IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 3 July 2014; revised 17 November 2014; accepted 28 December 2014; date of publication 18 November 2015;
date of current version 7 December 2016.

Digital Object Identifier 10.1109/TETC.2015.2501846

HyperspaceFlow: A System-Level Design Methodology for Smart Space

JING ZENG[1], LAURENCE T. YANG[2], JIANHUA MA[3], AND MINYI GUO[4]

[1]School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[2]Department of Mathematics, Statistics and Computer Science, St. Francis Xavier University, Antigonish, NS, B2G 2W5, Canada
[3]Faculty of Computer and Information Sciences, Hosei University, Tokyo 184-8584, Japan
[4]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

CORRESPONDING AUTHOR: L. T. YANG (ltyang@ieee.org)

The work of J. Zeng was supported by the National Natural Science Foundation of China under Grant 61472150 and Grant 61173045.
The work of L. T. Yang was supported in part by the National Sciences and Engineering Research Council, Canada,
and in part by the Canada Foundation for Innovation.

ABSTRACT Smart space technology gains intensive attentions of the worldwide due to the advancement of pervasive computing. However, the design of smart space is inherently complex as it resides in highly varying environment with multiple devices, also the power consumption, cost, and user preference are usually ignored by designers, resulting in huge design cost, high power dissipation, and unsatisfactory user experience. In this paper, we present a system-level design methodology for smart space, which is able to refine the function specification of smart space into underlying implementations leveraging an intermediate representation model HyperspaceFlow. The tasks in smart space are deposited into distributed execution platforms, and a network is automatically synthesized to satisfy their needs of communication at given QoS constraints. Meanwhile, the proposed design methodology is able to offer relatively satisfactory cost, power consumption, and user preference of the design solution at given constraints. At last, a home health care case is used to elucidate our design methodology, and the simulation results demonstrate the effectiveness and feasibility of our methodology.

INDEX TERMS System-level design, ubiquitous computing, smart space, home health care, multi-objective optimization.

I. INTRODUCTION

With the advancement of ubiquitous computing, the vision of Mark Weiser is gradually realized in our real world, which allows to offer the computing capabilities for individuals at anytime and anywhere [1]. Further, various novel computing paradigms have emerged, such as Smart Hyperspace [2], Cybermatics [3]–[5], etc.

Smart space is a complex computing system that highly fuses human, computers as well as things. The key technologies for designing smart space closely relate with ubiquitous computing, communication and control technologies [2]. Unfortunately, the design of smart space is a fairly tricky task. Firstly, smart space is a considerably complex system due that it involves multi-disciplinary technologies, such as communication, control and software technologies, etc. Secondly, the design process of smart space lacks reliable methods of designing, analyzing and validating. Thirdly, there are manifold aspects that seriously influence the development of smart space, such as low efficiency of the software tools, labor-consuming and time-costing development process, etc. Finally, power consumption and user preference are badly neglected by designers, which results in high power cost and poor user experience. Therefore, an effective, efficient, reliable and user-friendly design of smart space is a challenging issue. Hence, to address these challenges, systematic design methodologies are needed to guide the effective design of smart space.

Aiming at addressing the tricky design problems of smart space, we employ the system-level design methodology [6], which characterizes the mapping from the function specification to architecture implementation at given platform according to given constraints. Typically, the mapping procedure is the refinement of function specification leveraging an intermediate representation model (IRM).

We present the HyperspaceFlow as the IRM to capture the function specification and user preference then transform

them into underlying implementation, which includes object emplacement, system synthesis and preference synthesis. The object emplacement accomplishes the location allocation of the physical objects, cyber objects in space, then the system synthesis proceeds with computation synthesis and communication synthesis. Specifically, the former conducts computation resource assignment according to the requirements of given tasks, while the latter automatically synthesizes a network to meet the communication QoS requirements among computation devices. The preference synthesis considers the user satisfaction on device and HCI manner. Otherwise, the final implementation is capability of achieving the satisfactory cost, power consumption, and user preference at given constraints.

The contributions of our work can be summarized as follows:

- A coarse-grained model of computation is presented, called HyperspaceFlow, which can be used to model the design flow of smart space, which is modeled as physical flow, data flow, and human flow. Besides, the physical flow can specify the relations between cyberspace and physical space; the data flow is capability of capturing the computation and communication relation of cyberspace; human flow is utilized to model the interaction of cyberspace and human space. Furthermore, we can conduct system-level power, cost, user preference estimation leveraging the proposed model.
- The system-level design framework for smart space is proposed to accomplish the mapping from requirements of smart space applications to the underlying architecture. Also the space model and the proposed model are combined to ensure the nearly optimization design.
- We use the HyperspaceFlow to automatically synthesize the solutions at given platform library, meanwhile to attain the nearly optimal solution at cost, power and user preference. The power consumption estimation algorithm, cost estimation approach and preference estimation algorithm are presented to assist the triple-objective optimization. Besides, we solve the triple-objective integer linear optimization (ILP) problem by the genetic algorithm.

The reminder of the paper is organized as follows. In section II, we introduce the related work about the design of smart space. In section III, we firstly introduce the system-level design framework of smart space, then an IRM is specified as the formal computation model. Subsequently, the mapping process is described in detail. Section IV gives a case study to verify our design methodology with home health care application. Section V discusses the simulation results of the design. Finally, we summarize the discussions and give some conclusions in section VI.

II. RELATED WORK

Many efforts have been focused on the study of smart space. Notably, the work in [2] presents the smart hyberspace, and illustrates a case with kids care application in smart space.

Regarding to multiple domains, smart spaces increasingly gain widely applications. For smart home or house, the Gator tech Smart House at the University of Florida implements a programmable pervasive space [7], where various smart devices (such as smart plug, smart projector, smart floor, etc) are located in physical space. Another typical project is CASAS in Washington State University, which presents a lightweight smart home design [8]. The project highlights the activity recognition by sensor data collecting from the house. Elders or children care and health monitor are its major application.

With respect to intelligent transport system, MIT cartel project is developed to address the road transportation problem by means of mobile sensing, wireless networking and data-intensive algorithms [9]. The researchers of Nanyang Technological University in Singapore propose an approach to predict bus arrival time based on mobile phones [10] through passengers participating and sharing the information of bus location, which is transferred to backend server where online data analysis and processing are executed. Then, the result will be delivered to the real time query as soon as possible. Moreover, to assist drivers to enhance driving directions, Yuan *et al.* [11] present a smartcloud-based driving direction system leveraging experienced drivers. The system can provide the fastest route for a given destination at given time. Essentially, landmark graph approach and travel time estimation are the critical idea of the system.

Also, smart space can be recognized as an effective solution to improve traditional health care. Some of its capabilities such as automatic patient diagnosis and elder monitoring by kinect, smart phone as well as online social network [12], sleeping situation monitoring of the patient [13], u-pillbox system for humanistic geriatric healthcare [14], etc. Towards health monitoring, Vergari *et al.* [15] provide a cross-domain application to dynamically relate medical and ambient information.

Despite various applications of smart space are developed, however, most of them are application specific and lack of systematic design method, furthermore, user impact is badly ignored by designers. All of these shortcomings result in huge cost, high power consumption and poor user experience in smart space design.

System-level design methodology provides a promising approach for the design of smart space, but few works focus on the design methodology of smart space. In [16], the authors present a systematic design approach for smart space by using high level language RCAL to represent application logic, then transform it into the underlying distributed sensor network leveraging a compiler, which parses the RCAL language and automatically generates execution code that can be deployed into sensor nodes via given mapping rule. It enables the separation between application and implementation. However, it does not refer to the optimal solution of the design at performance and cost. The work in [17] utilizes the model driven architecture to develop home automation system. Component based approach and interactive model

are presented to enable the association between platform independent model and platform specific model. The authors use services and modes to model interaction. Services are employed to satisfy the interaction needs between user and environment. Modes indicate the restriction on computation resource according to user defined requirements. However, it also does not involve the design optimization of system.

Hence, most of previous research efforts mainly focus on the specific application of smart space, little works place more attention on the design methodology of smart space in system-level. We present a petri net based model to model the human, computers and things of smart space, meanwhile the nearly optimization at design level can be achieved leveraging the proposed model.

III. SYSTEM-LEVEL DESIGN METHODOLOGY

A. OVERVIEW OF THE DESIGN FRAMEWORK

System-level design methodology refers to the refinement from application specification to architecture implementation according to given constraints. The illustration given in Fig. 1. is the basic system-level design framework of smart space. We employ increasingly popular platform based design methodology [6], [18] to achieve the system-level design. This design methodology considers that the procedure of system design can be identified as the mapping from application model to the given platform, which consists of multiple platform libraries. The mapping procedure is the exploration of design space at given platform, which combines with corresponding space model, meanwhile it can accomplish the nearly optimal performance and cost according to given constraints.

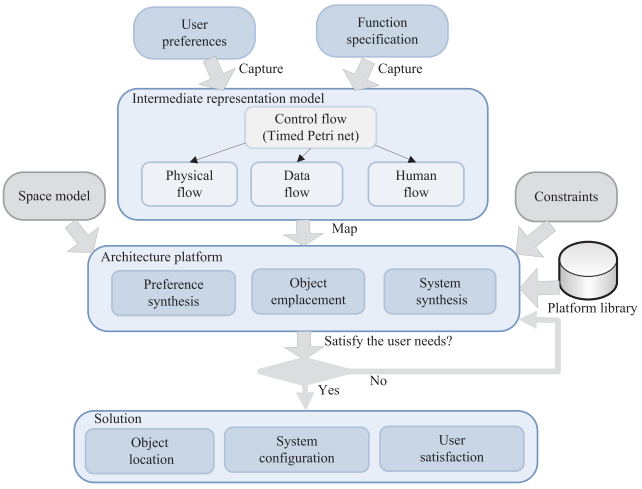


FIGURE 1. The system-level design framework for smart space. It is a refinement process from high-level definition to underlying architecture. The multiple abstraction levels mask the bottom design details.

As the Fig. 1. depicts, basically, our methodology is divided into three steps. Firstly, function specification is a state based language that allows to define the structure and behavior of smart space in unification. Specifically, we extend the language in [16] to achieve the definition of user preference and function specification. Secondly, the defined function in specification can be captured by a compiler, which allows to transform the function into IRM. Concretely, the output data of IRM are saved as PNML [19] for control flow, and signed directed graph denotes the data flow, physical flow and human flow. Finally, the mapping from HyperspaceFlow to architecture platform is abstracted as three classes of optimization problems, which take intermediate representation as input data. Specifically, it is typically divided into threefold: preference synthesis, object emplacement and system synthesis, respectively.

Preference synthesis refers to user satisfaction on the design of smart space. User preference incorporates location, device manufacturer, the manner of human computer interaction, etc. Furthermore, preference can be quantitatively collected from users leveraging specific computation model, which will be discussed in later section.

Object emplacement is that allocating the physical objects and cyber actors into the space and meanwhile ensuring the maximum location satisfaction to users. The process is modeled via physical flow and data flow of IRM. Further, to reduce the exploration space of candidate location, computation nodes location is able to be gained via calculate the distance of computation nodes with location of physical objects.

System synthesis incorporates computation synthesis and communication synthesis. The former achieves the software/hardware partition and assigns corresponding computation resources to the task defined in the specification. For the software, processors can be selected to tailor the given target task. Hardware is enabled like FPGA, ASIC, etc. Software/hardware partition is precisely determined by cost, performance and power of different tasks. The latter is conducted to guarantee the communication among tasks of the distributed system under given constraints, which refer to the assignment of protocol, topology as well as routing.

The design decisions of smart space in the mentioned three steps, in essence, can be abstracted as multi-objective ILP and NP-hard. Accordingly, we can get the design solution from solving the problem, and heuristic algorithms provide quite promising approaches for solving such class of problems [20].

The generated solutions can be outputted when they satisfy the users' needs, otherwise keep iterations to generated new solutions. The output consists of object location, system configuration and user satisfaction. The results of object emplacement are the physical object locations and the computation node locations. System configuration gives the required system platforms selected from the computation library, and network solution including network topology, protocol, routing, as well as the required network switch devices. User satisfaction is a real value to each generated solution.

B. INTERMEDIATE REPRESENTATION MODEL

Smart space is a distributed application that involves physical objects, cyber objects and human objects. Essentially, such distributed application exhibits partial order relation and characterizes as asynchronous and concurrent. To exactly model the design flow of smart space, reasonable IRM is needed to capture the design requirements and to tailor the needs of system design. Based on the investigation on existing IRM [21], we further extend it into smart space domain. By using IRM, we can precisely capture the relationships among physical objects, cyber objects as well as human objects. They are named with physical flow, data flow and human flow.

cyber objects and physical objects are modeled via physical flow, the relationships between data flow captures the computation and communication of cyber space, while human flow depicts the interaction relations between cyber and human. To capture these relationships in smart space, we present a coarse-grained model of computation that is used to capture the design representation of smart space and to aid for accomplishing the system-level design. Here, we call it Hyperspace-Flow model and make it adapt to the modeling requirements of smart space.

HyperspaceFlow allows to capture the flow relationships within smart space. Specifically, each cyber processing element is considered as an actor [22], which is a black box and capability of dealing with input events and producing output events. Thus, we only model the external communication events among actors, which are interconnected by communication network. Besides, the elements in physical space and social space are considered as objects. The model formulation is given as follows:

Definition 1: A physical object is a two-tuple, $PO = (PO_i, PO_s)$*, where*
PO_i *is the ID of the physical object, which distinctively indicates the physical object;*
PO_s *is the state value of the object.*

The physical object is the monitoring or operation object of sensor and actuator. It characterizes with state, and its changing affects cyberspace operation as well as human activity.

Definition 2: An Event is a two-tuple, $E = (E_n, E_v)$*, where*
E_n *is the name of the event;*
E_v *is the value of the event.*

The definition gives the event of the structure. An event is an object that includes a name to specify the type and a value to ensure the uniqueness. Particularly, various events will result in multiple actions. For example, a control event can trigger the control operation, while a data flow event triggers the data transmission.

Definition 3: An actor is a triple tuple, $A = (E_{in}, E_{out}, F_m)$*, where*
E_{in} *is the set of input events.* $E_{in} = E_1 \cup E_2 \cup \ldots \cup E_n$, E_j *denotes the input event j to the actor;*
E_{out} *is the set of output events.* $E_{out} = E_1 \cup E_2 \cup \ldots \cup E_n$, E_k *denotes the output event k to the actor;*
$F_m : E_{in} \rightarrow 2^{E_{out}}$ *is the mapping from the input events to the output events.*

Here, we use 2 to denote the mapping from one set to another set. Actors are well suited to model the asynchronous and concurrent system. In essence, the interaction among actors mainly relies on sending messages. Here, the critical task of the actors is to cope with input events from other actors and then to create output events. Furthermore, actor is the abstraction of the processing element, it can be realized through software or hardware.

Definition 4: Two actors A_i *and* A_j *are semantically equivalent, if they have the same input events and produce the same output events. It can be represented as* $A_i \equiv A_j$*, if* $A_i(E_{in}) = A_j(E_{in}), A_i(E_{out}) = A_j(E_{out})$.

Despite two actors own different internal structures, such as, processing the event by diverse methods. However, they have the same input and output, essentially, they are still considered as semantically equivalent. Thus, it is expected that various candidate processing elements for refining the actors can be selected to enable the system synthesis and meanwhile to maintain the semantics.

Definition 5: A human object is a three-tuple, $H = (H_i, H_f, H_p)$*, where*
H_i *is the ID of the human object;*
H_f *is the interface of the human and computer interaction;*
H_p *is the preference value of the human.*

This definition describes the characteristics of human objects. The interface of human and computer interaction (HCI) can employ multiple manners, such as textual, acoustic, visual, or hybrid, etc. Here, the human preference value can denote various types (location, manufacturer, interaction manner, etc), which are further determined by specific applications.

Definition 6: A physical flow P is a five-tuple, $P = \{PO, A, ST, AE, PW\}$*, where*
$PO = \{PO_1, PO_2, PO_3, \ldots, PO_n\}$ *is a finite set of vertices, each of them represents a physical object;*
$A = \{A_1, A_2, A_3, \ldots, A_n\}$ *is a finite set of vertices, each of them represents an actor;*
$ST = ST(PO_1) \cup ST(PO_2) \cup \ldots \cup ST(PO_n)$ *with* $ST(PO_j) =$ *the set of state with vertex* PO_j*;*
$AE = AE(A_1) \cup AE(A_2) \cup \ldots \cup AE(A_n)$ *with* $AE(A_j) =$ *the set of input event associated with vertex* A_j*;*
$PW \subseteq (ST \times AE) \bigcup (AE \times ST)$ *is a finite set of edges, each of them denotes the association between physical object and cyber object.*

The definition demonstrates that the change of the physical object state can affect the cyber object actor, also actor is able to act on the physical object and further to change its state value.

Definition 7: A data flow D is a five-tuple, $D = \{A, I, O, DW, C\}$*, where*
$A = \{A_1, A_2, A_3, \ldots, A_n\}$ *is a finite set of vertices, each of them represents an actor;*
$I = I(A_1) \cup I(A_2) \cup \ldots \cup I(A_n)$ *with* $I(A_j) =$ *the set of input events associated with vertex* A_j*;*

$O = O(A_1) \cup O(A_2) \cup \ldots \cup O(A_n)$ *with* $O(A_j)$ *= the set of output events associated with vertex* A_j*;*
$DW \subseteq O \times I = \{< O, I >| \ o \in O, i \in I\}$ *is a finite set of edges, each of them denotes an interconnection from an output event of an actor to an input event of another actor. They are interconnected through the communication network;*
$C : DW \rightarrow N$*, is a function that represents the exchange token numbers among actors. N denotes the natural number.*

The definition of data flow specifies the basic data dependency relationships among actors and the data flow is an acyclic directed graph. Specifically, they enable the association based on the events that are transmitted by network. According to [23], the communication model among actors is modeled as an unblocking FIFO channel, also, employing token as the metric of exchange data among actors. Token can be seen as the abstraction of actual bit numbers, they have a corresponding relationship.

Definition 8: A human flow S is a five-tuple, $Z = \{H, A, HF, AE, IW\}$*, where*
$H = \{H_1, H_2, H_3, \ldots, H_n\}$ *is a finite set of vertices, each of them represents a human object;*
$A = \{A_1, A_2, A_3, \ldots, A_n\}$ *is a finite set of vertices, each of them represents an actor;*
$HF = HF(H_1) \cup HF(H_2) \cup \ldots \cup HF(H_n)$ *with* $HF(H_j)$ *= the set of interface associated with vertex* H_j*;*
$AE = AE(A_1) \cup AE(A_2) \cup \ldots \cup AE(A_n)$ *with* $AE(A_j) =$ *the set of input events associated with vertex* A_j*;*
$IW \subseteq (HF \times AE) \bigcup (AE \times HF)$ *is a finite set of edges. Each of them indicates the interaction between actors and human objects by HCI.*

The definition of human flow specifies the interaction relationships between cyberspace and human space. HCI interface allows to employ various manners, which can be determined according to the actual application scenario, such as cost, user preference and power, etc.

Definition 9: A distributed physical, data, human/control flow system ϑ *is represented as nine-tuple,* $\vartheta = (P, D, Z, S, T, F, J, K, M_0)$*, where*
$P = (PO, A, ST, AE, PW)$ *is a physical flow;*
$D = (A, I, O, DW, C)$ *is a data flow;*
$Z = (H, A, HF, AE, IW)$ *is a human flow;*
$S = \{S_1, S_2, \ldots, S_n\}$ *is a finite set of control places;*
$T = \{T_1, T_2, \ldots, T_n\}$ *is a finite set of transitions;*
$F \subseteq (S \times T) \cup (T \times S)$ *is a binary relation, named control flow relation;*
$J : S \rightarrow 2^{(PW \cup DW \cup IW)}$*is a mapping from control places to sets of edges of the given physical flow, data flow, as well as human flow graph. i.e, the arc of the flows is controlled by a control place* S_j*;*
$K : O \rightarrow 2^T$ *is a mapping from output events of data flow vertices to sets of transitions. Each transition is guarded by the corresponding actor;*
$M_0 : S \rightarrow \{0, 1\}$ *is an initial marking function.*

To meet the needs of asynchronous and concurrent characteristics of smart space, we employ petri net to build the control flow model, where control places or S-elements are used to indicate the control signals among physical object, actor and human object. Each edge on physical flow, data flow and human flow can correspond a control place which determines the interaction among physical space and cyberspace, cyberspace internal, cyberspace and human space. When a control place owns a token, a control signal will be sent to the corresponding edge in physical flow, data flow or human flow. It is indicated by mapping J. Also, the diverse events which are created by actors, or triggered by physical objects and human objects are represented as T-elements or transitions. In fact, disparate transitions will result in triggering diverse control places, which further affect the cyberspace interaction with physical and human space. Especially, actors perform the condition judgement and produce the output events or the transitions. It is denoted as mapping K in the definition.

Definition 10: A timed distributed physical, data, human/control system is defined as two-tuple, (ϑ, L)*, where* ϑ *is a distributed physical, data, human/control flow system, and* $L = \{l_1, l_2, \ldots, l_n\}$ *denotes a finite set of time intervals. Each of them corresponds to a control place, and it must satisfy the following conditions:*

(1) When a place has a mark, the transition will be enabled immediately;
(2) The transition needs costing a time interval to finish the procedure.

For a real distributed system, timing property should be taken into account. Usually, in a real scenario, an event that is transmitted on the network will cost a period. Here, we model the temporal property with a timed petri net [24]. The time interval is the capability of representing the time interval for network transmission. However, it is a tricky problem to preserve the clock synchrony within the distributed systems. Hence, we use the approach presented in [25] to solve synchronous clocking problem. As a result, the time intervals can be denoted as one or various such clock cycles.

Definition 11: For distributed physical, data and human/control flow system, assuming that the marking function $M : S \rightarrow N$*,* $N = \{0, 1, 2, \cdots\}$*. Its behavior is defined as follows:*

(1) Initial marking function M_0 *assigns the tokens to places that determine the system initialization, i.e, for place* S_i*,* $M_0(S_i) = 1$*. Various places can consist of tokens when initialization. In fact, the distributed system allows to include multiple concurrent subsystems or processes;*
(2) The transition T is enabled when its input place S includes at least one token, i.e., $M(S) \geq 1$*;*
(3) When the transition contains various guarded conditions, the output event of actor performs the logic judgement and determines whether or not to transmit the data to another actor;
(4) When the transition is fired, the token will be removed from the control place and producing a token to each of their output places. In this procedure, the transition will be enabled immediately. However, it will keep a time interval to finish the transition. To some extent,

it is capable of tailoring the needs of time for actual system model, for instance, modeling the event processing time and data transmission time on the communication network;

(5) *Data will be transmitted from one actor to another actor in the dataflow after the corresponding transition is enabled;*

(6) *The quantity of exchange data among actors in the data flow are defined with the token numbers.*

This definition gives the detailed behavior model of the HyperspaceFlow. By analyzing the behavior of the system, we can achieve the system-level performance estimation for smart space, such as power consumption, cost and user preference.

Definition 12: To guarantee the right design of distributed physical, data and human/flow system, it must satisfy the restrictions as follows:

(1) $\forall w \subseteq DW$ *in the data flow D,* $\exists$ *Mapping* $J : S_i \rightarrow 2^w$ *in the control flow, and* S_i *is unique. However, not all required control places must exist in the data flow graph;*

(2) *Assume that* S_l *is the set of sink places of the control flow, which is the set that followed nodes of control places is empty, and reachable marking set from* M *is indicated by* $R(M)$, $\forall S_i \in S, S_i \notin S_l, M(S_i) \leq 1$ *. In other words, the petri net must keep safe;*

(3) *If a place owns more than two transitions, it must not satisfy the condition that is true at the same time, i.e., the transition should happen deterministically and conflicts are not allowed.*

The definition can ensure the distributed physical, data and human/control flow system safety, conflict free, and meanwhile maximally keep petri net property so that we can conveniently use it to analyze and optimize actual system synthesis processes.

Definition 13: Given the distributed physical, data, human/control flow system $\vartheta = (P, D, Z, S, T, F, J, K, M_0)$, *the precedent relation* $\mapsto$ *of events in* ϑ *can be defined as follows:*

(1) $E_i \mapsto E_j$ *if and only if* $E_i \in A_i, E_j \in A_j, (A_i, A_j) \in DW$;

(2) *if* $E_i \mapsto E_j$, *and* $E_j \mapsto E_k$, *then* $E_i \mapsto E_k$;

(3) *if* $E_i \mapsto E_j$, *or* $E_j \mapsto E_i$, *they are sequential order;*

(4) E_i *and* E_j *are said to be concurrent if they are not in sequential order, where* $E_i \in E, E_j \in E$, *and* $E_i \neq E_j$.

Definition 14: The event structure of the distributed physical, data and human/control flow system ϑ *can be defined as three-tuple* $(E, \lessdot, \|)$, *where*

$E = \{E_1, E_2, \ldots, E_n\}$ *is the event set which consists of input event and output event of the actors;*

$\lessdot \subseteq E \times E$ *is a binary relation,* $E_i \lessdot E_j$ *denotes that event* E_i *happens before event* E_j, *and it must satisfy* $E_i \mapsto E_j$;

$\| \subseteq E \times E$ *is a concurrent relation,* $E_i \parallel E_j$ *represents that event* E_i *and* E_j *happen at the same time.*

Typical distributed system is a partial order structure, definition 13 and 14 provide the exact relation specifications about how to indicate the happen before. Also, sequence and concurrent are clearly described in formal symbol.

C. MAPPING PROCESS

1) PRELIMINARY

In this section, the mapping process, critical components and the model of the architecture platform are discussed.

The mapping process is to allocate the nodes into physical spaces and then to automatically synthesize the distributed execution platform and network that satisfy the needs of task computation and communication. Regarding to the former process, the object emplacement is performed to deposit the sensors, actuators and computation nodes into actual physical space, which is a location optimization procedure leveraging the IRM data. In essence, it is the mapping from the defined symbol to actual space coordinates. Furthermore, system synthesis and preference synthesis achieve the architecture mapping leveraging the IRM data at given platform library, for simplicity, the one to one mapping strategy is employed to proceed platform mapping. Also, they are considered as a whole and to be modeled as a multi-objective ILP problem.

Platform library is comprised with computation library, communication library and the position set of smart space. Computation library includes candidate computing platforms, which are divided into software and hardware platform, respectively. The former refers to the implementation via general purpose processors, the latter is typically the FPGA, ASIC, etc. Basically, both of them characterize with performance and cost, for example, as shown in TABLE 1.

TABLE 1. Computation platform library.

Platform index	Silicon Family Name	Category	Performance	Cost($)
1	Xilinx spartan 6	Hardware	Logic cell: 3840 RAM: 12Kb	199
2	Actel IGLOO2	Hardware	Logic cell: 6060 RAM:703Kb	421
3	Altera Cyclone III	Hardware	Logic cell: 5136 RAM:424Kb	249
4	ARM Cortex-A5	Software	Speed: 600MHz SRAM:1Mb	487
5	LPC4350	Software	Speed: 204MHz SDRAM: 64Mb	112
6	MPC5125	Software	Speed: 400MHz RAM: 256Mb	169

Communication library gives the performance and cost of network switch devices and communication interface for given communication technology. Take the wireless technology library as an example, usually there are two types of communication components for our system, one is the communication interface that transmits data for our task nodes, which does not have the ability of routing. The other is the routers that forward the data to guarantee the communication of the distributed system. The performance of each communication component is specified with delay, maximum bandwidth and sending power and receiving power of each bit. Particularly, it is similar with the library in [26]. Regarding to the space model, existing space model [27] can be mainly partitioned into twofold: physical and symbolic. The former employs the coordinates to represent the location

of object, while the latter employs the symbols to represent the logic position relations of the objects.

Here, we employ physical location based approach [27] to model the object physical position, meanwhile, utilize our proposed computation model as the symbolic model. Due to the physical restrictions of the house, such as the geometry of the house or personal preference, Usually, there exist limited position set that can be used to deposit the physical objects, computation devices and routers. Intuitively, they can be modeled as the point of Euclid Space, which is denoted with longitude, latitude and height. Fig. 2. gives an example of the candidate position set for nodes placement. As TABLE 2 presents, for each position in Fig. 2., corresponding coordinate type is the same as TABLE 2. All candidate locations are indexed and correspond to a unique coordinate. Besides, the white dots are the candidate physical object and computation node positions, and the black dots represent the candidate router positions, respectively.

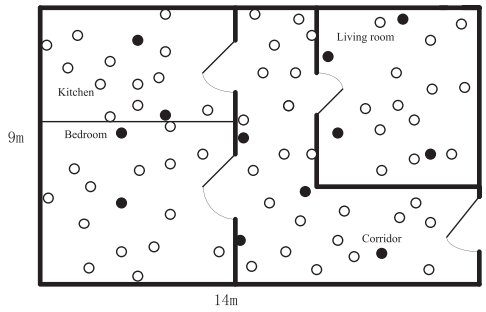


FIGURE 2. The floorplan of home environment. Black dots are the candidate location set of routers, white dots are the candidate location set of physical objects and computation nodes.

TABLE 2. Candidate position set.

Index	Position coordinate
1	(110,80,20)
2	(111,82,13)
...	(...,...,...)

User preference model plays a critical role in the smart space design. It severely influences user experience. We use quantitative model to represent user preference with respect to specific items. The items can consist of location, manufacturer, HCI manner, etc. Here, we employ the preference model mentioned in [28]. The user preference is divided into nine important levels, which correspond to a real value between -1 and 1.

2) OBJECT EMPLACEMENT

Based on the space model and the preference model, the physical objects in physical flow and the actors in data flow should be deposited into the space according to specific evaluation criteria. Object emplacement aims at assigning them into the space and guaranteeing the maximum user satisfaction. In essence, it is an ILP problem, since there are a large number of candidate locations, resulting in the explosion of location exploration space. To reduce the complexity of the expiration of location, we firstly seek to find the location of physical objects in candidate location set, then ensuring sensors and actuators are determined by physical object location, finally computation node is deposited by computing their distance with sensors and actuators.

To represent the objects in physical flow and data flow, we define a series of parameters as follows:
OBJ: the set of physical objects in physical flow;
SN: the set of sensors;
AC: the set of actuators;
CM: the set of computation nodes;
Their candidate location set is defined as L_{OBJ}, L_{SN}, L_{AC}, L_{CM}, respectively;
$Prefer_l^i$: the user preference of i-th location;
P_{ij}: is a binary variable, which denotes the i-th node assigned to j-th place.
Location exploration can be formulated as optimization problem as follows:

$$Max \sum_{i \in OBJ} \sum_{j \in L_{OBJ}} P_{ij} * Prefer_l^i. \tag{1}$$

Since each physical object can only be deposited into a location, it should satisfy the following constraint:

$$\sum_{j \in L_{OBJ}} P_{ij} = 1, \quad \forall i \in OBJ. \tag{2}$$

Therefore, we can get the coordinates set of physical objects by solving the ILP problem. Subsequently, sensors and actuators should be deposited closest with physical objects. Specifically, we use the approach in [26] to get one location set ρ of sensors and actuators. However, they are possibly not in our candidate location set. Thus, for each acquired coordinate, we choose the coordinate in candidate location set which is minimum Euclidean distance with the acquired coordinates, i.e,

$$\min_{v \in L_{SN} \cup L_{AC}} ||\psi_u - \psi_v||_2^2, \quad \forall u \in \rho; \tag{3}$$

where ψ_u, ψ_v denote the corresponding coordinates, $||\psi_u - \psi_v||_2^2$ is the Euclidean distance.

Hence, we can attain

$$P_{ij} = 1, \quad \forall i \in (SN \cup AC),\ j \in L_{SN} \cup L_{AC}; \tag{4}$$

Similarly, we further compute the coordinates of computation nodes according to their exchange tokens with sensors and actuators. i.e.,

$$P_{ij} = 1, \quad \forall i \in CM,\ j \in L_{CM}. \tag{5}$$

Accordingly, the locations of all objects in physical flow and data flow are deposited.

3) SYSTEM SYNTHESIS

The main goal of system synthesis is to achieve the assignment of computation and network resources of smart space. It incorporates computation and communication synthesis. Computation synthesis is to accomplish the software/hardware partition and to assign the tasks to distributed computation platforms. Here, considering the complexity of the model, the mapping procedure is exactly one to one, i.e., each actor in the data flow is exactly mapped into one processing element (PE). Hence, the schedule of various task assigned into a PE is beyond this paper. The actors in data flow whether realized by softwares or hardware, are determined by the needs of the task. Cost and power consumption are used as the criteria for the partition of software and hardware. Softwares are usually enabled through general purpose processors and feature with much better flexility and cost effectiveness to develop complicate algorithms, however, low performance and high power consumption. While hardwares exhibit better performance in worse case execution time and power efficient. However, there also exist huge developing cost and difficulty to realize complicated algorithms. Essentially, it is a trade-off for softwares and hardwares according to the cost and power. Therefore, we consider the software/hardware partition jointing with task assignment as an ILP problem.

Assuming that the data flow graph in section III-B is denoted as G, actors in G are divided into three classes of nodes, which are sensors, actuators and computation nodes. The design parameters are defined as follows:

N: the set of nodes in the data flow graph G; $N = SN \cup AC \cup CM$;

E: the set of arcs in G;

SW: the set of software computation devices, i.e, processors, corresponding to softwares of the computation library;

HW: the set of hardware devices, like FPGA, ASIC, etc, corresponding to hardwares of the computation library;

D: the set of computation devices, $D = SW \cup HW$.

Decision binary variables are defined as follows:

x_i: whether the node i in CM is enabled by software or hardwares, 1 softwares, 0 otherwise;

a_{ij}: denotes the i-th node in CM which is allocated to j-th computation devices.

To support the exploration of design space, the variables should satisfy a series of constraints, which are defined as follows:

$$\sum_{j \in D} a_{ij} = 1, \quad \forall i \in CM; \tag{6}$$

$$x_i = a_{ij}, \quad \forall i \in CM, \quad j \in SW; \tag{7}$$

$$x_i + a_{ij} = 1, \quad \forall i \in CM, \quad j \in HW; \tag{8}$$

$$a_{ij} * M_j \geqslant (m_i + RD_i + Cache_i) * a_{ij}, \quad \forall i \in CM,\ j \in SW; \tag{9}$$

$$a_{ij} * B_j \geq O_i * a_{ij}, \quad \forall i \in CM,\ j \in HW; \tag{10}$$

$$a_{ij} * MR_j \geqslant TR_i * a_{ij}, \quad \forall i \in CM,\ j \in HW. \tag{11}$$

Constraint (6) defines that each task node in G should be mapped into one computation devices. Constraints (7)(8) determine whether softwares or hardwares are in the exploration space according to variables x_i. Constraint (9) demonstrates that the memory of computation devices should be more than the required memory of actor i if it is enabled with softwares, where M_j is the memory of candidate platform, m_i is the program memory, RD_i is the data memory, $Cache_i$ is the network buffer storage required by actor i. When actors in G are realized by hardwares, they should satisfy constraints (10)(11). Constraint (10) indicates that logic cells of hardware devices should be more than the required numbers of actors, where B_j is the logic cell numbers of j-th hardware devices. O_i is the logic cells numbers to enable the actor i. Constraint (11) indicates that memory in hardware devices should be more than the required memory for storing configure data of actors, where MR_j is the memory of j-th hardware devices, TR_i is the memory requirement for actor i.

Objective function of computation synthesis refers to cost and power consumption, which will be given in later section and jointed the variables and constraints of communications synthesis.

The main goal of communication synthesis is to automatically synthesize a communication network, which can realize nearly optimal cost and power at given constraints. To guarantee the communication of nodes, we synthesize a network leveraging the exploration of design space in communication library. Here, we employ wireless network to exhibit the synthesis of communication. Specifically, considering the concise needs, we take the IEEE802.11a technology as an example to enable the synthesis.

To achieve the network synthesis, there are two aspects that need to be decided. On one hand, routing ability of network relies on routers. Thus, routers should be properly deposited into our candidate position set. The numbers and location of routers are determined by the quality of services for system, as well as cost and power consumption. On the other hand, for the topology, to lower the complexity of design, a cluster tree is employed for the network topology. Basically, the communication synthesis can be also considered as the ILP problem. The binary decision variables are defined as follows:

S_{ij}: the i-th location assigned to j-th type of routers;

W_{ij}: whether the node i and node j is linked, meanwhile j is the parent node;

Q_{ijc}: the end to end connection c uses the link between i and j;

SP_j: the j-th location is installed with the end device or router.

S_{ij} determines the placement of routers. Routers are assigned into the candidate positions to ensure the communication of nodes for their covering ranges. W_{ij} is used to start the link exploit to make a network. Q_{ijc} is used to assist the end to end routing computing.

Given the following specification parameters:

R: the set of routers;

L_R: the candidate location set of routers;

L_E: the end devices, $L_E = L_{SN} \cup L_{AC} \cup L_{CM}$;

L: the candidate location set of end devices and routers, $L_E = L_R \cup L_E$;

U_i: the worst case execution time for the node located in i-th position;
n_{max}: the maximum numbers of connections for each router;
R_{max}: the maximum numbers of routers for installing;
$Band_{ij}$: the bandwidth between location i and j;
$Band_c$: the required minimum bandwidth of connection c;
$Delay_{ij}$: the delay of the node located between i and j;
l_c: the required maximum delay of connection c;
DB_c: the maximum path loss of connection c;
$PLoss_{ij}$: the path loss of nodes located between i and j;
T_D: the designate deadline of connection c by user.

Assuming that there does not exist retransmission in the network, so the exploration of network design space is subject to the following constraints:

$$\sum_{j\in R} S_{ij} = 1, \quad \forall i \in L_R; \tag{12}$$

$$P_{ij} + SP_j \geqslant 2, \quad \forall i \in N, \quad j \in L_E; \tag{13}$$

$$S_{ij} + SP_i \geqslant 2, \quad \forall i \in L_R, \quad j \in R; \tag{14}$$

$$SP_j + SP_m \geqslant 2W_{jm}, \quad \forall j, m \in L,\ j \neq m; \tag{15}$$

$$W_{ij} + W_{ji} \leqslant 1, \quad \forall i, j \in L,\ i \neq j; \tag{16}$$

$$W_{ij} = 0, \quad \forall i \in L,\ j \in L_E,\ i \neq j; \tag{17}$$

$$\sum_{j\in L} W_{ij} \leqslant 1, \quad \forall i \in L,\ j \in L_R,\ i \neq j; \tag{18}$$

$$\sum_{i\in L} W_{ij} \leq n_{max}, \quad \forall j \in L_R,\ i \neq j; \tag{19}$$

$$W_{ij} + W_{ji} \geqslant Q_{ijc}, \quad \forall i, j \in L,\ c \in E,\ i \neq j; \tag{20}$$

$$W_{ij} + W_{ji} \geqslant Q_{jic}, \quad \forall i, j \in L,\ c \in E,\ j \neq i; \tag{21}$$

$$\sum_{i\in L_R}\sum_{j\in R} S_{ij} \leqslant R_{max}; \tag{22}$$

$$F * Q_p = C_p, \quad \forall p \in E; \tag{23}$$

$$min_{i,j\in L}(Q_{ijc} * Band_{ij}) \geqslant Band_c, \quad \forall c \in E,\ i \neq j; \tag{24}$$

$$\sum_{i,j\in L} Q_{ijc} * Delay_{ij} \leqslant l_c, \quad \forall c \in E,\ i, j \in L,\ i \neq j; \tag{25}$$

$$\sum_{i,j\in L} Q_{ijc} * PLoss_{ij} \leqslant DB_c, \quad \forall c \in E,\ i, j \in L,\ i \neq j; \tag{26}$$

$$\sum_{i,j\in L} Q_{ijc} * (U_i + Delay_{ij} + U_j) \leqslant T_D, \quad \forall c \in E,\ i, j \in L,\ i \neq j. \tag{27}$$

Constraint (12) requires that each candidate location should be exactly deposited a router. Constraints (13)(14) exhibit that whether the candidate locations are assigned with routers or computation devices. Constraint (15) shows that link can be built only if there are end devices or routers allocated between position j and position m. Constraint (16) illustrates that the tree topology is dissymmetry. Constraint (17) requires that end device can not be the parent node due that it does not own routing capability. Constraint (18) shows that each node has one parent at most. Maximum link numbers of each router are given by constraint (19). Constraints (20) (21) demonstrate that end to end connection c is routed if i and j are linked. Constraint (22) gives the maximum installing routers. Constraint (23) is the balance equation, which is used to compute the routing path of connection c. F is the incidence matrix where the row is the sited nodes, the column is the arcs that are instantiated. If k is the sited node and the source of arcs W_{ij}, $F(k) = 1$, else the destination $F(k) = -1$, otherwise it equals 0. Thus, the path of each connection C_p in E can be computed with this equation. Constraint (24) denotes that the minimum path bandwidth should be more than the requirement by the definition of the specification. Constraint (25) requires that the sum of path delay should be less than the required delay of connection c in E.

Constraint (26) is the sum of path loss of end to end connection c which should be less than required minimum path loss in the specification. Path loss [29], [30] is highly associated with the distance between nodes and physical attributes of buildings. Especially, the wall badly affects the indoor signal. Thus, we use the multi-wall model which is based on the work in [31]. Specifically, $PLoss_{ij} = l_0 + 10\gamma log(d(i,j)) + M_w$, where the l_0 is the path loss at 1 meter distance, γ is the path loss exponent, $d(i,j)$ is the distance between i and j, and M_w is the multi-wall model. Constraint (27) requires that the execution time of the task should be less than the deadline.

4) PREFERENCE SYNTHESIS

Preference synthesis aims to guarantee the maximum user satisfaction for the design of smart space. It can involve multiple aspects. Here, we consider device manufacturer and HCI interface of user preference. In real application scenarios, users usually own their preferences to software or hardware devices. Besides, users also exhibit different HCI manners, typically, textual, acoustic, visual, etc. By using the quantitative preference model mentioned in preliminary section, we can quantify the user satisfaction of smart space design. Hence, we define the parameters as follows:

τ_{ij}: binary variable, which denotes the i-th human object with j-th types of HCI in human flow, index j corresponds the j-th HCI manner.
$Prefer_H^i$: represents the user preference of the i-th type of HCI.
$Prefer_D^i$: represents the user preference of the i-th device.

The user preference estimation algorithm can be given in Algorithm 1. As the algorithm illustrates, user device preference and user HCI preference are considered as the design preference of smart space. $Prefer_c$ represents user satisfaction to generated device solution, $Prefer_n$ gives the satisfaction for HCI interaction manner. According to data flow, human flow in the IRM, we can calculate the total user preference value $Prefer_{total}$ to the design of smart space. Also, ω_1, ω_2 are the weight coefficient that can be determined by user needs.

Algorithm 1 User Preference Estimation Algorithm of the Smart Space Design

Input: The data flow D; The human flow H; The weight factor ω_1, ω_2; The set of parameters and variables;

Output: Total user preference value $Prefer_{total}$;

1: $Prefer_{total} = Prefer_c = Prefer_n = 0$;
2: **for** each actor $i \in D$ **do**
3: User device preference $Prefer_c = Prefer_c + a_{ij} * Prefer_D^i$;
4: **end for**
5: **for** each human object $i \in H$ **do**
6: User HCI preference $Prefer_n = Prefer_n + \tau_{ij} * Prefer_H^i$;
7: **end for**
8: $Prefer_{total} = \omega_1 * Prefer_c + \omega_2 * Prefer_n$
9: Return $Prefer_{total}$

5) EVALUATION FUNCTION

Beyond the preference synthesis, cost and power consumption are considered as the evaluation criteria for the system synthesis. They are given by control flow and data flow, and are used to guide the exploration of design space.

Given the following design parameters:

E_t^{ij}, E_r^{ij}: each bit transmission and receiving power from i to j;

E_p^{ij}: the processing power of node i on computation devices j;

E_O^{ij}: the overhead power of the connection between i and j;

b_c: the max bit numbers of connection c;

C_F^j: the j-th computation devices communication interface cost;

C_D^j: the j-th computation node cost;

C_R^i, C_R^j: the i-th router cost and j-th place installation cost;

C_S^i, C_A^j: the i-th sensor cost and j-th actuator cost;

C_H^i: the i-th type of HCI interface cost.

Firstly, we can compute the cost of the solution, which is defined by equation (28).

$$Cost_{total} = \sum_{\substack{i \in CM, \\ j \in SW \cup HW}} (C_F^j + C_D^j) a_{ij} + \sum_{j \in R, i \in L_R} (C_R^i + C_R^j) S_{ij} + \sum_{i \in SN} C_S^i + \sum_{j \in AC} C_A^j + \sum_{i \in H} C_H^i \quad (28)$$

Secondly, the power consumption estimation is given by control flow and data flow. Since the petri net can be used to model the behavior of the system, we can conduct the power analysis by state equation of control flow. However, the loop of control flow should be unrolled firstly by given rules for our system behavior analysis, specifically, we use the approach in [32]. General system-level power estimation algorithm is given in Algorithm 2.

As Algorithm 2 demonstrates, line 1-2 is to calculate the happen times of event that is defined in control flow leveraging state equation. Furthermore, line 4-14 proceeds the power consumption estimation, which consists of communication power given in line 8 and task processing power given in line 11. Specifically, as line 5-6 shows, it is enabled by exploiting the control place sign in data flow. Eventually, the total power consumption is the sum of event power consumption at specific time period.

Algorithm 2 Maximum Power Consumption Estimation Algorithm of the System

Input: The control flow C; The data flow D; The set of parameters and variables;

Output: Total power E_{total};

1: Solve the state equation for C, and get times of each event T_k in C;
2: Get the state numbers CN_k that corresponds to the T_k in step 1;
3: $E_{total} = E_c = E_n = 0$;
4: **for** each control place $k \in$ C **do**
5: Search the place C_k on the edge of D;
6: Get the associated vertexes of the arc, source V_s and destination V_d;
7: **for** each $i, j \in L$, $c \in (V_s, V_d)$ **do**
8: Communication power $E_c = E_c + CN_k * (\sum_{ij} Q_{ijc} * (E_t^{ij} + E_r^{ij} + E_O^{ij}) * b_c)$;
9: **end for**
10: **for** each $i \in \{V_s, V_d\}, j \in D$ **do**
11: Task processing power $E_n = E_n + CN_k * (a_{ij} * E_p^{ij})$;
12: **end for**
13: $E_{total} = E_{total} + E_c + E_n$
14: **end for**
15: Return E_{total}

The design of smart space is eventually abstracted as the solution of a co-synthesis problem. Essentially, it is summarized as a triple-objective ILP problem as follows:

$$min\{Cost_{total}, E_{total}, -Prefer_{total}\} \quad (29)$$

s.t. (2)-(27)

We solve the triple-objective optimization problem and get a pareto front as the candidate solution set of the design, then we choose suitable solutions according to the user expectation from the candidate solution set.

IV. CASE STUDY

A. HOME HEALTH CARE

Home health care is a emerging application of smart space. A well-known application scenario of home health care is fall detection. It offers the ability to automatically detect elders or kids fall and then send alarm or call the cellphone leveraging video sensors and actuators.

As the description in [33], the basic workflow of a health care in smart home is demonstrated as Fig. 3. Basically, the application can be partitioned into five tasks or subsystems, which are described as follows:

(1) Segmentation: this task aims to preprocess the image data which is sampled from the synchronous video sensors (cameras), and to gain the foreground field.

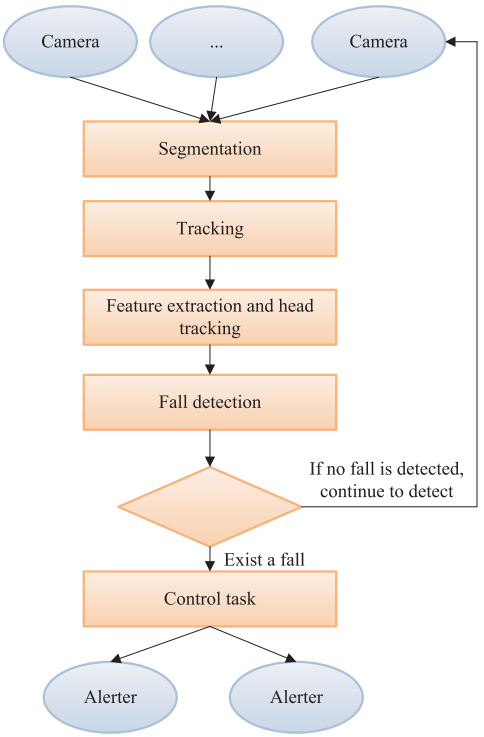


FIGURE 3. The flowchart of fall detection in smart home.

(2) Tracing: the task traces the objects of the screen and marks them as signs of human and non-human.
(3) Feature extraction and head tracking: regarding to each object marked as human, capturing its direction features and variance ratio based on the views of video sensor. Also, the head location is able to be tracked according to the skin color.
(4) Fall detection: based on the feature extraction from step 3, fall detection algorithm can make a judgement for the fall activity.
(5) Control task: the control node aims to send the affirmation to the user, and when the user does not give a response or time out, then sending the command to the actuators for alarming or calling emergency phone.

Therefore, the holistic application is able to be divided into five types of task nodes. Besides, physical objects consist of cameras and alerters. Sensors are equipped into the cameras and actuators are used to operate alerters.

B. MODELING BY HyperspaceFlow

In our smart home scenario, as the floorplan given by Fig. 2., a house covers 9m $\times$ 14m squares and is lived with an elder, who needs the health care by fall detection system. The house consists of kitchen, bedroom, living room as well as corridor. To achieve the home health care of the elder, a series of video sensors, alerters, actuators and distributed computing nodes should be deposited into these spaces and coordinates to finish the monitoring task.

To realize this application of smart space, we employ the specification given by extending the language defined in [16] as a function specification, which is a state based language for describing smart space. HyperspaceFlow model can capture the function of the distributed system as control flow, physical flow, data flow as well as human flow, demonstrated in Fig. 4.-Fig. 7.

Physical flow is demonstrated in Fig. 5., C1-C16 represent the number of cameras, each of which transmits the images

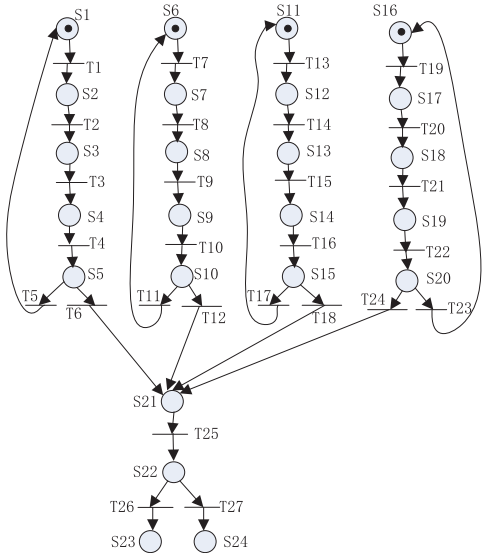


FIGURE 4. Control flow of the home health care.

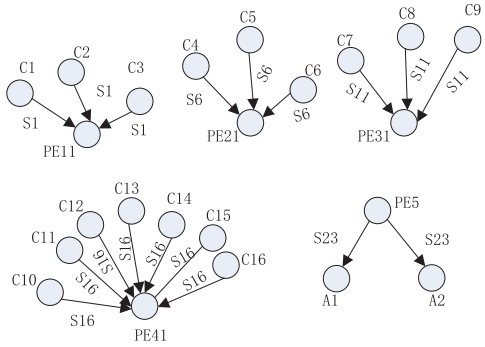


FIGURE 5. Physical flow of the home health care.

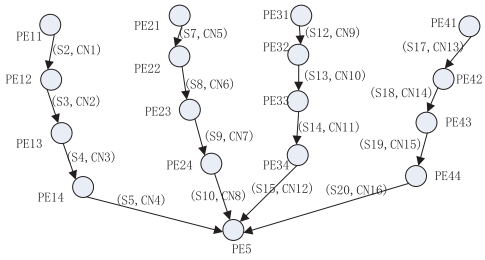


FIGURE 6. Data flow of the home health care.

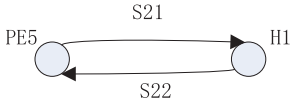


FIGURE 7. Human flow of the home health care.

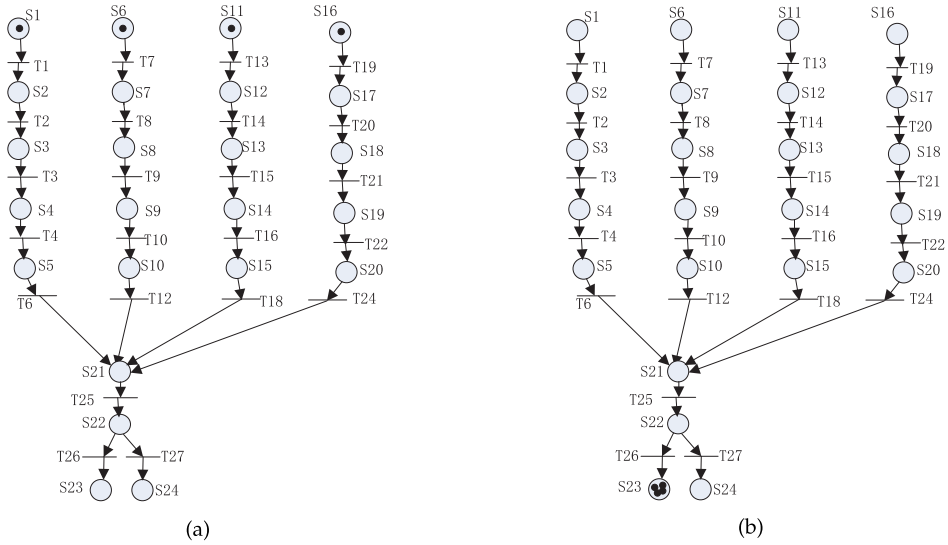


FIGURE 8. The loop of control flow is unrolled for power consumption estimation. (a) denotes the initial state M_0; (b) denotes the final state M.

to an actor when the control signal is enabled in control flow. A1 and A2 denote the alerters, which are triggered by the actor when the control signal is enabled. Each PE within physical flow corresponds to the actor in data flow.

Data flow is clearly presented in Fig. 6. The actors in data flow are computation nodes. Specifically, the fall detection system is comprised with four sub tasks, which correspond to the monitor of kitchen, living room, bedroom and corridor. According to the specification, for kitchen, living room and bedroom, each of them is deposited with three cameras. Besides, corridor is assigned with seven cameras. Furthermore, the $PE_{ij}(i = 1, 2, 3, 4, j = 1, 2, 3, 4)$ denotes i-th space with j-th type of task, $i = 1, 2, 3, 4$ corresponds to kitchen, living room, bedroom, corridor respectively. The types $j = 1, 2, 3, 4$ are given by section IV-A with respect to (1)-(4). $PE5$ is the central control node that gives commands for actuators. Two actuators are allocated into corridor, which are used to send alarm. Also, each of the arcs in data flow is associated with a control place given by control flow and the exchange token numbers $CN_i(i = 1, \ldots, 16)$ among actors. In essence, data flow is determined by control flow, when the fall event occurs, central control node will send control signals and trigger the control command to the actuators.

Human flow in Fig. 7. depicts the HCI. The actor PE5 will send the affirmation to the user when the fall is detected. H1 indicates the HCI interface. The interaction process is also controlled by the control flow. Here, HCI inteface can be the textual, acoustic, visual or hybrid manner.

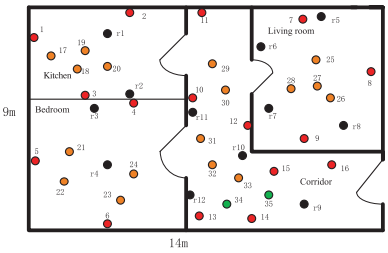
For control flow in Fig. 4., control places are used to indicate the control signals or messages for the actors in physical flow, data flow and human flow, while transitions are the output events created by the actors. Specifically, S1, S6, S11 and S16 are associated with the camera sampling tasks for kitchen, living room, bedroom and corridor, which are the loops due to their period attribute. Each of them is given an initial marking, actually, they possibly own various sampling period. Regarding to the task of monitoring kitchen, T1 denotes the transmission event of preprocessing data; T2 represents the transmission event of objects tracking data; T3 indicates the event of feature extraction and head tracking; T4 gives the event of fall detection. If fall is detected then trigger event T6, else trigger event T5. Corresponding control signals are illustrated with S2 to S6. Like the kitchen, control flow in rest three spaces is presented in Fig. 4. Any event from T6, T12, T18 and T24 can result in a control signal to the central control node in physical flow, furthermore, triggering the event T25, which exhibits that the elder is currently under danger, and then the actor will send the affirmation to the user. T27 represents that the user is out of danger by user responding and then transits into safe state S24, else T26 is triggered due that the user does not respond or time out then control signal S23 gives operations to actuators, which make a response to this fall event. Besides, for conciseness, we do not give the time label on T-elements. The time interval of transitions finish corresponds to the event transmission time among the actors.

Due that the health care system is a distributed application and consists of various sampling tasks, control flow in Fig. 4. includes multiple loops. Here, we choose the least common period of them. Hence, control flow is further conversed into Fig. 8. The maximum power estimation is conducted within this period. The critical step is to calculate each event happen times in this period leveraging the state equation of control flow. Then we start with a system-level power estimation including communication power and task processing power.

As Fig. 8 demonstrates, since it is reachable from initial M_0 to final M, where M_0 is the initial marking that is equal to $(1_1, 0, \ldots, 1_6, 0, \ldots, 0, \ldots, 1_{11}, 0, \ldots, 1_{16}, \ldots, 0)^T$ in Fig. 8(a), which denotes the marking of S1, S6, S11, S16 equal to 1, the rest are equal to 0. Assuming that each time can

Index	Actor	Index	Actor	Index	Actor	Index	Actor	Index	Actor
1	C1	8	C8	15	C15	22	PE21	29	PE42
2	C2	9	C9	16	C16	23	PE23	30	PE43
3	C3	10	C10	17	PE11	24	PE24	31	PE41
4	C4	11	C11	18	PE12	25	PE34	32	PE44
5	C5	12	C12	19	PE13	26	PE32	33	PE5
6	C6	13	C13	20	PE14	27	PE31	34	A1
7	C7	14	C14	21	PE22	28	PE33	35	A2

(a)



(b)

FIGURE 9. The result of object emplacement, (a) indicates the actors indexes, (b) indicates the locations of objects, the red dots denote cameras, the green dots denote alerters, the orange dots denote computation nodes and the black dots denote the wireless routers.

detect the fall event, thus, final state M depicted in Fig. 8(b) is equal $(0, 0, \ldots, 4_{23}, 0)^T$, which denotes S23 equal to 4, the rest are 0. Both of them are 24 dimension column vector. A is the incidence matrix of petri net which can be computed according to the approach in [32]. Hence, the state equation of Fig. 8 is given as follows:

$$M = M_0 + A^T X \tag{30}$$

Furthermore, we can get the event happen times vector X, which is used for power consumption estimation. We use the Algorithm 1 to estimate the maximum power consumption, and Algorithm 2 to estimate the user preference with respect to device and HCI interface. Thus, we solve the triple-objective ILP problem and can get a pareto front of the design. Detailed simulation results for home health care will be discussed in the next section.

V. SIMULATION RESULTS

In this section, we focus on using the proposed methodology to explore the architecture of the home health care. Simulation is conducted within the smart home environment.

Firstly, object emplacement is performed with the proposed approach. Cameras and alerters are firstly deposited with solving the user location preference optimization problem in section III-C.2. Then the coordinates of computation nodes are calculated by the approach in section III-C.2. Besides, the locations of routers are determined by communication synthesis. To reduce the complexity of distance calculation, we employ 2D coordinates to simulate the distance among the nodes. For the given candidate location set in Fig. 2., we employ SAT4J [34] to solve the ILP problem, then the nodes allocation result is illustrated in Fig. 9(b). and the maximum user location preference value is 8.0.

Secondly, system synthesis is performed according to the approach in section III-C.3. The procedure encompasses computation synthesis and communication synthesis. For the computation synthesis, candidate implementation platforms of the home health care are selected from the computation platform library, which consists of six computation platforms with different manufacturers. To conduct our simulation, we employ three FPGA based platforms and three general purpose processor based platforms. Candidate hardware implementation platforms include Xilinx spartan 6 [35], Actel IGLOO2 [36], Altera Cyclone III [37]. ARM Cortex-A5 [38], LPC4350 [39] and MPC5125 [40] are used as candidate general processor platforms. The price data and power consumption data in TABLE 1 can be attained from their data sheets. The parameters of computation synthesis are given in the specification defined in section IV-B.

To proceed the communication synthesis, IEEE 802.11a based wireless network is employed to show the communication synthesis approach. At the basis of the first step, the actors are linked with routers, specifically, each location of the routers is numbered with letter r and indicated by black dots in Fig. 9(b). Their numbers are determined by the system synthesis results. Meanwhile, each dot in Fig. 9(b). has a unique coordinate corresponding to the space model in our simulation environments.

Also according to the QoS requirements of the home health care, the constraint parameters of the objection function are specified, $n_{max} = 20, R_{max} = 12, Band_c = 2Mb$, $l_c = 2sec, DB_c = 76db$. Regarding to the deadline, T_D from the sensors to actuators should be less than 1.8sec.

Thirdly, for the preference synthesis, four types of HCI interface are considered: acoustic, visual, textual or hybrid interaction manner. The user preferences for devices and HCI interface are collected with natural language, then each of them can be quantified as a preference value between -1 and 1 according to the approach mentioned in the preliminary section.

Aim at getting the final solution, we employ NSGA-II [41], which is a well-known multi-objective optimization algorithm to solve the triple-objective optimization problem. Subsequently, we use the available optimization NSGA-II tool MOEA framework [42] as the optimization solver. Concretely, we implement the problem as the module of the framework. The NSGA-II running parameters are given as follows:

- population size=2000
- mutation probability=0.000001
- bit crossover probability=0.9

In Fig. 10., Pareto front is depicted as attainted by plotting 492 non-dominated solutions that are found by NSGA-II. Power consumption is represented as the power

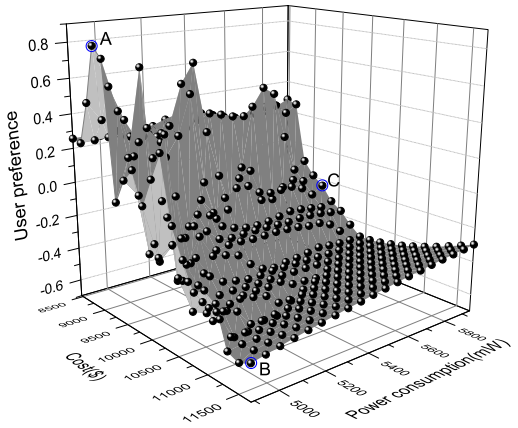


FIGURE 10. **The curved surface denotes the Pareto front with Pareto optimal points generated by design space exploration. The upper points on the curved surface are dominated by the Pareto points on the curved surface. Point A, B and C are chosen solutions to be compared.**

loss at common least cycle of the fall detection application. Cost represents the total price for realizing the application. The preference value of user is denoted as the opposite numbers of the attained solutions. The final solution is the trade-off among power consumption, cost and user preference value. It is a nearly optimal or relatively satisfactory solution. Also the Fig. 10. demonstrates that the solutions at the bottom of the curved surface own much better performance and the points at the upper of the surface have relatively poor performance.

TABLE 3. **Computation synthesis result of point A.**

Actor index	Platform index	HCI interface manner
17,18,21,22,25,26,29,30	1	Textual
19,23,24,27,32	2	
31	3	
-	4	
20	5	
28,33	6	

We choose point A, B, C as examples to investigate our generated design solutions. The generated system configurations are represented as computational and communication solutions and HCI interface. TABLE 3 and Fig. 11(a). show the solution of point A, which depicts the selected computation platform, network link and HCI interface manner. In TABLE 3, actor index indicates the number in the Fig. 9a. Platform index corresponds to the computation platform library index in TABLE 1. The HCI interface manner is generated by the preference synthesis. Besides, Fig. 11(a). elucidates the interconnections of the wireless network. It consists of the connection between end devices and wireless routers. The directions for the arrows demonstrate that the nodes target to their parent nodes. The communication solution is automatically generated by searching in the design space. Besides, TABLE 4 and Fig. 11(b)., TABLE 5 and Fig. 11(c). illustrate the solution of point B and point C. As a result, point A can achieve much lower cost and power efficient, however it owns lower user satisfaction quantified with user preference value, point B is power efficient and much higher user satisfaction than A. However, it requires much higher cost. Point C means low cost and high user satisfaction, but the power consumption is relatively high. In fact, for real scenarios, users is able to further select a fit solution by set a threshold range for these criteria.

TABLE 4. **Computation synthesis result of point B.**

Actor index	Platform index	HCI interface manner
17,18,20,21,22,24,25, 26,28,29,30,31,32	1	Hybrid
19,23,27	2	
-	3	
-	4	
-	5	
33	6	

TABLE 5. **Computation synthesis result of point C.**

Actor index	Platform index	HCI interface manner
17,18,21,22,24,25,26,29,30	1	Acoustic
19,23,27,31	2	
-	3	
28,32	4	
-	5	
20,33	6	

To compare our design method with manual design, we simulate the manual design of the health care case in the same library based on monte carlo method [43]. As Fig. 12. shows,

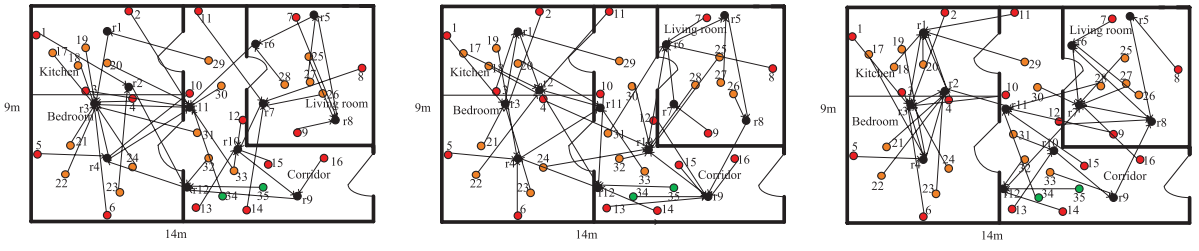


FIGURE 11. **The selected solution of communication synthesis from Fig. 10, (a) denotes point A, (b) denotes point B, (c) denotes point C.**

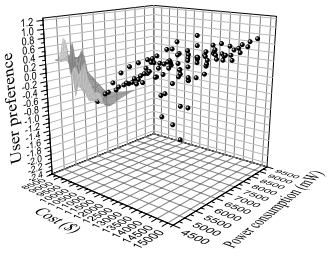


FIGURE 12. The grey curved surface denotes Pareto front got by our proposed approach, and the black dots represent manual design solutions.

the performance of design solutions, which are generated by our approach, are much better than the manual design under most conditions. In essence, the reason is that the manual design seriously relies on the experience of designers, leading to the result that manual design can not achieve the balance of power consumption, cost as well as user preference. Besides, designers hardly consider the design holistically, which results in the loss of the design information. The proposed approach can take computation, communication, user impact and space model into a systematic manner to consider, eventually accomplish the design optimization.

VI. CONCLUSION

In this article, we propose a system-level design methodology for smart space. The application specification of smart space can be further transformed into underlying architecture leveraging our proposed IRM-HyperspaceFlow, meanwhile, meeting the demands of cost, power consumption and user preference optimization. The design solution can be automatically generated and it consists of computation solution, communication solution as well as the HCI interface selection. The health care case is used to elaborate the feasibility and effectiveness of the proposed method. We simulate the smart home environment and eventually yield the design solution.

Future works need to be focused on two aspects: on one hand, our design methodology only considers single user modeling in smart space. The social interaction among users within smart space should be considered as another critical design aspect. On the other hand, heterogenous wireless network synthesis will be developed to tailor for more complicated communication scenarios design of smart space.

REFERENCES

[1] M. Weiser, "The computer for the 21st century," *Sci. Amer.*, vol. 265, no. 3, pp. 94–104, 1991.

[2] J. Ma, L. T. Yang, B. O. Apduhan, R. Huang, L. Barolli, and M. Takizawa, "Towards a smart world and ubiquitous intelligence: A walkthrough from smart things to smart hyperspaces and UbicKids," *Int. J. Pervasive Comput. Commun.*, vol. 1, no. 1, pp. 53–68, 2005.

[3] *Cybermatics*. [Online]. Available: http://www.cybermatics.org, acceessed Nov. 15, 2015.

[4] J. Ma *et al.*, "Cybermatics: A holistic field for systematic study of cyber-enabled new worlds," *IEEE Access*, vol. 3, pp. 2270–2280.

[5] H. Ning, H. Liu, J. Ma, L. T. Yang, and R. Huang, "Cybermatics: Cyber–physical–social–thinking hyperspace based science and technology," *Future Generat. Comput. Syst.*, 2015, doi: 10.1016/j.future.2015.07.012.

[6] K. Keutzer, A. R. Newton, J. M. Rabaey, and A. Sangiovanni-Vincentelli, "System-level design: Orthogonalization of concerns and platform-based design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 12, pp. 1523–1543, Dec. 2000.

[7] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, "The Gator Tech Smart House: A programmable pervasive space," *Computer*, vol. 38, no. 3, pp. 50–60, Mar. 2005.

[8] D. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan, "CASAS: A smart home in a box," *Computer*, vol. 46, no. 7, pp. 62–69, Jul. 2013.

[9] *Cartel*. [Online]. Available: http://cartel.csail.mit.edu/doku.php/, accessed Oct. 9, 2014.

[10] P. Zhou, Y. Zheng, and M. Li, "How long to wait? Predicting bus arrival time with mobile phone based participatory sensing," in *Proc. 10th ACM Int. Conf. Mobile Syst., Appl., Services*, 2012, pp. 379–392.

[11] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "T-drive: Enhancing driving directions with taxi drivers' intelligence," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 220–232, Jan. 2013.

[12] A. Ghose, P. Sinha, C. Bhaumik, A. Sinha, A. Agrawal, and A. D. Choudhury, "UbiHeld: Ubiquitous healthcare monitoring system for elderly and chronic patients," in *Proc. ACM Conf. Pervasive Ubiquitous Comput. Adjunct Publication (UbiComp Adjunct)*, 2013, pp. 1255–1264.

[13] C.-W. Jeong, S.-C. Joo, and Y. S. Jeong, "Sleeping situation monitoring system in ubiquitous environments," *Pers. Ubiquitous Comput.*, vol. 17, no. 7, pp. 1357–1364, Oct. 2013.

[14] R. Huang, X. Zhao, and J. Ma, "The contours of a human individual model based empathetic u-pillbox system for humanistic geriatric healthcare," *Future Generat. Comput. Syst.*, vol. 37, pp. 404–416, Jul. 2014.

[15] F. Vergari *et al.*, "A smart space application to dynamically relate medical and environmental information," in *Proc. Conf. Design, Autom. Test Eur. (DATE)*, Mar. 2010, pp. 1542–1547.

[16] U. Bischoff and G. Kortuem, "A compiler for the smart space," in *Proc. Eur. Conf. Ambient Intell. (AmI)*, vol. 4794. 2007, pp. 230–247.

[17] W. Allègre, T. Burger, and P. Berruet, "Model-driven flow for assistive home automation system design," in *Proc. 18th IFAC World Congr.*, 2011, pp. 6466–6471.

[18] A. Sangiovanni-Vincentelli, "Quo vadis, SLD? Reasoning about the trends and challenges of system level design," *Proc. IEEE*, vol. 95, no. 3, pp. 467–506, Mar. 2007.

[19] *Petri Net Markup Language*. [Online]. Available: http://www.pnml.org, accessed Dec. 9, 2014.

[20] S. Das and B. K. Panigrahi, "Multi-objective evolutionary algorithms," *Encyclopedia Artif. Intell.*, vol. 3, pp. 1145–1151, Nov. 2009.

[21] Z. Peng and K. Kuchcinski, "Automated transformation of algorithms into register-transfer level implementations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 13, no. 2, pp. 150–166, Feb. 1994.

[22] C. Hewitt, P. Bishop, and R. Steiger, "A universal modular ACTOR formalism for artificial intelligence," in *Proc. 3rd Int. Joint Conf. Artif. Intell.*, 1973, pp. 235–245.

[23] R. B. Ortega and G. Borriello, "Communication synthesis for distributed embedded systems," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 1998, pp. 437–444.

[24] C. Ramchandani, "Analysis of asynchronous concurrent systems by timed Petri nets," Ph.D. dissertation, Dept. Elect. Eng., Massachusetts Inst. Technol., Cambridge, MA, USA, 1974.

[25] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, no. 7, pp. 558–565, Jul. 1978.

[26] F. Leonardi, A. Pinto, and L. P. Carloni, "Synthesis of distributed execution platforms for cyber-physical systems with applications to high-performance buildings," in *Proc. IEEE/ACM Int. Conf. Cyber-Phys. Syst. (ICCPS)*, Apr. 2011, pp. 215–224.

[27] C. Becker and F. Dürr, "On location models for ubiquitous computing," *Pers. Ubiquitous Comput.*, vol. 9, no. 1, pp. 20–31, Jan. 2005.

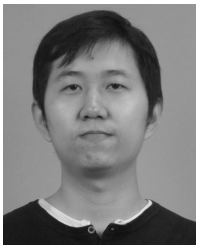
[28] H. Mukhtar, D. Belaïd, and G. Bernard, "Dynamic user task composition based on user preferences," *ACM Trans. Auto. Adapt. Syst.*, vol. 6, no. 1, p. 4, Feb. 2011.

[29] A. Bose and C. H. Foh, "A practical path loss model for indoor WiFi positioning enhancement," in *Proc. 6th Int. Conf. Inf., Commun. Signal Process.*, Dec. 2007, pp. 1–5.

[30] U. Naik and V. N. Bapat, "Adaptive empirical path loss prediction models for indoor WLAN," *Wireless Pers. Commun.*, vol. 79, no. 2, pp. 1003–1016, Nov. 2014.

[31] F. Capulli, C. Monti, M. Vari, and F. Mazzenga, "Path loss models for IEEE 802.11a wireless local area networks," in *Proc. 3rd IEEE Int. Symp. Wireless Commun. Syst. (ISWCS)*, Sep. 2006, pp. 621–624.

[32] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.
[33] L. Hazelhoff, J. Han, and P. H. N. de With, "Video-based fall detection in the home using principal component analysis," in *Advanced Concepts for Intelligent Vision Systems*. New York, NY, USA: Springer-Verlag, 2008, pp. 298–309.
[34] *Sat4j*. [Online]. Available: http://www.sat4j.org/, accessed Nov. 10, 2014.
[35] *Xilinx Datasheet*. [Online]. Available: http://www.xilinx.com
[36] *IGLOO2 Datasheet*. [Online]. Available: http://www.microsemi.com/products/fpga-soc/fpga-and-soc
[37] *Cyclone Datasheet*. [Online]. Available: http://www.altera.com/devices/fpga
[38] *Cortex-a5 Datasheet*. [Online]. Available: http://www.arm.com/zh/products/processors/cortex-a/cortex-a5.php
[39] *LPC4350 Datasheet*. [Online]. Available: http://www.nxp.com
[40] *MPC5125 Datasheet*. [Online]. Available: http://www.freescale.com, accessed Nov. 20, 2014.
[41] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[42] *MOEA Framework*. [Online]. Available: http://www.moeaframework.org, accessed Nov. 20, 2014.
[43] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo Method*, vol. 707. New York, NY, USA: Wiley, 2011.



JING ZENG received the B.Sc. degree in information and computing science from the Hubei University of Technology, Wuhan, China. He is currently pursuing the Ph.D. degree in computer architecture with the Huazhong University of Science and Technology. His research interests include ubiquitous computing, system-level design methodology, smart space, and embedded systems.

LAURENCE T. YANG received the B.E. degree in computer science and technology from Tsinghua University, China, and the Ph.D. degree in computer science from the University of Victoria, Canada. His research has been supported by the National Sciences and Engineering Research Council, and the Canada Foundation for Innovation. He is currently with the Department of Mathematics, Statistics and Computer Science, St. Francis Xavier University, Canada. His research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing, and big data.

JIANHUA MA received the Ph.D. degree in information and electronics engineering from Xidian University. He is currently a Professor and the Head of the Department of Digital Media with the Faculty of Computer and Information Sciences, Hosei University, Japan. His research interests include multimedia, ubiquitous computing, social computing, and cyber intelligence.

MINYI GUO received the Ph.D. degree in computer science from the University of Tsukuba, Japan. Before 2000, he had been a Research Scientist with NEC Corporation, Japan, and a Professor with the School of Computer Science and Engineering, The University of Aizu, Japan. He is currently a Distinguished Chair Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China, and an Adjunct Professor with The University of Aizu. His research interests include parallel and distributed processing, parallelizing compilers, pervasive computing, embedded systems software optimization, and software engineering.