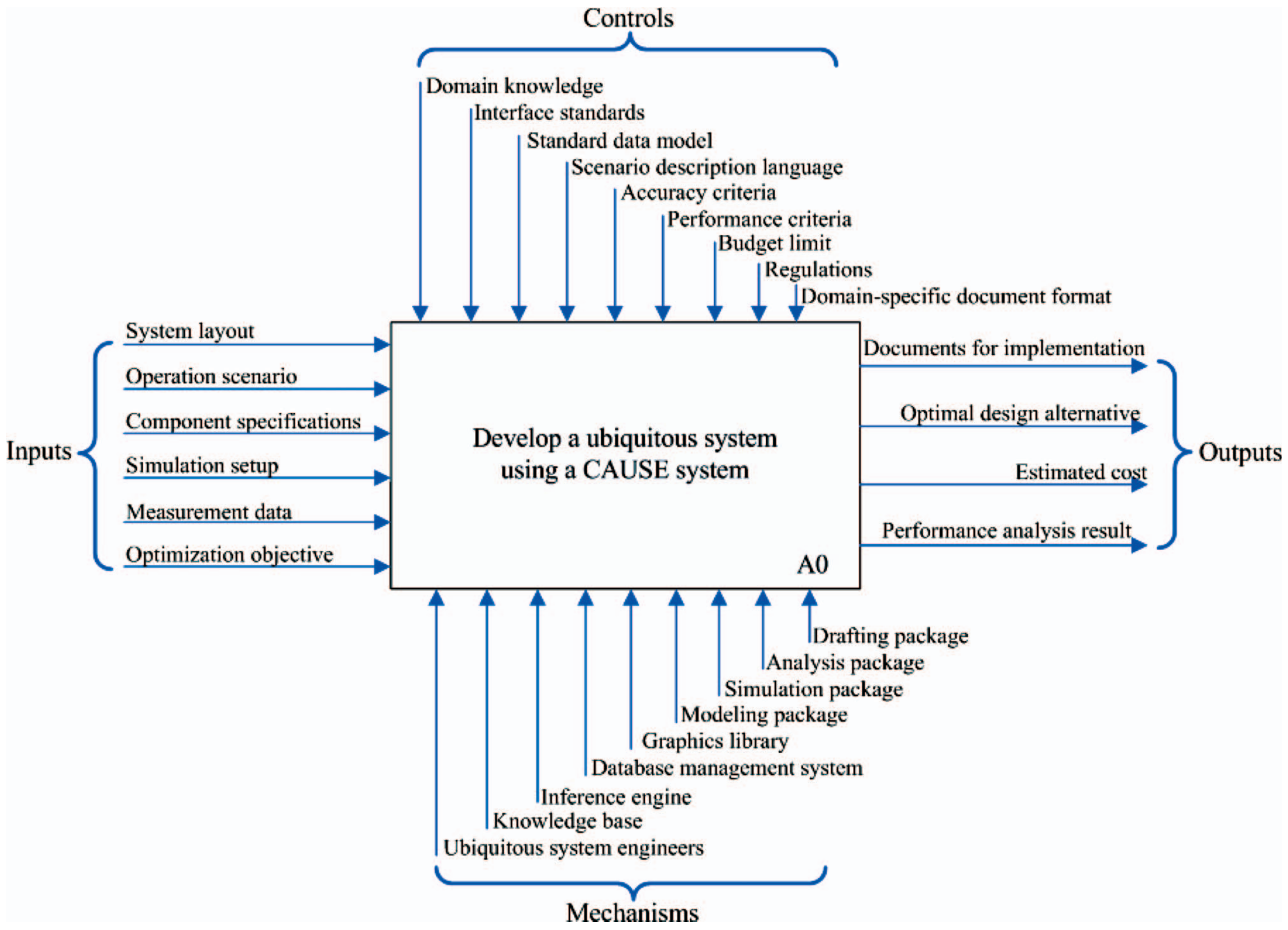


Figure 3. Topmost IDEF0 diagram of CAUSE system.

application domain. In this way, the developer can select a set of appropriate functions and configure them to build a specific CAUSE system for the target domain.

Adopting the SPL approach, we derived a generic architecture of the CAUSE system including three layers: 1) User interface layer, 2) Application layer, and 3) Database layer, as shown in Figure 5. The user interface layer is the custom asset which provides configurable functionalities to develop a domain-specific CAUSE system ([SR#1]). The application layer is the core asset which includes broad functions for engineering a ubiquitous system. The database layer supports a collaborative environment through Internet access ([SR#8]). Note that the architecture has rooms for plug-in modules to accommodate the new technologies and devices. Further, it also supports component substitution and parameter setting for adaptation (note that this is not shown in Figure 5). The following subsections discuss the details for the major modules of the generic architecture shown in Figure 5.

4.2.2. *Designer*

In the CAUSE system, the ubiquitous system is defined as a combination of three kinds of components (people, objects, and devices) with a certain scenario over an environment. Thus, the CAUSE system regards the ubiquitous system as a three-layered structure of an environment layer, component layer, and scenario layer ([SR#4]). To create the ubiquitous system model, the Designer has six sub-modules whose functions are as follows:

Environment designer: Environmental elements, including temperature, humidity, etc., and physical structures such as a factory, hospital, etc., are modelled here. They are represented as the combination of grids.

Component designer: Components are selected by accessing component database. Also, new components can be defined, or existing components can be modified.

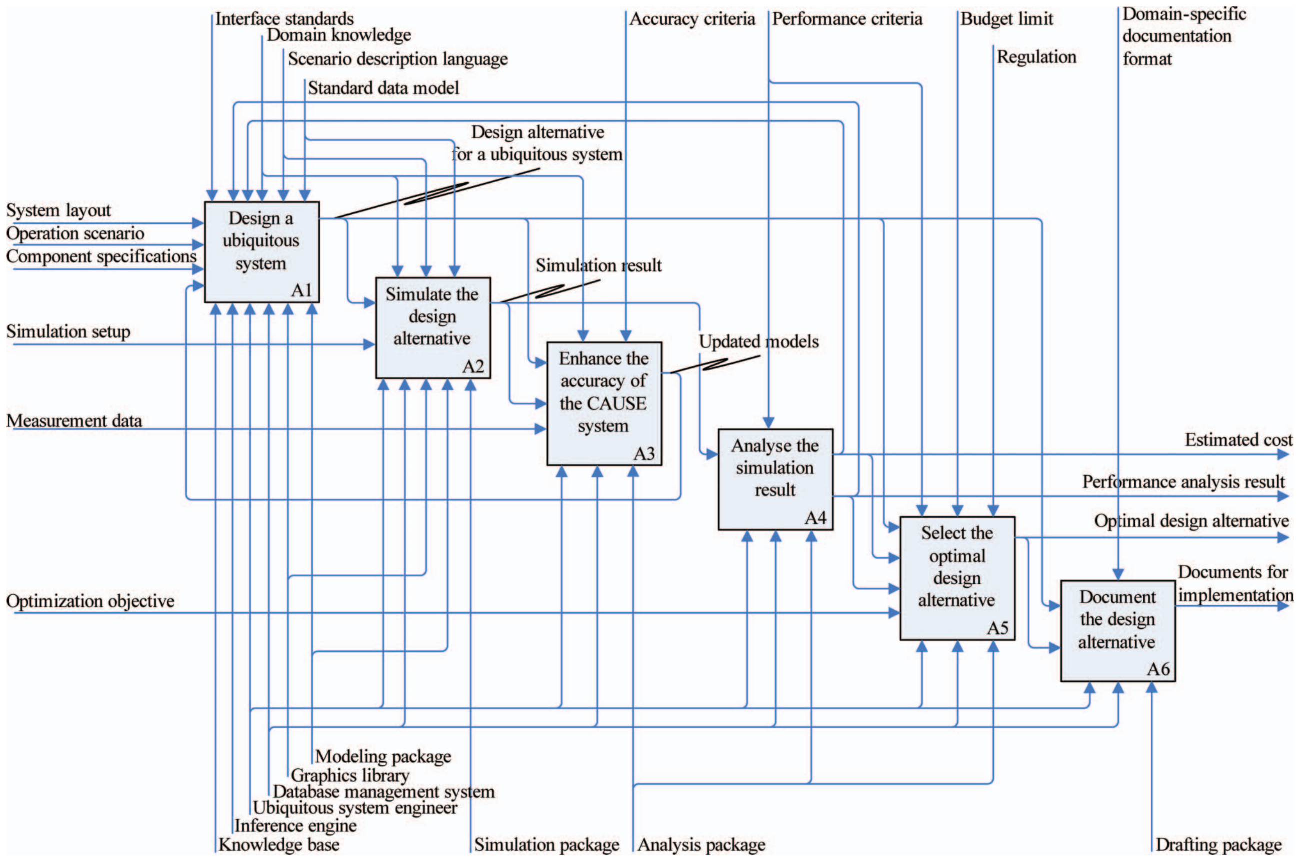

Interface standards
Domain knowledge
Scenario description language
Standard data model
Accuracy criteria
Performance criteria
Budget limit
Regulation
Domain-specific documentation format
Design alternative for a ubiquitous system
System layout
Operation scenario
Component specifications
Design a ubiquitous system A1
Simulate the design alternative A2
Simulation result
Simulation setup
Enhance the accuracy of the CAUSE system A3
Updated models
Measurement data
Analyse the simulation result A4
Estimated cost
Performance analysis result
Optimal design alternative
Select the optimal design alternative A5
Documents for implementation
Document the design alternative A6
Optimization objective
Modeling package
Graphics library
Database management system
Ubiquitous system engineer
Inference engine
Knowledge base
Simulation package
Analysis package
Drafting package

Figure 4. Second level of IDEF0 diagram for CAUSE system.

Table 2. Relationship between the design considerations and IDEF0 functions.

Design considerations in functional-level aspect	IDEF0 2nd-level function
#1, #2, #3, #4, #5, #8	design a ubiquitous system
#1, #2, #3, #5, #8	simulate the design alternatives
#3, #9	enhance the accuracy of the CAUSE system
#1, #6	analyse the simulation result
#1, #6, #7	select the optimal design alternative
#1	document the design alternative

Scenario designer: The behaviour of the component and the interaction among components are defined. Also, the parameters (e.g. position, direction, etc.) of the components are set. The completed design is stored in the ubiquitous system model database through the modelling support module ([FR#2]).

Design advisor: It provides relevant expertise to the system engineer, for example, when a sensor should be installed. This module gives the information about the available range of the sensor, and some guidelines on the installation ([FR#3]).

Modelling support module: It provides functionalities required to use various modelling methods such as programming language, scenario description language, and diagrams. If a system engineer uses a UML diagram to design a scenario or behaviours of a component, the diagram editing function should be supported.

Ubiquitous system model generator: A complete design is generated by putting together all scenarios, components, and environments used in the design. Then, it is stored as a design alternative in the form of the ubiquitous system data model.

4.2.3. *Simulator*

Simulating a ubiquitous system can verify the design alternatives. Besides, the simulation results are stored in the databases for subsequent analysis. The simulation progress is visualised in 3D, and the result can be shown during the simulation. The Simulator is a discrete event simulator including five sub-modules: a batch simulator, interactive simulator, virtual reality

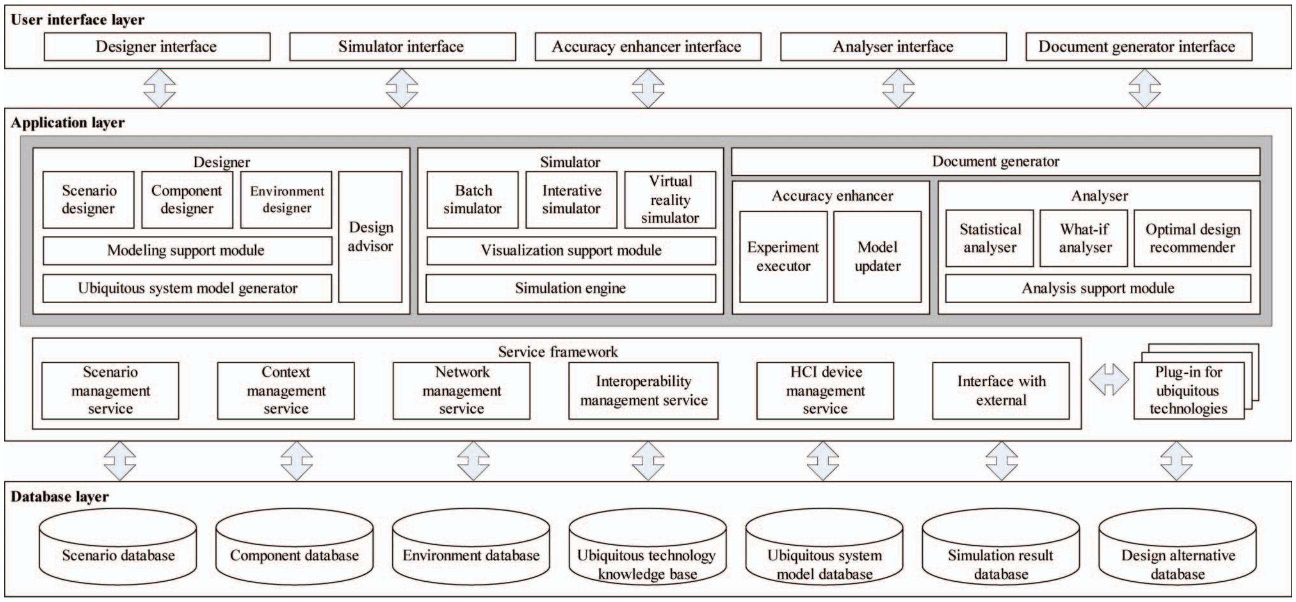


Figure 5. Implementation architecture of CAUSE system.

simulator, visualisation support module, and simulation engine.

Batch simulator: It performs a simulation which does not require user inputs during the simulation. The simulation is executed according to pre-defined scenarios and algorithms.

Interactive simulator: It carries out a simulation which takes the inputs from a user during the simulation. Like in a computer game, a user can control components in the system.

Virtual reality simulator: It provides virtual reality environments where a user can experience the designed system. For this purpose, it can be connected with virtual reality devices such as HMD (Head Mounted Displays) ([SR#9]).

Visualisation support module: It supports the graphical functionalities for the simulators. It includes algorithms for implementing various reality-levels, which range from 2D to 3D to the virtual reality ([FR#8]).

Simulation engine: In order to test the design alternative, CAUSE can repeat the simulation tens or hundreds times. Thus, a simulation engine should process a number of events within a short time ([SR#3]). For example, a fast scheduling algorithm and parallel processing techniques are applied to the simulation engine.

4.2.4. *Accuracy enhancer*

A module for enhancing the accuracy is required as the accuracy of simulation and analysis is crucial for the practical usage of CAUSE ([SR#2]). This would be used in conjunction with physical experiments. This module consists of two sub-modules, and their functions are as follows:

Experiment executor: It defines an accuracy parameter, and makes an experimental plan to estimate the parameter. It takes the real raw data from the experiments and post-processes them. It also provides basic functionalities to analyse the data.

Model updater: It utilises the experimental result to update the analysis model, used in the Simulator and the Analyser. Then, the gap between the reality and the simulation is filled gradually.

4.2.5. *Analyser*

The Analyser includes four sub-modules: statistical analysis, what-if analysis, optimal design recommender, and analysis support modules. Based on the simulation results stored in the log files of the databases, the performance of the design alternative is evaluated by statistical analysis ([FR#6]). Furthermore, it performs what-if analysis, and recommends the optimal design alternative.

Statistical analysis: Statistical techniques such as confidence interval analysis and correlation analysis are used to analyse the simulation result ([FR#6]). The output, in the form of report or chart, can be used in decision making.

What-if analysis: The simulation results of multiple design alternatives are analysed and compared. The various design alternatives are simulated repeatedly, and the results are analysed to derive the optimal design.

Optimal design recommender: The optimal design alternative is recommended based on the cost information and the result of what-if analysis ([FR#7]). Various optimisation techniques are used to maximise the overall objective of the system performance.

Analysis support module: This module supports algorithms used in the analysis and a computing engine for the algorithms.

4.2.6. *Service framework*

In order to adopt the SPL approach, this service framework employs the framework and plug-ins mechanism which provide broad functionalities required to deal with a ubiquitous system. The framework consists of five services and one interface, which provide functions used commonly in the modules of the application layer. It communicates with the database layer and modifies the ubiquitous system data in the database layer. The details are as follows:

Scenario management service: This service provides functions related to the scenario ([FR#2]). For example, it schedules events, triggers the behaviours of components based on the scenario, and changes the states of the components and the ubiquitous system according to the result of the behaviours at the simulation stage.

Context management service: A ubiquitous system completes tasks smartly based on the context. Thus, this service gives functionalities about context generation and awareness. The functionalities enable the devices and system to act intelligently.

Network management service: A network is an indispensable component of a ubiquitous system. This service supports network simulations. It provides capabilities for network protocol and electromagnetic propagation simulation.

Interoperability management service: The ubiquitous environment requires devices and application systems to communicate with each other. For this, this service provides functions related with the communication, transport, storage, and representation of data ([FR#4]).

HCI device management service: HCI devices are widely used for human operators to access a ubiquitous system. This service provides emulation of HCI device in the CAUSE system. This facilitates the development of customised HCI services in the ubiquitous system.

Interface with externals: As developing a complete CAUSE system is very challenging, external packages/devices and databases should be exploited as much as possible ([SR#6], [SR#7]). Furthermore, to enhance the accuracy of a CAUSE system and to test a newly developed HCI service, the interfaces with externals such as hardware devices are required. The function also enables the *one-centre* concept ([SR#5]).

5. Implementation issues of CAUSE system

To embody CAUSE architecture given in section 4, a great deal of research is required, encompassing various domains. This section classifies technology for the implementation, followed by implementation strategy.

5.1. *Enabling technologies for CAUSE System*

Roughly speaking, technologies involved in the implementation of the CAUSE system are classified into three types: 1) System engineering-related, 2) Ubiquitous technology including information and communication, and 3) Sensors and devices including RFID and mobile devices. Taking the state-of-the-art technology in each area into account, they can be summarised as Table 3.

System engineering: Technologies to design, simulate, analyse, and optimise the ubiquitous systems under consideration characterised by the product type and environment where the product is manufactured, used, and recycled.

u-Technology: Mainly networking technologies covered in the CAUSE system. Note that although other u-Technologies can also be covered, discussion about them is omitted here.

Real-world environment & devices: Modelling of environment such as radio propagation model and error model as well as device-level technology for data acquisition technology such as embedded devices, RFIDs, active badges, systems on chip, multi-function sensors and smart cards.

5.2. *Implementation strategy for CAUSE system*

Maximal utilisation of contemporary technology: This is to deal with crucial requirements of CAUSE addressed by [SR#5], [SR#6], and [SR#7]. In implementing the CAUSE system, it is inefficient to make all things. Some simulators or tools for technologies required in CAUSE already exist. For example, dedicated network simulators like ns-2 are available for use. If the existing tools are used, the development time and effort are reduced.

Table 3. Enabling technologies for CAUSE system.

Group	Technology	Features
System engineering		
Design	Modeling language	System modeling and specification language (DEVS, Perti Net, UML, etc.)
	Database	DBs for environment, component, and scenario
	Design advisor	Providing advice for the installation to prevent the design error
Simulation	Models for u-Technology	Models that describe the behaviours of u-Technology listed below
	Simulation engine	Continuous or discrete event simulation Real-time simulation, parallel and distributed computing, etc. DEVS simulation toolkit (DEVSJAVA, CD++, DEVS/C++, etc.)
	Interface with the external	Hardware-, software-, human-in-the-loop simulation Standardised interface such as DIS, HLA, TCP/IP socket
	Visualiser	Animator or visualiser in 2D or 3D manner
Analysis	Statistical analysis toolkit	Technique to process the simulation result
	Optimisation toolkit	Technique to derive the optimal design alternative
u-Technology		
Networking	WPAN	A computer network used for communication among computer devices close to one person (ZigBee, IEEE 802.15, Bluetooth, UWB, IrDA)
	WLAN	Wireless local area network (IEEE 802.11)
	WMAN	Telecommunication technology that provides wireless data long distances in a variety of ways (WiBro, IEEE 802.16, WiMax)
	IP (IPv4, IPv6)	Providing unique address for resources and devices
	Routing protocol	Protocols that specify how routers communicate with each other (For example, DSDV, DSR, AODV, OLSR, etc. as wireless ad-hoc routing)
	Transport protocol	Delivering messages between networked hosts (reliable (TCP) or unreliable (UDP))
Real-world environment & devices		
Channel	Radio Propagation	Channel propagation models (free space model, two-ray ground model, shadowing model, etc.)
Device	RFID	An automatic identification method (Standardisation: IS/IEC 18000 series and EPCglobal)
	Multi-function sensor	Devices to sense the status of resources and products (smart sensor, active badge, web camera, smart card, etc.)
	Mobile device	Portable devices for real-time information exchange (PDA, Laptop, etc.)

Hence, it is better for CAUSE to utilise the contemporary technologies, if possible. Thus, the generic architecture of CAUSE system supports the interface with externals. Here, externals includes real devices such as sensors, VR (Virtual Reality) instrument, etc., as well as existing simulators (e.g. ns-2, QualNet, etc. for network simulators). In implementing the interface, existing standardised interfaces such as DIS (IEEE 1278), HLA (IEEE 1516.1-2000), TCP/IP socket may be used.

Step-by-step implementation: This is to deal with several issues addressed by [SR#1], [SR#2], and [SR#4]. Accuracy of the simulation result is a crucial factor for CAUSE. However, it is impossible to implement CAUSE with a required level of accuracy from the beginning. Thus, we adopt a 'step-by-step implementation' strategy. This means that CAUSE makes up for the inaccuracy by learning. It is related to the 'accuracy enhancer' module. CAUSE system supports it in three ways. It provides 1) various model databases, 2) a function to add new models, and 3) configuration of the parameters. First, by providing the database of various models, a user can select an appropriate model depending on the environment. However, if there is not an appropriate model in the database, a new model can be added. Furthermore, the parameters of the model are adjustable according to the environment of the target ubiquitous system.

Engineering approach to the undealt problem: This is related with issues addressed by [SR#2], [FR#6], and [FR#7]. Even if a number of researches have been made on the modelling of ubiquitous computing technology, their usage in practice is still very limited. This is mainly owing to the fact that theoretic model is not fully accommodating all the parameters existing in the reality, and/or a model is not suitable to all cases. To fulfil the mission of CAUSE, however, uncovered practical parameters should be accommodated in the analysis model. To cope with such a problem, engineering approach based on experimental design and analysis, for instance, has to be developed.

6. Illustrative prototype

Implementation of the CAUSE system presented in this paper requires a great amount of research, time, and effort. This section shows how the CAUSE system looks and addresses the usefulness of CAUSE with a small-scale prototype. The prototype was developed while minimising the effort by implementing only a subset of crucial modules required for the illustration. Thus, the accuracy of the simulation results is not sufficiently guaranteed in the prototype. The accuracy enhancement will be sought in forthcoming publications.

6.1. Prototype system

As shown in Figure 6, the target application of the prototype is a conveyer line carrying multiple parts, where the selection and installation of RFID is a concern for implementation. Thus, the prototype is named as 'CAUSE for RFID'. According to the modules of CAUSE architecture, the *scenario* herein corresponds to a conveyor system delivering multiple products with RFID tags attached to the products, and *objects* and *devices* are products, and RFID readers and tags, respectively. There is no *person* in this prototype.

A main concern in this prototype is the readability of the RFID tags by the RFID readers depending on the setting and configuration. Thus, the performance index is whether or not the RFID tags can be read, together with additional information such as the received power. Another concern is to compare various alternatives with different setting and configuration to derive an optimal design alternative via what-if analysis.

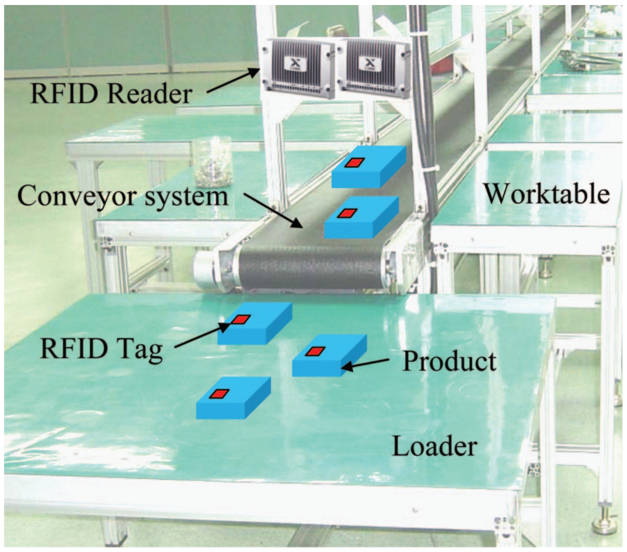


Figure 6. Target application of the prototype.

The developed prototype is composed of three main modules: Designer, Simulator, and Analyser based on the functional structure of section 4. Each module has its own interface and function. The prototype was implemented using Visual C++ and OpenGL. Technically, the prototype adopts the air interface RFID protocol of EPCglobal (EPCglobal), and the parameters of the Hand IT-2G RFID reader and of the PICOPASS 16K RFID tag are used as default values (INSIDE Contactless). The wireless communication channel is modelled by the Friis propagation model (Balanis 1997) with error rate (the ratio of the signals with error bits to the transmitted signals) of $\alpha \in [0,1)$.

Note that Friis propagation model (Balanis 1997) is one of the most used propagation models, some of which are shown in Table 4, where P_t and P_r represent the transmitted and received powers in the unit of dB, respectively; G_t and G_r the antenna gains of the transmitter and the receiver; h_t and h_r the antenna heights of the transmitter and the receiver; d_0 and d the reference distance and the distance between the transmitter and the receiver; λ and L the wavelength and the system loss factor; Γ_t and Γ_r represent the transmitter and receiver reflection coefficients, respectively; $\hat{p}_t$ and $\hat{p}_r$ represent the transmitter and receiver polarisation vectors, respectively; θ_r and ϕ_r represent the angles related to the relative orientation; β the path-loss exponent; X_{dB} a normal random variable with zero-mean and the variance of σ^2.

6.2. Usage scenario

In this subsection, the usage scenario of the prototype is illustrated. In CAUSE for RFID, to design the target system means to find the best configuration (location, specs, number, etc.) of RFID reader and tags so that the products cannot be missed in reading with the minimum cost among the alternatives. The ubiquitous system engineer wants to design and verify the RFID system with the prototype, to find the following: 1) Number, position, and angle of RFID readers, 2) Position of RFID tags, and 3) Specification of RFID readers and tags. To achieve such a goal, the following procedures with CAUSE for RFID are taken step by step.

(1) *Designing the environment*: When CAUSE for RFID starts, a new environment is created, or an existing environment is loaded (Figure 7).

Table 4. Some radio propagation models.

Model	Equation
Free space (Friis) (Balanis 1997)	$P_r(d) = P_t \frac{G_t(\theta_t, \phi_t)G_r(\theta_r, \phi_r)\lambda^2}{(4\pi d)^2}(1 - \lvert\Gamma_t\rvert^2)(1 - \lvert\Gamma_r\rvert^2)\lvert\hat{p}_t \cdot \hat{p}_r\rvert^2$
Two-ray ground reflection (Rappaport 2002)	$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4 L}$
Shadowing (Rappaport 2002)	$\left[\frac{P_r(d)}{P_r(d_0)}\right]_{dB} = -10\beta \log\left(\frac{d}{d_0}\right) + X_{dB}$

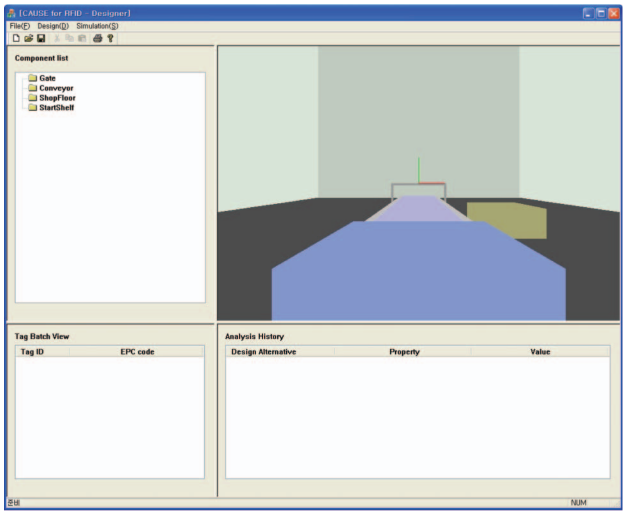
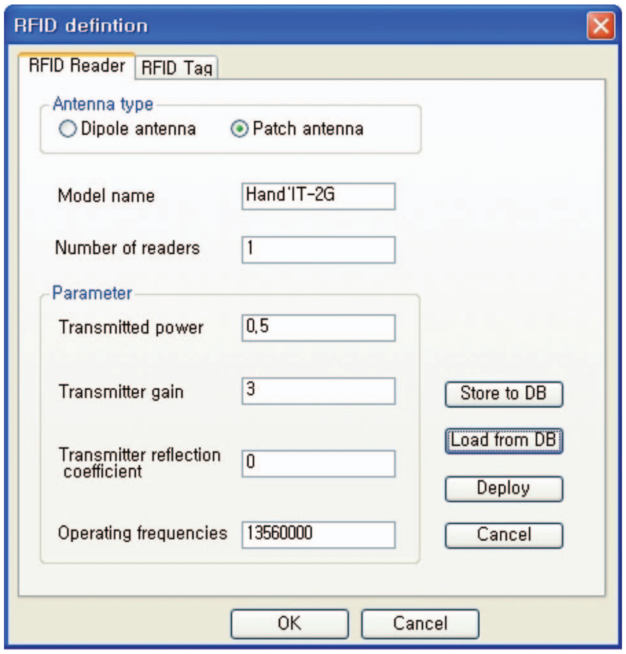


Figure 7. Environment design.


RFID defintion

RFID Reader | RFID Tag

Antenna type: Dipole antenna [✓ Patch antenna]

Model name: Hand'IT-2G

Number of readers: 1

Parameter

Transmitted power: 0,5

Transmitter gain: 3

Transmitter reflection coefficient: 0

Operating frequencies: 13560000

Figure 8. Component design.

(2) *Creating components*: RFID readers and tags and products are created either by selecting them from databases (currently from local databases, but could be linked to that of parts vendors), or by defining new specifications (Figure 8).

(3) *Setting up a scenario*: For what-if analysis, the setting and configuration are changed. Here, the number, the position, and the angle of RFID readers are variables.

(4) *Performing simulation*: The received power is calculated, and the readability is determined. As shown in Figure 9, the simulation progress is visualised in 3D, and the simulation results are displayed. Simulation results are stored as log files as well.

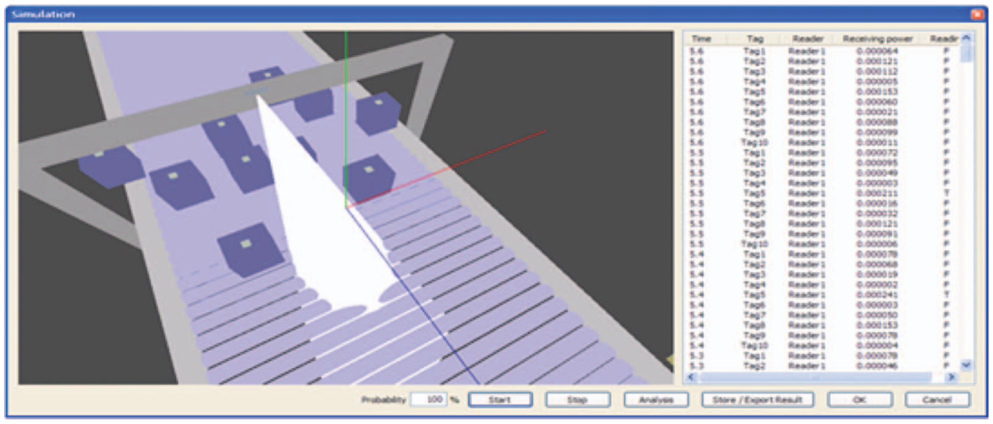
(5) *Analysing the simulation result*: The stored results can be shown in tables or graphs. Figure 10 shows a table for the readability and a graph for the change of the received power according to time.

(6) *Adding the result as a design alternative*: The designed system with its simulation result is added to design alternative list. The stored design alternatives are used for the system optimisation.

(7) *Repeating 2) to 6)*: For what-if analysis, the above steps are repeated under different conditions. The result of each run is stored one by one, and multiple design alternatives are created. Table 5 shows seven design alternatives, among which alternative 6 is selected as the optimal design alternative, as the fewest RFID readers are required.

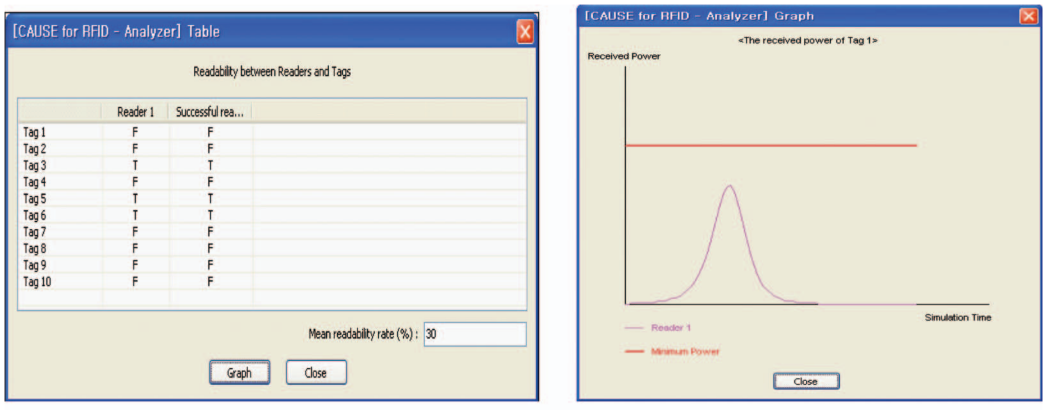
(8) *Recommending the optimal design alternative and printing the details*: After obtaining the optimal design alternative, the building specification is created, and the implementation is launched.

In this way, CAUSE can be used to design the ubiquitous manufacturing system systematically and to determine the optimal design alternative through simulation and analysis. Although CAUSE for RFID is not currently perfect, the above example of the



Time	Tag	Reader	Receiving power	Readir
5.6	Tag1	Reader 1	0.000064	F
5.6	Tag2	Reader 1	0.000121	F
5.6	Tag3	Reader 1	0.000112	F
5.6	Tag4	Reader 1	0.000005	F
5.6	Tag5	Reader 1	0.000153	F
5.6	Tag6	Reader 1	0.000060	F
5.6	Tag7	Reader 1	0.000021	F
5.6	Tag8	Reader 1	0.000088	F
5.6	Tag9	Reader 1	0.000099	F
5.6	Tag10	Reader 1	0.000011	F
5.5	Tag1	Reader 1	0.000072	F
5.5	Tag2	Reader 1	0.000095	F
5.5	Tag3	Reader 1	0.000049	F
5.5	Tag4	Reader 1	0.000003	F
5.5	Tag5	Reader 1	0.000211	T
5.5	Tag6	Reader 1	0.000016	F
5.5	Tag7	Reader 1	0.000032	F
5.5	Tag8	Reader 1	0.000121	F
5.5	Tag9	Reader 1	0.000091	F
5.5	Tag10	Reader 1	0.000006	F
5.4	Tag1	Reader 1	0.000078	F
5.4	Tag2	Reader 1	0.000068	F
5.4	Tag3	Reader 1	0.000019	F
5.4	Tag4	Reader 1	0.000002	F
5.4	Tag5	Reader 1	0.000241	T
5.4	Tag6	Reader 1	0.000003	F
5.4	Tag7	Reader 1	0.000050	F
5.4	Tag8	Reader 1	0.000153	F
5.4	Tag9	Reader 1	0.000078	F
5.4	Tag10	Reader 1	0.000004	F
5.3	Tag1	Reader 1	0.000078	F
5.3	Tag2	Reader 1	0.000046	F

Figure 9. Simulation.


[CAUSE for RFID - Analyzer] Table
Readability between Readers and Tags
Graph
Close
[CAUSE for RFID - Analyzer] Graph
<The received power of Tag 1>
Received Power
Simulation Time
Reader 1
Minimum Power
Close

	Reader 1	Successful rea...
Tag 1	F	F
Tag 2	F	F
Tag 3	T	T
Tag 4	F	F
Tag 5	T	T
Tag 6	T	T
Tag 7	F	F
Tag 8	F	F
Tag 9	F	F
Tag 10	F	F

Mean readability rate (%) : 30

(a) Readability between readers and tags (b) The received power of tag

Figure 10. Two types of simulation results.

Table 5. Deriving the optimal design alternative.

	Condition of RFID readers			
No.	Number	Position	Angle	Readability
1	1	1/2	45	40%
2	2	1/2, 2/3	45	80%
3	3	1/4, 2/4, 3/4	45	100%
4	4	1/5, 2/5, 3/5, 4/5	45	100%
5	3	1/4, 2/4, 3/4	0	100%
6	2	1/3, 2/3	0	100%
7	1	1/2	0	70%

usage scenario is sufficient to show that CAUSE system enables a system engineer to obtain the optimal design alternative before the physical implementation.

7. Concluding remarks

This paper proposed a system engineering approach called CAUSE, as a core element embodying the UbiDMR paradigm, for ubiquitous manufacturing system design, analysis, and verification. Considering that the contemporary method is a manual method based on trial-and-error, requiring time and cost, CAUSE is a research topic of great worth. In terms of realisation, CAUSE requires highly complex technologies throughout the domains of system engineering, and ubiquitous computing software/hardware technologies.

Through the comprehensive survey and analysis of the contemporary technology, we derived a systematic procedure for the CAUSE system: 1) Requirement analysis from the perspectives of system and functional aspects, followed by 2) Architecture design, and 3) Implementation strategy. To show the usefulness of the CAUSE, we developed a prototype system for designing manufacturing system with RFID. The prototype showed that CAUSE can be used as a means for implementing ubiquitous manufacturing system before the physical implementation.

As previously mentioned, the accuracy issue is very important, as it is directly coupled with the effectiveness of CAUSE. Accuracy enhancement of CAUSE and incorporation of other ubiquitous elements, such as ubiquitous sensors/network are left for the future. Currently, we are developing accuracy measurement and enhancement methods under various conditions.

Acknowledgements

This research was in part supported by the BK21 program for *emerging technology* funded by the Ministry of Education and Science in Korea, and by the grant for *The ubiquitous factory system design and control* funded by Doosan Infracore Ltd. in Changwon, Korea between 2006 and 2008.

References

Arena [Online]. Available from: http://www.arenasimulation.com

Balanis, C.A., 1997. *Antenna theory: analysis and design*. 2nd ed. New York: Wiley.

Barton, J.J. and Vijayaraghavan, V., 2002. *UBIWISE, a ubiquitous wireless infrastructure simulation environment*. HP Lab.

Dey, A.K., Hamid, R., Beckmann, C., Li, I., and Hsu, D., 2004. A CAPpella: programming by demonstration of context-aware applications. *In*: *Proc. CHI'04*.

EPCglobal, 2006. *EPC Radio-Frequeny Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communication at 860MHz-960MHz*.

Gorton, I., 2006. *Essential Software Architecture*. New York: Springer.

Huang, G.Q., Wright, P.K., and Newman, S.T., 2009. Wireless manufacturing: a literature review, recent developments, and case studies. *International Journal of Computer Integrated Manufacturing*.

Huang, G.Q., Zhang, Y.F., and Jiang, P.Y., 2007. RFID-based wireless manufacturing for walking-worker assembly islands with fixed-position layouts. *Robotics and Computer-Integrated Manufacturing*, 23, 469–477.

Huang, G.Q., Zhang, Y.F., and Jiang, P.Y., 2008. RFID-based wireless manufacturing for real-time management of job shop WIP inventories. *International Journal of Advanced Manufacturing Technology*, 36, 752–764.

IEEE 1278, 1995. *Standard for Distributed Interactive Simulation*.

IEEE 1516.1-2000, 2000. *Standard for Modeling and Simulation High Level Architecture – Federate Interface Specification*.

INSIDE Contactless [Online]. Available from: www.insidecontactless.com

Lee, B.E. and Suh, S.H., 2008. An architecture for ubiquitous product life cycle support system and its extension to machine tools with product data model. *International Journal of Advanced Manufacturing Technology*, 42, 606–620.

Nishikawa, H., Yamamoto, S., Tamai, M., Nishigaki, K., Kitani, T., Shibata, N., Yasumoto, K., and Ito, M., 2006. UbiREAL: Realistic smartspace simulator for systematic testing. *LNCS*, 4206, 459–476.

ns-2 simulator [Online]. Available from: http://www.isi.edu/nsnam/ns

O'Neil, E., Klepal, M., Lewis, D., O'Donnell, T., O'Sullivan, D., and Pesch, D., 2005. A testbed for evaluating human interaction with ubiquitous computing environments. *In: Proceedings of Tridentcom 2005*.

Promodel [Online]. Available from: http://www.promodel.com

QualNet [Online]. Available from: http://www.scalable-networks.com

Rappaport, T., 2002. *Wireless communications: principles and practice*. 2nd ed. New York: Prentice Hall.

Rifidi: software defined RFID technology, 2006 [Online]. Available from: http://www.rifidi.org

Sargent, R.G., 2005. Verification and validation of simulation models. *In*: *Proceedings of Winter Simulation Conference*, 130–143.

Shin, S.J., Yoon, J.S., and Suh, S.H., 2007. A conceptual model for ubiquitous factory. *In*: *Proceedings of Korean Society of Precision Engineering*.

Sohn, T. and Dey, A., 2003. iCAP: an informal tool for interactive prototyping of context-aware applications. *In*: *Proc. CHI'03*.

Suh, S.H., Shin, S.J., Yoon, J.S., and Um, J.M., 2008. UbiDM: A new paradigm for product design and manufacturing via ubiquitous computing technology. *International Journal of Computer Integrated Manufacturing*, 21, 540–549.

Um, J.M., Yoon, J.S., and Suh, S.H., 2008. An architecture design with data model for product recovery management systems. *Resources, Conservation and Recycling*, 52, 1175–1184.