Automated Planning for Ubiquitous Computing

ILCHE GEORGIEVSKI and MARCO AIELLO, University of Groningen

The goal of ubiquitous computing is to create ambience in which one's experiences and quality of life are improved by monitoring and assisting people using ubiquitous technologies and computation in coherence. The continuous advancements of involved technologies, such as wireless communications, mobile devices, and sensors, imply fast evolution of ubiquitous computing environments too. The complexity of these environments is reaching a point where traditional solutions simply no longer work. The environments are in need of computational techniques that can deal with the evolution and uncertainty of ubiquitous computing environments dynamically and automatically. Artificial Intelligence (AI) can contribute towards satisfying this future scenario in many ways, while numerous approaches inspired by work in the AI planning community have already been designed for ubiquitous computing. We devote this study to investigate the current progress of AI planning for ubiquitous computing by analysing those approaches. We rigorously search for and select relevant literature out of which we extract qualitative information. Using the extracted qualities, we derive a generic framework that consists of aspects important to planning for ubiquitous computing. The framework's main purpose is to facilitate the understanding of those aspects, and classify the literature according to them. We then analyse the literature in a consolidated way, and identify future challenges of planning for ubiquitous computing.

CCS Concepts: • **General and reference → Surveys and overviews**; • **Human-centered computing → Ubiquitous and mobile computing**; • **Computing methodologies → Planning and scheduling**; *Knowledge representation and reasoning*; Intelligent agents;

Additional Key Words and Phrases: Planning domain modelling, planning techniques, pervasive computing, smart environments, ambient intelligence

ACM Reference Format:
Ilche Georgievski and Marco Aiello. 2016. Automated planning for ubiquitous computing. ACM Comput. Surv. 49, 4, Article 63 (December 2016), 46 pages.
DOI: http://dx.doi.org/10.1145/3004294

1. INTRODUCTION

Ubiquitous computing tends towards revolutionising the way we live in terms of comfort, assistance, and safety by cooperatively utilising diverse technologies and various forms of computation to monitor and assist us. While wireless communications, mobile devices, and sensors are prominent examples of the rapid technological progress, the development of computational techniques is yet a challenge. Ubiquitous computing environments are often computationally enabled by predefining sequences of device actions that are usually executed under predetermined conditions and in well-known situations. Such approaches are shown to be too limited to handle the dynamics and uncertainty of these environments, to deal with a large number of environment

This work is supported by the Dutch National Research Council Beijing Groningen Smart Energy Cities Project, contract no. 467-14-037.
Authors' addresses: I. Georgievski and M. Aiello, Johann Bernoulli Institute for Mathematics and Computer Science, University of Groningen, Nijenborgh 9, 9747AG Groningen, The Netherlands; emails: {i.georgievski, m.aiello}@rug.nl.

DOI: http://dx.doi.org/10.1145/3004294

conditions, to support the needs of those populating the environments, and even to achieve some global environment objectives, such as energy saving. Let us illustrate these issues concretely in the following adventurous scenarios.

Theodore, a writer, bought a new home that has several rooms. While each room is equipped with various home appliances, such as a TV in the living room, the home has recently been enriched with numerous unobtrusive devices, such as movement sensors, door actuators, gas-leakage sensors, etc. Theodore also bought Tars, a domestic robot that can sense human presence, clean rooms, move around the home, pick up and drop items, and help and support Theodore. This home of Theodore is precisely an example of a ubiquitous computing environment [Weiser 1999].

Theodore now purchases a system to handle his requests, anticipate his activities, coordinate all devices and appliances, cooperate with Tars, and take care of the home. The system he obtains is named Samantha.[1] Suppose Samantha has a large set of rules for triggering device actions predefined for typical situations in homes. For example, Samantha can instruct Tars to clean rooms from dust in such a way that he does not disturb Theodore. In reality, the needs of Theodore are more personalised and involve spatial and temporal properties for which Samantha may fail short of rules. For example, Theodore wants the kitchen and bedroom clean with some items to be delivered to him at the same time. Samantha has to be able to come up with the best plan that satisfies this objective.

One day Theodore decides to prepare lunch, and while he chooses the dish from the menu, Samantha takes the dish's recipe and selects plans for Theodore to follow. During the selection, Samantha finds out that an ingredient is missing that cannot be replaced with any of the available ones in the kitchen. Samantha runs out of options, though a solution exists. For example, Tars can go to the storage room and get the missing ingredient. In this way, Theodore can still cook his dish. Suppose Theodore gets the ingredient eventually, and at some point during cooking, Samantha detects a dangerous situation, a gas leak. She triggers a predefined goal for such situations, and selects a safety plan consisting of instructions for Theodore to leave the home, and actions to close all doors leading to the kitchen so as to isolate spreading of the gas as much as possible. At the same time, Samantha issues actions for pulling up window blinds and opening the window in the kitchen. However, it happens that the blinds are stuck, which prevent the window from opening. Samantha finds herself in an unexpected situation for which there are no predefined plans. Uncertain situations such as these are rules rather than exceptions in ubiquitous computing environments.

Ubiquitous computing is therefore in need of techniques that go beyond predefined solutions, and act intelligently and autonomously. The field of Artificial Intelligence (AI) focuses on developing highly flexible and effective systems for intelligent behaviour, where AI planning or automated planning provides means for automated and dynamic creation of plans [Ghallab et al. 2004]. Planning requires a goal, an initial state of an environment, and some knowledge to select and combine courses of actions that, when executed, achieve the goal.

Planning is recognised as a central technique to achieve intelligent behaviour of ubiquitous computing environments by virtue of its general capabilities to search in a large space of possible solutions to a given planning problem, reason about space, time, and resources, address dynamism and uncertainty, support the heterogeneity of constituents of environments, and support modelling of knowledge with reasonably expressive constructs. The result is numerous studies that aim to use planning in different types of ubiquitous computing environments. Yet when it comes to characterising

[1]Theodore, Samantha and Tars are inspired by the eponymous characters in the films "Her" [Phoenix and Johansson 2013] and "Interstellar" [Irwin 2014].

what exactly the current research of automated planning for ubiquitous computing consists of, and whether and how it might proceed in the future, there is no analysis in the literature. Existing efforts in using planning for ubiquitous computing differ in too many extents and have unclear premises in relation to the questions of what kind of planning problems are being solved, how problems are modelled, how planning systems are actually designed, and how those systems are used and evaluated in ubiquitous computing. In light of these ambiguities, it seems prudent to methodologically analyse and improve this situation. On the one hand, a well-developed analysis could produce means that can be used to support designs and developments of future ubiquitous computing systems. On the other hand, analysing planning from the perspective of ubiquitous computing may raise challenges different than those found in the discerning trends in the general progress of automated planning.

We aim to look into these ambiguities by analysing a full range of relevant literature. Our contributions are listed next.

—We develop a rigorous methodology to acquire and select existing relevant literature and extract qualitative information from the literature.
—We derive a generic framework for planning for ubiquitous computing from qualitative information. The framework consists of a set of dimensions each defining a particular aspect. The framework can help ubiquitous computing developers make decisions regarding designing and implementing systems based on planning, and future efforts to orient their research in an appropriate direction.
—We classify the selected studies according to their spectrum of supported dimensions, and analyse each of these dimensions as invoked in the studies.
—We derive the limitations of the current progress in planning for ubiquitous computing, aiming at improved perspectives for both ubiquitous computing and planning.

The remainder of the article is organised as follows. Section 2 briefly introduces automated planning and our methodology. Section 3 presents the selected relevant studies and the basic structure of our framework. The next three sections gradually develop the framework into more details by defining its dimensions, and classifying and analysing the relevant studies according to those dimensions. Section 7 provides the possible directions for future research. Section 8 finalises the article with concluding remarks. Appendix B presents examples of planning encodings from Theodore's home, and Appendix C lists planning systems used in the selected relevant literature. The online Appendix A presents our methodology in detail and provides some insights into the year, type, and venue of publication of selected literature.

2. PRELIMINARIES

The task of a planner is to compute a plan as one possible solution to a given planning problem. The plan computed consists of actions that are provided to the planner for each domain of interest. Each of the plan actions is then executed into the world. The following section briefly introduces the meaning of these terms and how they relate to each other. After this introduction, we turn to the methodology developed to rigorously search, classify, and analyse relevant literature. For simplicity, we introduce only briefly the basic steps of the methodology, while we fully account for it in Appendix A.

2.1. Automated Planning

A *planning problem* consists of an initial state, a goal state, and a set of actions. The *initial state* describes how the world is when the planner is invoked. The *goal state* or goal describes how one wants the world to be after some plan is executed. The world in which planning takes place is called *a planning domain*. In our case, the planning domain can be any ubiquitous computing environment. We henceforth refer to world states as (ubiquitous computing) environment states.

To define actions, we use the terminology of the Planning Domain Definition Language (PDDL) [McDermott et al. 1998]. *Actions* are templates consisting of parameters, preconditions, and effects. Each action characterises a set of possible action instances by using the parameters to which values can be assigned to derive specific actions. *Preconditions* contain predicates that must hold before an action can be applied, and *effects* include predicates that simulate action occurrence. *Predicates* are statements that can be true or false, and consist of names and parameters.[2] A predicate's name describes relations between parameters. The combinations of parameters' values determine the truthfulness of predicates. Assuming an action's preconditions hold in some state, applying the action means using its effects to produce a new state.

A *plan* is a structure of actions. A plan is a solution to a given planning problem if the plan is applicable in the problem's initial state, and if after the execution of the plan, the goal is satisfied. A plan is applicable if the preconditions for the execution of the plan's first action hold in the initial state. The plan execution continues by applying other plan's actions in the order specified by the plan. If the goal is satisfied at the end of plan execution, the plan is a solution to the given problem.

The input to a planner is a planning problem specified in some syntax, commonly PDDL. A PDDL planning problem consists of two parts: a domain definition and a problem definition. Each definition is usually written in a separate file. The domain definition is used to define the actions and predicates, but it also allows one to specify other constructs, such as types of parameters, constants that have the same meaning for all planning problems in the given domain, functions to access and update numeric values, and axioms to assert relationships between predicates that are true in some state. On the other hand, the problem definition describes the initial state and goal. For a detailed introduction to PDDL, we refer to McDermott et al. [1998], while for examples of PDDL domain and problem definition, we refer to Appendix B. The output of the planner is a plan that guarantees the achievement of the goal. Planning is the process that connects the input and output. The planner defines a search space and goes through this potentially large space looking for a point that is defined as a solution. The search space can be of different forms and structures, where the difficulty of the search increases with the complexity of space.

The classical approach of solving planning problems relies on several simplifying assumptions: environments have a finite set of states, the initial state is complete and fully observable, actions are deterministic, environment states change only by executing actions, goals are either satisfied or not by plans, plans are ordered sequences of actions, and there is no explicit use of time [Ghallab et al. 2004]. Clearly, this approach is too restrictive for environments in which information completeness and consistency cannot be guaranteed, and plan execution cannot be guaranteed to proceed as expected. Therefore, much work has focused on developing planning approaches that relax one or more of these assumptions by allowing incomplete knowledge about the initial state, partially observable states, nondeterministic actions, actions with conditional effects, preferences and extended goals, partially ordered plans, durative actions, etc. We define and describe these concepts as needed throughout the article.

2.2. Methodology

We perform a comprehensive search for studies relevant to our treatment based on a systematic method [Kitchenham and Charters 2007]. We derive the framework of

[2]A PDDL predicate has the form `(NAME ?A1 ... ?An)`, where the arguments beginning with a question mark are parameters.

Table I. Primary Studies

ID	Study	ID	Study
S1	Qasem et al. [2004]	S28	Di Rocco et al. [2014]
S2	Ranganathan and Campbell [2004]	S29	Cirillo et al. [2012]
S3	Kotsovinos and Vukovic [2005]	S30	Madkour et al. [2013]
S4	Amigoni et al. [2005]	S31	Ortiz et al. [2013]
S5	Ding et al. [2006]	S32	Milani and Poggioni [2007]
S6	Vukovic et al. [2007]	S33	Georgievski et al. [2013]
S7	Carolis and Cozzolongo [2007]	S34	Garro et al. [2008]
S8	Courtemanche et al. [2008]	S35	Marquardt and Uhrmacher [2009a]
S9	Bajo et al. [2009]	S36	Jih et al. [2007b]
S10	Liang et al. [2010]	S37	Harrington and Cahill [2011]
S11	De Giacomo et al. [2012]	S38	Song and Kim [2011]
S12	Mastrogiovanni et al. [2010]	S39	Plociennik et al. [2009]
S13	Santofimia et al. [2010]	S40	Honold et al. [2014]
S14	Bidot et al. [2011]	S41	Caruso et al. [2013]
S15	Yordanova [2011]	S42	Corchado et al. [2009]
S16	Sando and Hishiyama [2011]	S43	Bacon et al. [2013]
S17	Hidalgo et al. [2011]	S44	Chen et al. [2014]
S18	Kaldeli et al. [2012]	S45	Köckemann et al. [2014]
S19	Song and Lee [2013]	S46	Yau and Buduru [2014]
S20	Sánchez-Garzón et al. [2012]	S47	Wang et al. [2015]
S21	Pajares Ferrando and Onaindia [2013]	S48	Eppe and Bhatt [2015]
S22	Fraile et al. [2013]	S49	Jean-Baptiste et al. [2015]
S23	Ha et al. [2005]	S50	Amato et al. [2015]
S24	Krüger et al. [2011]	S51	Sukkerd et al. [2015]
S25	Grześ et al. [2014]	S52	Vaquero et al. [2015]
S26	Marquardt et al. [2008]	S53	Chen et al. [2016]
S27	Heider [2003]		

dimensions for automated planning for ubiquitous computing from the relevant studies using qualitative analysis [Corbin and Strauss 2008]. We systematically arrange the relevant studies according to the framework's dimensions using a classification process [Smidts et al. 2014]. We then analyse the studies and aggregate meaningful information per dimension. Since studies belonging to the same dimension share properties, we can consolidate explanations effectively.

3. PRIMARY STUDIES AND FRAMEWORK

We select a total number of 53 relevant studies which we refer to as *primary studies*. Table I shows the studies, where their references are associated with unique identifiers used for convenience later for classification. Insights into the distribution of primary studies with respect to publication year, type, and venue are given in Appendix A.2.

Using the contents of primary studies, we develop our generic framework for automated planning for ubiquitous computing. Its basic structure consists of the following three main dimensions, each of which may in turn be split into subdimensions. We provide details on such structuring when appropriate in the remainder of the article.

—**Environments:** This dimension deals with questions related to ubiquitous computing environments. It determines what types of requests users can make, what types of sources that enable environment change can be supported, what aspects of environment physics are needed, and what sources of uncertainty can be identified.
—**Planning:** This dimension focuses on questions related to automated planning. It determines the types of purpose of using planning, types of planning techniques, definition, modelling and representation of planning problems, design and

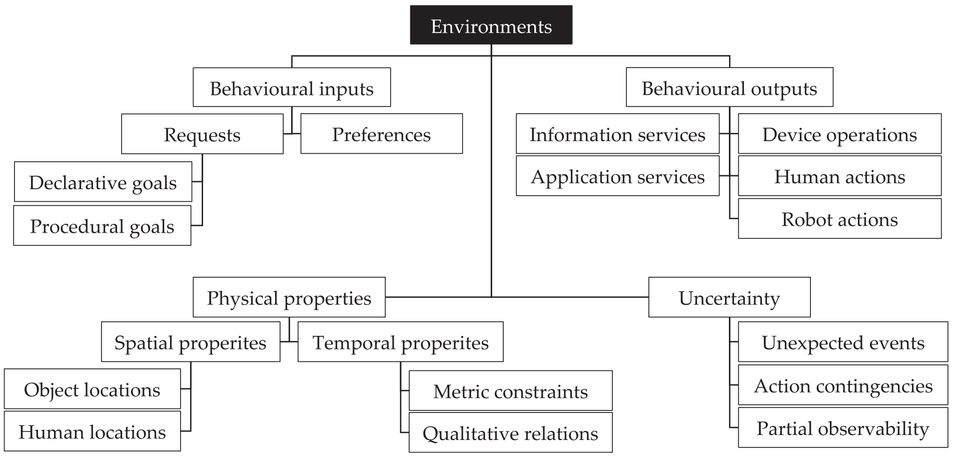

Fig. 1. Hierarchy of dimensions for environments.

implementation of planning systems, and integration and use of planners in ubiquitous computing.

—**Interpretation:** This dimension provides insights into the practical aspects of planning theories and solutions. The practical aspects refer to the approaches taken to understand better and demonstrate the complexity and applicability of theories. Practical aspects may also include technical and qualitative evaluations of solutions, and examinations of user satisfaction and acceptance of solutions.

4. ENVIRONMENTS

For the environments dimension, we catalogue four subdimensions, namely, behavioural inputs, behavioural outputs, physical properties, and uncertainty. Some of these may be further split into subdimensions. Figure 1 gives a quick overview of the hierarchy of the environments dimensions. In the following, we define each dimension and provide insights into the correspondence between these dimensions and planning.

4.1. Behavioural Inputs

We define *behavioural inputs* as the information representing one's desires according to which a ubiquitous computing environment should behave. We identify two subdimensions of behavioural inputs, depending on whether the satisfaction of one's desires is required or it is preferred but not required. *Requests* provide models of desired results issued for the purpose of achieving a mandatory behaviour, adaptation, or organisation of environments. One model of requests is *declarative goals*, which specify explicitly states of environments that need to be established. It relates to the question of *what* has to be achieved in some setting. For example, Theodore requires his bedroom to be clean. Another requests model is *procedural goals*, which specify a set of procedures to be performed so as to satisfy requests. These goals relate to the question of *how* to accomplish something in some setting. For example, Theodore wants Tars to first clean his bedroom and then the living room.

The *preferences* subdimension encompasses individual attitudes towards the environment behaviour (or organisation). Contrary to requests, preferences do no enforce mandatory satisfaction and are satisfied as much as possible. For example, Theodore may prefer a tomato sauce over ketchup for the chosen dish in the cooking scenario.

The basic connection between the behavioural inputs and planning can be established through requests, which can be directly mapped to goals in planning problems. As for preferences, these are regarded as *soft constraints* on plans whose quality increases when more constraints are satisfied. The specification of preferences in planning has been supported by the syntax of PDDL (version 3.0 [Gerevini and Long 2006]).

4.2. Behavioural Outputs

We define *behavioural outputs* as acts performed in ubiquitous computing environments whose outcomes modify the states of the environments. The acts may be performed by various physical objects, such as devices and robots, software components, and humans. We identify the following five subdimensions of behavioural outputs.

—*Device operations* are functionalities that specific devices can perform. Device operations are used to change the states of associated devices. By device we mean a piece of equipment embedded in an environment that has limited capabilities to interact autonomously with (other entities in) the environment. Examples of device operations include setting up a smoke alarm, pulling down window blinds, dimming lights, etc.
—*Human actions* represent behaviours to be performed by persons. Human actions are used to assist or guide people on the path to accomplish their goals or activities. For example, Theodore can be assisted with recipe steps to prepare his dish.
—*Robot actions* represent behaviours of robots performed in order to achieve some goals. Robots are autonomous and intelligent to a certain degree entities, hence their actions represent a separate dimension. Robot actions can be categorised in two groups: actions that transform the environment, such as Tars cleaning rooms, and those communicated to people, such as Tars wishing Theodore a nice evening.
—*Application services* define purposeful behaviours of applications installed on computers deployed in ubiquitous computing environments. Applications might be commercial, such as Adobe Acrobat and Apple Keynote, or developed for a specific purpose. An example of application service could be setting up the recipe steps for Theodore on slides using Microsoft PowerPoint.
—*Information services* represent knowledgeable behaviours built by collecting, managing, and reasoning over possibly distributed data. For example, Samantha uses an information service to find a restaurant for dinner for Theodore.

All types of behavioural outputs are in fact represented as actions in planning domains. The actions representing device operations, robot actions, and application services all have a similar purpose–to control some entity. Human actions differ in that their execution cannot be enforced as it depends directly on the will of involved humans, but the actions can be used by planners to create human-aware plans [Sisbot et al. 2007; Hoffman and Breazeal 2007; Talamadupula et al. 2010].

4.3. Physical Properties

Physical properties characterise situations of ubiquitous computing environments with respect to space and time. People and objects have a physical extension that relates them to one another and with space. *Spatial properties* qualify the relations among entities and their environment. Spatial representations define the types and qualities of the spatial properties and allow for proper reasoning over these [Aiello et al. 2007a]. There are two ways of spatial representations: one respecting the spatial characteristics of the underlying models, and the other treating space simply as a set of symbols without considering any geometrical or physical laws [Aiello et al. 2007b]. We call the former *purely spatial representation* and the latter *abstract spatial representation*.

Consider Theodore moving from his bedroom to the kitchen. In an abstract representation, one can represent the move as instantaneous. In a "pure" representation, this would not be possible and the fact that the locations have to be topologically connected and that the likelihood of instantaneous actions is little would have to be taken into account. There is thus an issue related to spatial arrangements, elsewhere known as spatial *realizability* [Lemon and Pratt 1997; Kontchakov et al. 2014], and at times also the need for the consideration of spatiotemporal properties [Andréka et al. 2007].

We identify two subdimensions of spatial properties, depending on whether the spatial relation is with objects or humans. These two subdimensions fall within the abstract way of representing spatial properties.

—*Object locations* comprise locations of environment objects. For example, Theodore's TV is in the living room. Such locations represent static information often predefined manually in the calibration phase of the environments.
—*Human locations* define the position of people within environments. As dynamic information, the location of humans is usually tracked to the level of some predefined areas, for example, rooms. More fine-grained information can include the posture and orientation of persons, which is crucial for many scenarios such as those concerning people's safety (e.g., fall detection and appropriate reaction). Such fine-grained spatial information may, however, become a subject to privacy concerns, which have to be considered when gathering and reasoning over it [Bettini and Riboni 2015].

The interpretation of human behaviour involves reasoning about time. For example, the activity of washing hands in care centres may be performed in a different time span by different people. The duration of the activity can be thus personalised with respect to people's abilities and health conditions. *Temporal properties* characterise the relationship of entities with time. In its generality, the time is defined using *temporal individuals* connected by temporal relations [Benthem 1983]. The most common temporal individuals are intervals and time points.

—*Intervals* are used to describe activities, where numerous relations between the intervals can be used to describe the interdependencies between activities, such as *X precedes Y*, *X meets Y*, *X same* as *Y*, etc. ([Benthem 1983; Allen 1983]). For example, consider the gas leakage scenario in Theodore's kitchen, where the activity of Theodore leaving the kitchen occurs before the activity of the kitchen door being closed. If *X* is the interval in which Theodore leaves the kitchen, and *Y* is the interval for closing the door, then the relation would be *X precedes Y*. We refer to temporal representations based on intervals as *qualitative relations*.
—*Time points* are used to describe the start and end times of activities, where three relations can be used to specify the constraints between the time points, namely, *P precedes Q*, *P same* as *Q*, and *P follows Q*, where *P* and *Q* are time points [Benthem 1983; Vilain and Kautz 1986]. For example, the time point when Theodore starts leaving the kitchen precedes the time point at which the door starts closing. We refer to temporal representations based on time points as *metric constraints*.

Thus, qualitative relations and metric constraints are two subdimensions of temporal properties. Though each dimension captures a different temporal representation, in general, it is possible to combine both types of temporal representations, which means that the two subdimensions are not mutually exclusive.

Spatial properties can be represented in planning problems using combinations of constants, objects, and predicates. As for the temporal properties, there are various approaches providing support for temporal planning. Many of these are based on durative actions in PDDL (version 2.1 [Fox and Long 2003]) where metric temporal annotations are incorporated into preconditions and effects of actions. The temporal annotation of preconditions indicates explicitly when the associated predicate must hold: at the start of the action, at the end of the action, or over the action's duration. The temporal annotation of effects signifies that the effect is immediate, when it happens at the start of the action, or delayed, when it happens at the end of the action. In addition to modelling temporal properties in domains, some planners may induce a temporal order of actions in plans without providing explicit temporal knowledge in the domains.

4.4. Uncertainty

Uncertainty refers to situations of ubiquitous computing environments in which the information describing the current state is ambiguous and unreliable due to the inherited dynamism of the environments. The dynamism can be characterised by diverse and continuous events, the nondeterminism concerning the behavioural outputs, and partial observability. Accordingly, we split uncertainty into three subdimensions.

—Events happening in exceptional and unpredicted situations are *unexpected events*. Recall the cooking scenario in Theodore's home. The missing ingredient might be classified as an unexpected event in the given situation.

—*Action contingencies* indicate states of actions (or operations or services) in which actions do not work correctly during execution. Action contingencies can be *failures* or *timeouts*. The former happen when action invocations return erroneous responses. These responses may be simple failures, when actions fail to respond within a limited number of invocation attempts; Byzantine behaviour, when actions complete successfully without providing the expected result; and permanent failures, when entities providing behavioural outputs break down [Kaldeli et al. 2016]. In the gas leakage scenario in Theodore's home, the window blinds get stuck, which might be classified as a simple failure. Timeouts occur when invocations provide no response after a certain amount of time.

—*Partial observability* refers to the imperfectness and incompleteness of information about environment states. State constituents, say, variables, may have different possible actual values and even unknown values, making the representatives of behavioural outputs no longer depend directly on such uncertain states. For example, Samantha might not know why the window blinds are stuck in the kitchen due to observing only partially the window segment.

Unexpected events and action contingencies interfere with the aspect of predictability in planning—planners do not know what may happen when plan actions are executed. Common ways to model unpredictability in planning are by incorporating nondeterministic outcomes and conditional effects into actions. Nondeterministic actions have alternative outcomes that are completely determined at the time of plan execution. Conditional effects enable actions to have effects conditioned on secondary preconditions. A plan would then contain conditional steps that are chosen depending on the environment state during execution.

Partial observability interferes with sensing, making environment states not necessarily known. Sensing is the process of observing and providing up-to-date views of environment states at planning (and/or execution) time. Generally, there are four sources of information about environment states: the initial states themselves, sensing actions, observations, and already executed plan actions. Sensing actions are planning actions executed explicitly to determine the truth value of some predicates. Observations sense state information automatically. The memory of previous actions is used to aid the inconsistency of states. Combinations of these and other concepts have been also used to improve the capability of planning to deal with uncertainty.

4.5. Review of Primary Studies

Table II shows the mapping of primary studies onto the subdimensions of the environments dimension. The "✘" indicates the support of a study for a respective dimension independently of how that support is provided.

Table II. The Dimensions of Primary Studies within the Environments Dimension

	Behavioural inputs								Physical properties						
	Requests			Behavioural outputs					Spatial		Temporal		Uncertainty		
Study	Decl. goals	Proc. goals	Preferences	Device op.	Human actions	Robot actions	App. services	Info. services	Object loc.	Human loc.	Qualitative rel.	Metric constr.	Unexp. changes	Action cont.	Partial observ.
S1		✘		✘											✘
S2	✘		✘	✘			✘	✘		✘				✘	
S3	✘		✘	✘	✘							✘	✘		
S4		✘		✘											
S5		✘	✘	✘					✘	✘					
S6	✘							✘		✘			✘	✘	
S7						✘									
S8		✘			✘							✘			
S9		✘	✘		✘					✘		✘	✘		
S10		✘	✘	✘											
S11	✘			✘	✘					✘				✘	
S12			✘		✘				✘		✘				
S13		✘		✘											
S14	✘	✘	✘	✘					✘	✘	✘		✘	✘	
S15					✘				✘						
S16					✘					✘			✘		
S17		✘			✘							✘			
S18	✘			✘					✘	✘	✘		✘	✘	✘
S19	✘	✘	✘	✘			✘	✘		✘					
S20		✘	✘		✘							✘	✘		
S21	✘			✘	✘			✘	✘	✘	✘				
S22	✘		✘	✘					✘	✘		✘	✘		
S23		✘		✘		✘				✘					
S24	✘			✘						✘					✘
S25	✘				✘		✘								✘
S26		✘		✘										✘	
S27	✘			✘							✘				
S28	✘			✘		✘				✘	✘	✘			✘
S29	✘				✘	✘			✘	✘		✘			✘
S30		✘						✘					✘	✘	
S31	✘				✘				✘	✘					
S32	✘			✘					✘	✘	✘				
S33		✘		✘					✘						
S34	✘			✘					✘				✘		
S35	✘			✘	✘		✘	✘							
S36		✘		✘					✘	✘					
S37				✘					✘						✘
S38								✘	✘	✘					
S39															
S40		✘	✘		✘						✘			✘	
S41	✘				✘				✘						
S42			✘		✘			✘	✘	✘			✘		
S43					✘								✘		
S44		✘	✘			✘			✘	✘					
S45	✘		✘			✘					✘				✘
S46			✘	✘					✘						
S47	✘					✘									
S48	✘			✘										✘	✘
S49	✘				✘										✘
S50	✘			✘		✘			✘	✘		✘	✘		
S51	✘		✘	✘	✘										
S52	✘		✘			✘			✘	✘		✘		✘	
S53								✘	✘						

Declarative Goals. The majority of classified studies have a traditional approach towards representing declarative goals, that is, descriptions of final states, and several studies incorporate extended forms of declarative goals. De Giacomo et al. [2012] combine maintenance and achievement goals using propositional formulae over domain propositions. In Kaldeli et al. [2012], extended goals are declarative expressions on numerical variables, temporal constructs, and maintainability properties. For an example of extended goal, we refer to Appendix B.4.

Declarative goals are interpreted as constraint networks that may include temporal, resource, symbolical, and information dependencies. In Di Rocco et al. [2014] and Amato et al. [2015], studies are based on the same planning approach. In Cirillo et al. [2012], the goal is a logical formula over the state and consists of subgoals each of which is associated with a value denoting its importance of achievement. For example, let `(dirt ?r ?dv ?i)` denote that the dirt in a room `?r` should be `?dv` with importance of achievement `?i`. Then, if Tars' goal is to clean the kitchen and bedroom, while more important is to clean the kitchen, it can be represented as the conjunction `(and (dirt kitchen 0 0.6) (dirt bedroom 0 0.4))`. The goals can be violated in some cases at the expense of a less efficient but valid solution. In addition to these reachability goals, maintenance goals are supported too.

Procedural Goals. The classified studies adopt an exceptionally conventional approach towards using procedural goals. A goal is either a single task or a list of tasks, where tasks are commonly interpreted analogously to the definition of tasks in Hierarchical Task Network (HTN) planning (see Section 5.2). Incidentally, Amigoni et al. [2005] enhance tasks with additional information on performance, cost, and probability of success for each method of a task. The performance parameter expresses the expected effectiveness of a method, the cost parameter indicates the amount of resource that would be consumed if the method is applied, and the probability of success gives the expectation that no failures will occur when the method is applied. While these parameters might appear useful to ubiquitous computing environments, their semantics are not defined, and their values should be provided manually by domain authors.

Preferences. A user may select and customise recipes [Kotsovinos and Vukovic 2005], indicate personalised choices on various devices [Ding et al. 2006; Fraile et al. 2013], and provide preferences on services [Liang et al. 2010; Song and Lee 2013; Chen et al. 2014], daily activities [Mastrogiovanni et al. 2010; Vaquero et al. 2015; Köckemann et al. 2014], domains [Bidot et al. 2011; Yau and Buduru 2014], treatments [Sánchez-Garzón et al. 2012], timetables [Bajo et al. 2009], companion systems [Honold et al. 2014], and products [Corchado et al. 2009]. In Ranganathan and Campbell [2004], user preferences are in the form of utility u for each predicate in different contexts, where $u \in [-10, 10]$. A predicate without a value has $u = 0$. So, each user has a utility for each environment state, whereas the utility of the goal state is a linear combination of the utilities of all entities relevant to the goal. The incorporation, maintenance, and handling of such preferences becomes cumbersome when predicates and users proliferate. Finally, willingness or user desires to perform tasks may be captured using variables that influence such desires [Sukkerd et al. 2015]. For example, the willingness of cooking task is influenced by whether Theodore is tired. Each value of such a variable is mapped to a willingness probability. That is, the willingness probabilities for when Theodore is tired or not are 0.3 and 0.9, respectively.

Device Operations. One group of studies represent device operations as planning actions directly [Ranganathan and Campbell 2004; Amigoni et al. 2005; Ding et al. 2006; Fraile et al. 2013; Krüger et al. 2011; Heider 2003; Di Rocco et al. 2014; Milani and Poggioni 2007; Georgievski et al. 2013; Garro et al. 2008; Yau and Buduru 2014; Eppe and Bhatt 2015; Amato et al. 2015; Sukkerd et al. 2015]. Preconditions typically represent the state in which a device must be to achieve the desired behaviour. Additionally, preconditions may encode other properties, such as the device's spatial attributes, which include the device location and environment region over which the action has effects, for example, Harrington and Cahill [2011]. For examples of device operations encoded in PDDL, we refer to Appendix B.2.

The rest of the classified studies conceptualise device operations using the notion of *service*, which is an abstraction of a device operation from its implementation details. Commonly, a single device may offer one or more services. Santofimia et al. [2010] propose a semantic model for the relationships between devices, operations, and services. From our point of interest, the model defines that devices provide services that execute operations over one or more entities found in environments.[3] Kaldeli et al. [2012] require each device to expose its functionalities as one or more Universal Plug and Play (UPnP)[4] services. UPnP services provide method calls that constitute UPnP actions with input and output parameters. UPnP services are translated into planning actions and augmented with additional semantics for the purpose of planning.

Human Actions. Human actions are primary entities in the domain of assisted living for implementing applications for care giving [Courtemanche et al. 2008; Bajo et al. 2009; Hidalgo et al. 2011; Sánchez-Garzón et al. 2012], cooking [Kotsovinos and Vukovic 2005; Sando and Hishiyama 2011; Ortiz et al. 2013], shopping [Corchado et al. 2009], training [Bacon et al. 2013], and system assembling [Honold et al. 2014]. It is practically difficult to extract a generalisation of human actions because of the diversity of adopted models and the lack of representation details (for examples of human actions encoded in PDDL, we refer to Appendix B.2). In addition to being represented as classical planning actions, human actions may consist of preconditions, a duration, and probability of state transition Cirillo et al. [2012]. These actions are successful under the assumption that their duration is fixed and they cannot be interrupted once started. Preconditions are not required, but if considered, they are verified only at fixed time points, meaning that the actions are always instantaneous. Sukkerd et al. [2015] use a performance function in the effect of a human action to quantify the belief of the planner about the human's performance of that action in different states.

One different representation of human actions is by using affordances and capabilities, which are regions in a proper space characterised by a set of attributes [Mastrogiovanni et al. 2010]. For example, an object affording a capability "to take" is a region in the respective affordance space characterised by the weight and grasp size attributes. People have initial capabilities and can acquire new ones by using object affordances. Say that the object is a vacuum cleaner and Theodore has the initial capability to take it. By taking the vacuum cleaner, he acquires the capability "to clean."

Robot Actions. A robot may move to some location, sleep for a certain period, stay in some position, clean a room, pick up some item, remind a person to take medication, etc. Part of the classified studies represent robot actions as classical planning operators (for examples, see Appendix B.2). Carolis and Cozzolongo [2007] describe robot actions by sets of preconditions and possibly nondeterministic effects. Both sets are associated with probability values, that is, the probability that preconditions will hold in the current state, and the probability that the action will have the effect in the state. In Di Rocco et al. [2014] and Amato et al. [2015], the same planning approach is adopted [Di Rocco et al. 2013]. Namely, robot actions are represented using *activities*, which are extended predicates that besides variables include a temporal interval, resources,

[3]An extended and formalised version of the model can be found in Santofimia et al. [2011].
[4]www.upnp.org.

and required and provided information, and temporal constraints describing interactions and dependencies between those activities. Activities can be used to represent the functionality, preconditions, and effects of actions. The instantiation of activities either produces information or modifies the state. Köckemann et al. [2014] describe the conditions and effects of robot actions by attaching state variables to temporal intervals to model environment information, and by using temporal and logical constraints to determine possible assignments of the variables involved. Ha et al. [2005] use the concept of service to represent robot actions, namely, as atomic processes in the Semantic Markup for Web Services (OWL-S) terminology [Martin et al. 2007].

Application Services and Information Services. While several studies aim at incorporating application services, only Ranganathan and Campbell [2004] describe their representation at a planning level, namely, as PDDL actions (for an example, see Appendix B.2). On the other hand, information services are commonly used within the scope of mobile applications and are implemented as Web services. Examples of information services found in the classified primary studies include currency converter, weather information provider, restaurant finder (which provides a list of available restaurants), provider of information about the schedule of a person from Google Calendar, translator (which translates some content from one language to another), speech synthesizer (which converts text to speech), scheduler synchroniser, shop finder, get a location of facility for baby changing, etc. In most cases, it is unclear how information services are represented at the planning level.

Spatial Properties. Typically, some form of structure is given to the spatial properties, such as a hierarchy of locations that represents the being-part-of relation (mereology [Whitehead 2010]) and at times also includes connectedness information (mereotopology [Casati and Varzi 1999]). Though these models are weak from the realizability point of view, they can offer sufficient knowledge for proper planning.

A general form of object locations is a predicate whose name denotes a spatial relation and its arguments represent objects and locations. Among the spatial relations considered in the primary studies are `at`, `pos`, `in`, `distance`, and `near-by`, all with the intuitive meaning (for examples, see Appendix B.1). In many cases, it is not the only requirement that an object (e.g., lamp) is at a specific location, but also the knowledge about how it affects other objects at or nearby its location (e.g., Milani and Poggioni [2007] and Harrington and Cahill [2011]). Mastrogiovanni et al. [2010] use the concept of capabilities and affordances to represent object locations, where in a representation scale, level one corresponds to furniture and containers inside rooms and level two represents different rooms. Harrington and Cahill [2011] use a geometric model to represent device locations.

Human locations are also commonly represented using a predicate abstracted as `(in ?human ?location)`. Scarcely any study uses more fine-grained information about human locations (e.g., coordinates are used in Corchado et al. [2009]). Human locations are extracted from wearable devices, such as radio frequency identification tags [Bajo et al. 2009; Fraile et al. 2013; Ha et al. 2005; Ortiz et al. 2013; Corchado et al. 2009; Amato et al. 2015] and mobile phones [Song and Lee 2013], and global positioning systems [Sando and Hishiyama 2011; Fraile et al. 2013] or other location tracking systems [Krüger et al. 2011; Kaldeli et al. 2012; Di Rocco et al. 2014; Jih et al. 2007b].

Temporal Properties. Support for qualitative relations using temporal relations between intervals is provided by only a handful of studies. The approaches presented in Di Rocco et al. [2014] and Amato et al. [2015] use robot actions in which temporal relations restrict the bounds of intervals of activities (recall that these are extended predicates over variables). Suppose that the moving action for Tars has the following two activities: a_1, representing the functionality of Tars to move from the bedroom to the kitchen, and a_2, being a precondition and representing that Tars is in the bedroom. A temporal relation is used to describe that the moving of Tars is constrained by Tars being in the bedroom, that is, a_1 *is met by* a_2. Temporal relations between intervals attached to preconditions and effects of actions are defined in a similar manner in Köckemann et al. [2014]. The rest of the studies classified in Qualitative relations use planning techniques that can produce partially ordered plans.

Explicit temporal annotations of preconditions and effects in actions are presented in the usual way: at the start of an interval and at the end of an interval (e.g., Bajo et al. [2009], Sánchez-Garzón et al. [2012], Fraile et al. [2013], Di Rocco et al. [2014], Amato et al. [2015], Köckemann et al. [2014], and Vaquero et al. [2015]).

Unexpected Events. The primary studies consider two types of unexpected events: dynamic goals and context changes. Dynamic goals represent updates of current planning goals for which there is already a computed plan under execution [Bidot et al. 2011; Madkour et al. 2013; Corchado et al. 2009; Bacon et al. 2013]. Dynamic goals may thus interrupt the plan execution. Other classified studies focus on unexpected context changes. These may satisfy effects of already planned actions, or invalidate preconditions that were true during planning or action instantiation. These unexpected events are usually handled by plan repair or replanning (see Sections 5.7 and 5.10).

Action Contingencies. The majority of primary studies provide support for simple failures, and only few additionally for permanent failures and/or timeouts [Vukovic et al. 2007; Bidot et al. 2011; Kaldeli et al. 2012; Madkour et al. 2013]. The provided support is based on domain representation, such as nondeterministic effects [De Giacomo et al. 2012; Vaquero et al. 2015], conditional effects [De Giacomo et al. 2012; Marquardt et al. 2008; Eppe and Bhatt 2015], and other domain-specific knowledge [Ranganathan and Campbell 2004]; and conditions incorporated during plan execution and recovery [Vukovic et al. 2007; Bidot et al. 2011; Kaldeli et al. 2012; Madkour et al. 2013; Honold et al. 2014]. For example, Ranganathan and Campbell [2004] encode additional domain knowledge in specific actions to inform the planner that action execution can be retried, if failed. Conditions for timeouts may also depend on the type of actions. For example, an action invoked to close a door may take a shorter time to execute than an action to pull down window blinds. A timeout can be set to as much as the time an average fast action is expected to respond [Kaldeli et al. 2012].

Partial Observability. Constraint networks can cope with information incompleteness by performing "online sensing," meaning context changes are dynamically incorporated into the networks [Kaldeli et al. 2012; Di Rocco et al. 2014; Köckemann et al. 2014]. Qasem et al. [2004] deal with information imperfectness by sensing based on local completeness of information and relevance of information sources. When there is insufficient information to validate conditions, information is sensed from relevant sources. For unknown values, Eppe and Bhatt [2015] create multiple possible states and pick the one that contains the missing information provided by sensing actions. The rest of the classified studies deal with partial observability using observations based on probabilistic models, such as dynamic Bayesian networks [Krüger et al. 2011], Markov decision processes [Grześ et al. 2014; Jean-Baptiste et al. 2015], and other probability distributions [Cirillo et al. 2012; Harrington and Cahill 2011].

5. PLANNING

Figure 2 gives a glimpse of the hierarchy of the planning dimension for which we catalogue nine subdimensions.

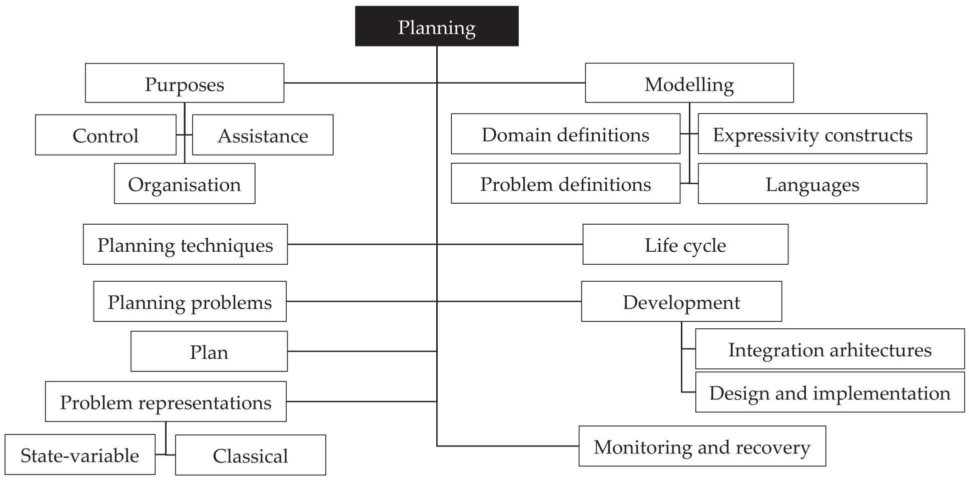

Fig. 2. Hierarchy of dimensions for planning.

5.1. Purposes

Purposes define what planning is used in ubiquitous computing. We identify the following three types of purpose.

—*Control* specifies that planning is used to create courses of actions related to the operation of environments. The execution of such actions does not involve human intervention. For example, planning used to control the devices in Theodore's home.
—*Assistance* involves planning for courses of action that either are aware of people's activities, or provide help and guidance to people (e.g., elderly people in care centres).
—*Organisation* indicates that planning outputs courses of tasks used to manage or arrange some environment entities. For example, organise schedules of patients and staff in hospitals.

5.2. Planning Techniques

Planning techniques encompass the techniques used to realise planning. Next, we introduce the basic ideas behind techniques attracting attention in ubiquitous computing.

—HTN planning requires an initial state, an initial task network (a list of tasks) to be accomplished, and a domain model consisting of a set of actions and a set of tasks each of which has one or more methods [Erol et al. 1994]. The methods associated with a task encode different ways of accomplishing that task. Planning begins with the initial task network and uses methods to decompose a network's tasks until actions are reached that constitute the plan [Georgievski and Aiello 2015].
—Probabilistic planning uses probabilities associated with nondeterministic actions to search for plans. Probabilities are used to choose only one outcome for each action. Probabilistic planning is naturally modelled using Markov Decision Processes (MDPs) [Boutilier et al. 1999]. MDPs are fully observable models that use state-transition probabilities and action costs. A next state and an expected cost depend only on a previous state and action applied, and not on additional previous states. Partially observable MDP (POMDP) planning makes use of sensing actions and memory of previous actions to aid unobservable states.
—Heuristic-based planning uses heuristic information to guide the search for plans. The heuristic information or heuristic functions tell planners about regions of the

search space in which plans may be found. Heuristic functions can be provided by users or extracted automatically from domain models [Bonet and Geffner 2001].
—Temporal planning deals with durative actions and actions that may overlap in time. States have to include information about the time, for example, timed literals in PDDL; when actions have started; and the actions that have not yet finished.
—Graph-based planning uses planning graphs to reduce the search for plans [Blum and Furst 1997]. A planning graph consists of nodes, represented by actions and predicates, and edges, which are links either from a predicate to actions or an action to predicates. Planning graphs enable planners to maintain information about incompatible predicates and incompatible actions, and use it to prune the search space.
—CSP-based planning assumes planning problems encoded as Constraint Satisfaction Problems (CSPs) whose inconsistencies are to be solved by constraint solvers. A CSP consists of variables, domains of variables, and constraints over variables. The constraints represent restrictions over the values that can be assigned to variables. A solution to a CSP is an assignment for each variable with a value from its domain such that all constraints are satisfied [Kaldeli et al. 2016].
—Partial-order planning makes choices relevant only to solving the current part of a planning problem. The choices may involve decisions about ordering among plan actions, which are left unordered until necessary to be sequential; and decisions about variable assignments, which are left unassigned until needed for the satisfaction of some conditions. The search is performed in the space of partial plans, which are structures containing actions, ordering among action, causal connections between actions, and constraints on variable assignments.
—Case-based planning uses plans or portions of plans from previous cases stored in memory to solve new planning problems [Hammond 1989]. A case is a past experience consisting of an initial problem, a plan that solves the problem, and a final state reached after the solution is applied. Case-based planners create and modify plans by relying on their memory rather than rules. Examples of memories include tasks, resources, and time. The memories of past successes are used when creating new plans, the memories of past failures are used to identify potential problems, and the memories of past repairs are used to instruct planners how to handle plan repairs.

5.3. Planning Problems

The dimension of *planning problems* examines whether and to what degree planning problems intended to be solved in ubiquitous computing are defined. Defining problems is an important step for the analysis of the complexity and validation of domains in ubiquitous computing. A set of well-defined planning problems will provide the means to also measure the progress in the field of planning for ubiquitous computing.

5.4. Plans

Plans represent the solutions to planning problems. The dimension of *plans* investigates how such plans are usually defined and what types of plans can be found. A plan definition is important for two main reasons. On the one hand, a plan definition clearly shows that a plan is indeed a sound solution towards achieving a desired behaviour of ubiquitous computing environments. On the other hand, a plan definition that includes the structure and conditions under which a plan is executable may be crucial to further execute plan actions. Considering plan types, plan can include instantiated actions or uninstantiated actions in which case plans are called *abstract*. The actions in abstract plans are usually bound to concrete instances later in a system's operation lifecycle. Furthermore, plans can be *totally ordered*, in which case actions are ordered in a sequence, or *partially ordered*, where actions are unordered with respect to each other.

5.5. Problem Representations

Problem representations are concerned with the models used to represent planning problems. We identify two general models that are equivalent in expressive power, meaning problems represented in one representation can also be encoded using the other one [Ghallab et al. 2004]. *Classical representations* assume an environment state as a set of ground predicates. The actions are expressions that specify which predicates should be in the state so that the action is applicable, and which predicates should change their values after the action application. *State-variable representations* assume a state as a set of values of a finite set of state variables. Actions represent functions that change the values of those variables.

5.6. Modelling

Planning needs domain knowledge that contains various pieces of information about ubiquitous computing environments. *Modelling* domain knowledge is a difficult task that often includes studying the requirements of these environments, specifying domain and problem definitions, and testing those definitions using planners. We focus here on the approaches used to create domain and problem definitions, specific constructs used to express planning problems, and languages used to specify problems.

—*Domain definitions* focus on the ways of defining domain knowledge. One way is to define domain knowledge manually. Given only a theory about ubiquitous computing environments, it is tedious, error-prone, and often impractical for designers to manually model domain knowledge [McCluskey 2002]. Even if designers are experts in ubiquitous computing, typing the domain specification takes a great deal of time and usually causes unintentional errors. A more practical way would be to use tools that support engineering of the domain knowledge. The main benefits of automated acquisition of knowledge are making planners to be even more autonomous by learning new knowledge and correcting existing domain models of ubiquitous computing environments. We are mainly interested in the second way of domain definitions.

—*Problem definitions* focus on the process of generating and composing planning problems, that is, domain specifications and problem specifications. The creation of the former is discussed in the domain definitions dimension. Independently whether the domain specification is encoded manually or acquired automatically, it is a common practice to automatically translate it into a planning-level representation once retrieved from some storage point. The problem specification is generated automatically from the information describing the current state of environments and the request.

—*Expressivity constructs* encompass required or preferred expressive power of languages adopted to define ubiquitous computing environments. While the vision would be to have planners supporting expressivity constructs that cover a wide spectrum of properties of the environments, finding the right balance between the expressiveness and computational complexity of planners is crucial [Heider 2003].

—*Languages* refer to the syntax used to specify domain definitions and problem definitions of ubiquitous computing environments.

5.7. Monitoring and Recovery

When executing plans, ubiquitous computing environments may evolve differently than what was expected during planning. The discrepancy between actual and anticipated outcomes calls for planning systems to decide how to proceed. *Monitoring and recovery* cover the monitoring of environments and cases of what to do when plans no longer will achieve the goal. We abstract away the following two general processes.

—*Monitoring process* observes changes in environment states and during plan execution. Two main tasks that this process consists of are sensing and execution monitoring. *Sensing* observes and provides an up-to-date view of the current environment state at planning/execution time. *Execution monitoring* executes plan actions, monitors, and verifies that they are executed as expected during planning/recovery.
—*Recovery process* handles unexpected events and action contingencies occurring during plan execution. In general, various tasks may be used in the recovery process, such as regression (e.g., Fritz and McIlraith [2007]), precondition delay, action retrying (e.g., Ranganathan and Campbell [2004]), action omitting (e.g., Muise et al. [2013]), action replacement, replanning, and so forth.

5.8. Lifecycle

Lifecycle defines the phase of operation cycle of ubiquitous computing systems in which planners are invoked. The choice of the phase generally depends on the type of decisions needed to be made. A strategic decision would answer the question of what basic actions are to be executed and in which order, while an operational decision would refer to what entities should execute the actions [Bidot et al. 2011]. Planning can provide only a strategic decision at design time or compile time, while both types of decisions can be supported if planning is invoked at a system's runtime.

5.9. Development

Development refers to the framework that brings together various aspects of developing planners for ubiquitous computing. The development dimension in fact focuses on the aspects ranging from the design of planners through their manifestation as software to the integration of the planners into larger ubiquitous computing systems. *Design and implementation* focuses on a common component design upon which planners are realised. It also includes the level of software development (or maturity) of the planners realised for ubiquitous computing. On the other hand, *Integration architecture* is the paradigm upon which ubiquitous computing systems integrating planners are designed and implemented. We identify the following three paradigms.

—*Multiagent systems* consist of collections of agents. Each agent is a computer system capable of exhibiting to some extent an autonomous behaviour—to decide what to do so as to satisfy some objectives, and to interact with other agents—to exchange messages through a network [Wooldridge 2009]. Successful interactions depend on the abilities of agents to cooperate, coordinate, and negotiate with each other.
—*Modular architectures* are a design model in which systems consist of distinct modules interconnected together. Modules represent a separation of functionality of systems into independent and logically bound concerns.
—*Service-Oriented Architectures* (SOAs) are an architectural model that enhances efficiency, evolution, and productivity by considering services as a primary way through which logic is represented [Erl 2007]. Services are independent software programs with distinct characteristics remotely accessible through standardised interfaces.

5.10. Review of Primary Studies

Table III shows the arrangement of primary studies into the subdimensions of the planning dimension.

Purposes. The primary studies use planning almost equally to control devices and services as to assist people. Several studies use planning for both types. Most of the

Table III. The Dimensions of Primary Studies within the Planning Dimension

	Purposes						Prob. rep.		Modelling						Development		
																Integ. arch.		
Study	Control	Assistance	Organisation	Planning tech.	Planning prob.	Plan	Classical	State-variable	Domain def.	Problem def.	Express. constr.	Languages	Monit. and recov.	Lifecycle	Design and impl.	Multiagent	Modular	Service-orient.
S1	✘			✘	✘		✘						✘		✘			✘
S2	✘	✘		✘		✘	✘			✘		✘	✘		✘		✘	
S3	✘	✘		✘	✘	✘	✘			✘	✘	✘	✘	✘	✘			✘
S4	✘			✘	✘	✘				✘					✘	✘		
S5	✘			✘	✘	✘				✘		✘	✘					
S6		✘		✘		✘	✘			✘	✘	✘	✘	✘	✘			
S7	✘			✘						✘			✘					
S8		✘		✘	✘	✘											✘	
S9			✘	✘	✘	✘							✘	✘	✘	✘		
S10				✘											✘			✘
S11		✘		✘	✘	✘	✘	✘										✘
S12		✘		✘	✘	✘									✘			
S13				✘	✘							✘			✘	✘		
S14	✘			✘	✘	✘	✘			✘		✘	✘	✘	✘			✘
S15		✘		✘		✘	✘				✘	✘						
S16						✘							✘					
S17		✘	✘	✘	✘				✘	✘		✘			✘			
S18	✘			✘	✘	✘		✘		✘	✘	✘	✘	✘	✘			✘
S19		✘		✘	✘	✘	✘					✘			✘			
S20				✘		✘						✘	✘		✘		✘	
S21	✘	✘		✘	✘	✘	✘	✘				✘				✘		
S22	✘	✘		✘		✘				✘			✘	✘	✘	✘		
S23	✘			✘		✘	✘			✘		✘			✘			✘
S24	✘	✘		✘		✘	✘					✘		✘				
S25		✘			✘			✘	✘	✘					✘			
S26	✘	✘		✘		✘								✘				✘
S27	✘			✘	✘	✘	✘				✘	✘			✘			
S28	✘			✘		✘		✘		✘			✘		✘			
S29	✘	✘		✘	✘	✘		✘					✘		✘			
S30		✘			✘	✘				✘			✘	✘	✘			
S31		✘			✘	✘	✘		✘	✘		✘						
S32	✘	✘		✘	✘	✘	✘	✘				✘			✘			
S33	✘			✘	✘	✘	✘		✘	✘		✘		✘	✘			✘
S34	✘					✘							✘				✘	
S35	✘			✘	✘	✘	✘				✘	✘		✘	✘			✘
S36	✘			✘		✘				✘		✘			✘	✘		✘
S37				✘	✘			✘		✘		✘						✘
S38		✘	✘			✘					✘	✘						
S39	✘			✘	✘	✘	✘					✘						
S40		✘		✘		✘				✘			✘		✘		✘	
S41	✘	✘		✘		✘	✘					✘						
S42		✘		✘			✘			✘			✘	✘	✘	✘		
S43		✘					✘						✘		✘			
S44	✘			✘	✘		✘								✘			✘
S45	✘			✘	✘	✘		✘					✘		✘			
S46	✘			✘	✘	✘												
S47	✘			✘	✘		✘					✘			✘			✘
S48	✘			✘	✘	✘	✘						✘		✘			
S49		✘		✘	✘										✘			
S50	✘			✘										✘	✘			✘
S51	✘	✘		✘	✘		✘					✘			✘			
S52		✘		✘	✘	✘	✘		✘			✘	✘	✘	✘		✘	
S53		✘		✘											✘			✘

Table IV. Planning Techniques Employed by the Primary Studies

Study	Technique	Study	Technique	Study	Technique	Study	Technique
S1	HTN	S15	HTN+POP,H	S29	P	S43	
S2	G+SAT	S16		S30		S44	HTN
S3	T	S17	HTN	S31		S45	CSP+H
S4	HTN	S18	CSP	S32	G+SAT+MIP	S46	MDP
S5	HTN	S19	HTN	S33	HTN	S47	H
S6	T	S20	HTN	S34		S48	Approximation
S7	P	S21	POP	S35	G	S49	POMDP
S8	MDP	S22	CBP	S36	G	S50	CSP
S9	CBP	S23	HTN	S37	P	S51	MC
S10	HTN	S24		S38		S52	T
S11	H	S25	POMDP	S39	T	S53	H
S12	Affordance	S26	HTN	S40	HTN+POP		
S13	HTN	S27	H,POP,MC	S41	STRIPS		
S14	HTN+POP	S28	CSP	S42	CBP		

Legend: Graph (G); Heuristic (H); Temporal (T); Probabilistic (P); Hierarchical Task Network (HTN); Markov Decision Process (MDP); Partial-Order Planning (POP); Constraint Satisfaction Problem (CSP); Case-Based Planning (CBP); Partially Observable MDP (POMDP); Mixed Integer Programming (MIP); Satisfiability problem (SAT); Model-checking (MC).

studies producing control sequences include device actions only, followed by studies involving robots only, and a few studies combining both robot and device actions. Cirillo et al. [2012] create plans that include only robot actions; however, the plans are human-aware, that is, they are generated based on forecasts of human actions and constraints on human interaction. Other studies that use planning for assistance produce plans involving human actions commonly represented by people's daily activities. These plans are used to either alert or guide people. For example, plan steps that should be followed by elderly people, care plans that should be performed by patients and/or caregivers, route plans that should guide people through a shopping mall, etc.

A handful of studies use planning for organisational purposes. Bajo et al. [2009] improve the management in hospitals by creating plans that organise dynamically nurses' working time, manage standard working reports about nurses' activities, and guarantee that patients assigned to particular nurses receive proper care. Similarly, Yordanova [2011] uses organisational plans to arrange nurse activities related to patient care. Besides plans with activities for patients, Hidalgo et al. [2011] provide organisational plans for care centres based on their current context, available resources (e.g., rooms) and staff, and their organisation rules.

Planning Techniques. Table IV shows the planning techniques adopted by the primary studies. HTN planning is the most frequent one with the following reasons for its suitability to ubiquitous computing.

—*Causality*. Yordanova [2011] argues that causal reasoning can be lost when modelling (human) actions in some domains. HTNs help in addressing this issue by encoding a small set of simple actions at the lowest level of hierarchies. These actions would then constitute a next, more coarse-grained level represented by compound tasks.
—*Flexibility*. Compound tasks allow formulation of multiple strategies in their methods before reasoning on low-level actions [Qasem et al. 2004]. The flexibility is reflected in the minor effort needed to add or remove a strategy from some task.
—*Effectiveness*. HTN planners are a "good compromise" between wide reusability and effectiveness in comparison to domain-independent planners [Marquardt et al. 2008]. The latter planners do not require domain-specific knowledge, which is included in

the compound tasks of HTN planning, thus they are widely applicable, however characterised by weak efficiency.

Probabilistic planning is another common approach for which the main reason of using lies in the ability to capture uncertainty and quantify the cost or success of computed plans. Carolis and Cozzolongo [2007] adopt a model based on Bayesian Networks (BNs) in which goals are associated with probabilities, and plans maximise the expected utility given the probabilities of goal BNs. MDPs are used to produce plans that maximise an expected reward [Courtemanche et al. 2008] and maximise user requirements [Yau and Buduru 2014]. POMDPs are employed to model uncertainty coming from the unpredictability of human behaviour and inconsistency of data [Grześ et al. 2014], and to compute plans that provide optimal alerts to people [Jean-Baptiste et al. 2015]. Cirillo et al. [2012] adopt a partially observable planning approach to reason over belief situations and reach situations in which plans with satisfactory values are found. Finally, Harrington and Cahill [2011] envision a probabilistic approach in which objects modelled in actions are associated with probability values that indicate the confidence of successful state transitions of actions.

A few studies make use of heuristic-based planning with the purpose of improving the time spent searching for plans. De Giacomo et al. [2012] identify that with the growth of planning problems, which affects the branching factor and depth of the search space, their approach becomes intractable. The time spent to solve a relaxed planning problem is reduced to polynomial by adopting the delete-relaxation heuristic function, which requires removal of negative preconditions and negative effects from the original planning problems [Hoffmann and Nebel 2001]. Chen et al. [2016] use a heuristic function to estimate the reliability of an execution path from the current service to the last service.

Di Rocco et al. [2014] adopt a meta-CSP approach: the problem of refining the goal, which is a constraint network, is cast as a high-level CSP that builds on lower-level CSPs, each for a specific type of constraint inconsistencies. The main reasons for the suitability of CSP-based planning to ubiquitous computing are as follows.

—*Numeric variables*. Variables whose values range over large domains are common in ubiquitous computing. For instance, variables representing temperature measurements, locations, etc. CSP-based planners handle numeric variables efficiently [Kaldeli et al. 2012].
—*Rich interrelationships*. Entities within ubiquitous computing environments often have causal, temporal, or other type of interdependencies. For instance, actions may be subject to deadlines, or they may include spatial information. CSP-based planning supports modelling of causal, temporal, resource, and information dependencies [Di Rocco et al. 2014; Amato et al. 2015].
—*Online sensing*. Recall that (online) sensing is the process incorporating context changes into planning models dynamically and efficiently. CSP-based planning supports online sensing by having context changes, expressed as constraints, added to or removed from constraint networks on the fly [Kaldeli et al. 2012; Di Rocco et al. 2014; Amato et al. 2015].
—*Continual planning*. Continual planning integrates the processes of planning, execution, and monitoring. It mainly focuses on questions of when and how to postpone parts of planning to phases of execution and monitoring, and when to refine or revise partially executed plans [Brenner and Nebel 2009]. Constraint networks seem suitable for continual planning due to the support for dynamic reconfigurations [Kaldeli et al. 2012; Di Rocco et al. 2014].

Memories within case-based planning are used by several studies. For instance, Bajo et al. [2009] use memories of tasks, resources, and time to create working schedules in hospitals. Case-based planning appears suitable for ubiquitous computing due to the following reasons [Bajo et al. 2009; Fraile et al. 2013; Corchado et al. 2009].

—*Learning ability*. Case-based planning can learn from initial knowledge and past experiences, which is needed for environments with incomplete and inconsistent information.
—*Adaptive capacity*. In ubiquitous computing, the needs of people or environment objectives may change over time. The learning ability together with the capability of case-based planners to interact autonomously with the environments provides the planners with a large capacity to adapt to people's needs and environment objectives.
—*Improve planning*. The learning ability and adaptive capacity enable case-based planners to increase their ability to solve planning problems over time.

There are two main reasons for the suitability of partial-order planning to ubiquitous computing: the predisposition to deal with durative actions, temporal and resource constraints [Pajares Ferrando and Onaindia 2013; Smith et al. 2000]; and the inherent support for partially ordered plans, thus a high degree of execution flexibility.

Sukkerd et al. [2015] describe cooperation between ubiquitous computing systems and humans as a planning problem, which is then modelled as a stochastic multiplayer game. Model checking is used to reason over the game to find a strategy that guarantees that a coalition of players achieves the given goal. The approach appears suitable for ubiquitous computing because it allows for explicit modelling of humans and their appropriate involvement in the operation of ubiquitous computing systems.

Some primary studies adopt a hybrid planning approach, which is a combination of planning techniques or planning with other techniques. There is a hybrid between HTN planning and partial-order planning, which does not require additional (control) knowledge (in comparison to pure HTN planning) [Bidot et al. 2011; Yordanova 2011; Honold et al. 2014]. This hybrid allows one to encode and deal with procedural knowledge supported by HTN planning, and to reason about causal dependencies provided by partial-order planning. Milani and Poggioni [2007] use a hybrid between graph-based planning, satisfiability, and mixed integer programming. Planning problems are first encoded as planning graphs, then the planning-graph relationships are translated into logical and numerical formulae, and finally, the formulae are converted to mixed integer linear programming constraints. Partial-order planning in combination with argumentation is proposed In Pajares Ferrando and Onaindia [2013]. This hybrid uses a reasoning mechanism that allows the agents in a multiagent system (see Section 5.9) to defeasibly support their decisions, interact among each other, and jointly create plans. A hybrid between CSP-based planning and heuristic-based planning is employed in Köckemann et al. [2014]. Incorporating heuristic functions in CSP planning provides performance gain in plan generation.

Two primary studies develop particular planning techniques. In Mastrogiovanni et al. [2010], objects, locations, and actions of ubiquitous computing environments are represented functionally based on the concept of affordances and capabilities, which are regions in a proper space with some attributes. The functional representation is motivated by the need to support human cognition that is otherwise not directly related to problem solving. The planning process first reduces the search space by identifying the key objects and possible actions along with an incomplete list of their causal relationships, and then searches for plans using the reduced space. Eppe and Bhatt [2015] propose to use an approximation of the knowledge about the environment state and a reasoning mechanism that takes into account the causal relationships between temporally ordered states to explain state properties that are not directly perceivable.

The planning problem is translated into an answer set programming problem. This approach is suitable mainly because it is able to detect and deal with uncertainty.

The rest of the studies use already existing planners and provide no information about adopted planning techniques.

Planning Problems. Only a limited number of primary studies provide clear and articulated definitions of the planning problems they are trying to solve. In a few other studies, one might extrapolate the planning problems by inspecting the input specified to the algorithms implementing the adopted planning techniques. Many primary studies offer only informative descriptions of what planning problems consist of, often with scarce details about which ubiquitous computing problem is actually targeted. Several studies unfortunately only make a superficial reference to planning.

Plans. The primary studies commonly describe plans intuitively as sequences of actions, while several provide more precise definitions. A *partially ordered plan* is a tuple $\langle A, \prec, L, V \rangle$, where A is a set of actions, $\prec$ is a set of ordering constraints between the actions in A, L is a set of casual links between the actions in A, and V is a set of constraints on variable assignments [Bajo et al. 2009; Bidot et al. 2011; Pajares Ferrando and Onaindia 2013]. A *causal link* is a relation of the form $\langle a, \phi, a' \rangle$ indicating that a logical formula ϕ, which is an effect of action a, is a part of the precondition of action a'. Di Rocco et al. [2014] define a plan as a constraint network in which intervals have fixed bounds allocated such that all temporal constraints are satisfied; resources consumed by extended predicates (activities) are not depleted over time; each variable has a single value at a time; and for each required information input there is an extended predicate that produces a consistent information output. Cirillo et al. [2012] define the solution to their planning problems as a graph whose nodes represent actions and edges represent observation sequences. The mapping from observation sequences to actions is generally known as a *policy*. The conditions under which a graph-based policy is considered as a solution to a planning problem are known and well defined. Some studies also define conditions under which plans are executable in a given state [Bajo et al. 2009; De Giacomo et al. 2012; Cirillo et al. 2012; Milani and Poggioni 2007]. The conditions involve the applicability of the first plan action into the initial state, which gives a new, successor state, and the sequence of pairs of actions and states that lead to the final state in which the given goal is satisfied. Furthermore, three studies make use of abstract plans, which are sequences of high-level descriptions of service operations that cannot be directly invoked [Vukovic et al. 2007; Bidot et al. 2011; Madkour et al. 2013]. Abstract plans are then made executable by binding abstract services to currently available service instances at runtime. Finally, considering the ordering among plan actions, totally ordered plans are more common than partially ordered plans.

Problem Representation. Most primary studies represent planning problems classically using predicates and actions with preconditions and effects. A combination of predicates with state variables is adopted in Milani and Poggioni [2007] and Pajares Ferrando and Onaindia [2013], where state variables are represented as functions in the manner of PDDL (version 2.1 [Fox and Long 2003]). State-variable representation comes naturally for approaches based on CSP-based planning. Kaldeli et al. [2012] model domains based on the multivalued planning task encoding [Helmert 2009] because of the possibility to reduce the number of variables (useful for constraint solvers). Pairs of variables and values are also incorporated in Cirillo et al. [2012] and Harrington and Cahill [2011]. Grześ et al. [2014] adopt a state-variable representation, but limit the type of variables to Boolean only. As for the actions, they may be represented as Boolean variables with preconditions and effects encoded as constraints on state variables (e.g., Kaldeli et al. [2012]).

Domain Definitions. Grześ et al. [2014] propose a knowledge engineering approach to create domain specifications based on a probabilistic relational model. A designer uses standard database tools, such as Web interfaces, to analyse a given domain and populate a relational database with the analysis results. Based on the relational model, a POMDP specification representing the corresponding planning problem is automatically generated. On the other hand, Ortiz et al. [2013] do not require explicit inputs from people to generate domain specifications. The approach segments sensor readings in order to recognise actions performed by users, and states produced by such actions. Action preconditions and effects are learned from those sensor readings, and a PDDL domain specification is automatically generated. Vaquero et al. [2015] gather data from graphical user interfaces and database, and transform that data into a PDDL domain specification using a knowledge engineering framework [Vaquero et al. 2013].

There are simpler and more limited ways of generating domain models than the ones just discussed. Hidalgo et al. [2011] realise a knowledge modelling tool that uses skills and experiences gathered from experts when they solved known problems. Georgievski et al. [2013] propose a domain modeller that provides intuitive guidance to users for creation, viewing, and modification of domain specifications.

Problem Definitions. It is common among the primary studies for domain specifications to be stored in some database and queried by planners upon initialisation or when necessary. In some cases, domains specifications may be enriched with additional semantic annotations needed for planning [Vukovic et al. 2007; Kaldeli et al. 2012; Madkour et al. 2013], or after planning and during plan instantiation [Bidot et al. 2011].

With respect to problem specifications, the context information is typically supplied to planners by other context-aware components. The current context information is automatically translated into an initial planning state using a standardised form (for an example, see Appendix B.3). Once generated, the initial state, which can be further maintained by planners, may automatically and dynamically incorporate future context changes (e.g., Kaldeli et al. [2012] and Courtemanche et al. [2008]). The goal is generated automatically from a request coming from a human or another system component.

Expressivity Constructs. We identify a collection of expressivity constructs that are suggested by some primary studies as needed for ubiquitous computing.

—*Numeric variables* to model variables with large domains (e.g., a variable for measuring temperature) [Kaldeli et al. 2012].
—*Multityping* to reduce the number of predicates and actions needed to be modelled. Multityping would enable one type to directly inherit from multiple types [Yordanova 2011].
—*Disjunction in preconditions* to represent action semantics compactly [Heider 2003].
—*Universal quantification in preconditions and effects* to enable actions deal with an arbitrary number of objects (e.g., an action with universal quantification can turn off all lamps in Theodore's home at once). Also, this construct may support the constant evolution and dynamic extension of environments with new objects [Heider 2003].
—*Conditional effects* in actions to provide compact representations of action semantics [Heider 2003], and to solve problems that involve moving objects [Marquardt and Uhrmacher 2009a].
—*Time and resources* are needed for most ubiquitous computing domains. Discrete and continuous resources may be necessary for some more specific domains [Heider 2003].
—*Extended goals* enable users to provide more powerful/personalised requests, making the environments more adaptive to user needs and also user centric [Vukovic et al. 2007; Kaldeli et al. 2012].

—*Axioms* to separate domain-specific knowledge from action semantics, and use that knowledge to derive new information (e.g., TV brightness increases when the brightness of the surrounding environment decreases) [Marquardt and Uhrmacher 2009a].

Languages. The majority of classified studies use PDDL as a modelling syntax for their domains. Several studies make use of HTN-based languages, such as the SHOP2 language in Song and Lee [2013] and Ha et al. [2005], and Hierarchical Planning Definition Language in Sánchez-Garzón et al. [2012] and Georgievski et al. [2013]; and two studies employ the Action Description Language [Kotsovinos and Vukovic 2005; Vukovic et al. 2007]. Others use nonplanning modelling languages, such as SMIL [Ding et al. 2006], Scone [Santofimia et al. 2010], XML [Kaldeli et al. 2012; Harrington and Cahill 2011], OWL-S [Bidot et al. 2011], and PRISM [Sukkerd et al. 2015]. The studies usually use the same language for states and goals as for domain specifications.

Monitoring and Recovery. In Qasem et al. [2004], the process of sensing involves searching for appropriate sources of information and using those to update the environment knowledge whenever there is missing information. The sensing implementation is based on local closed-world statements and relevance of information sources rather than using sensing actions. Avoiding sensing actions can reduce the search space by lowering the number of possible actions, and can decrease the knowledge complexity by not modelling each type of (query to) an information source as an action. Incorporation of new information when using CSP-based planning can also be accomplished without modelling and invoking sensing actions. Instead, some external data collecting process may periodically, or when some conditions are detected, provide most recent values of variables. These are then automatically incorporated into constraint networks by adding or removing constraints. Eppe and Bhatt [2015] consider explicit sensing actions, where different states are created for each possible outcome of the action.

Execution monitoring involves executing low-level invocations in right order and time, and monitoring and verifying the execution before and after the invocations. Given a plan to be executed, Algorithm 1 shows the common execution-monitoring steps taken in the primary studies.

The recovery process consists of the following stages. If preconditions cannot be verified (due to unexpected context changes), the precondition satisfaction can be delayed by inserting temporal constraints and reevaluating the preconditions later (e.g., Di Rocco et al. [2014]), or replanning is invoked (e.g., Vukovic et al. [2007]). If the outcome of an action invocation is a permanent failure, the plan is terminated and replanning for the same goal (e.g., Kaldeli et al. [2012]) or for a new goal is invoked (e.g., Ranganathan and Campbell [2004]). If the action outcome is another failure, the plan is repaired (e.g., Honold et al. [2014]), the action is re-invoked (e.g., Ranganathan and Campbell [2004]), or replaced with another instance of the same type (characteristic for approaches that build abstract plans [Vukovic et al. 2007; Bidot et al. 2011; Madkour et al. 2013]). If failure is observed again, replanning is invoked.

Several studies support the recovery process by using some predefined knowledge, such as conditional statements [Sando and Hishiyama 2011; Bajo et al. 2009; Sánchez-Garzón et al. 2012; Fraile et al. 2013]. Case-based planners support the recovery process by initiating a new planning cycle and considering already executed actions.

Lifecycle. We identify three types of planning considering the lifecycle of ubiquitous computing systems. *Design-time planning* is used to make strategic decisions and create abstract plans (e.g., Bidot et al. [2011]). Once an abstract plan is found, it is handled by another system component at runtime (abstract actions are instantiated by available behavioural outputs). In some cases, ubiquitous computing environments may require a time-bounded system, that is, a system that reacts in real time to every

ALGORITHM 1: Common Steps Taken by the Primary Studies During Execution Monitoring

```
Input: Plan
for each action in plan do
    Query the action to check its availability in the list of currently available actions;
    if action not available then
        call recovery process;
    end
    Check the validity of preconditions of the action;
    if precondition cannot be satisfied then
        call recovery process;
    end
    // Once all preconditions are satisfied, observe action effects
    if effects are unexpectedly satisfied // due to some exogenous event
    then
        avoid action execution;
    else
        execute the action and analyse its outcome;
    end
    if outcome is not expected then
        call recovery process;
    end
end
```

possible environment situation. It may be thus needed to shift the computationally expensive operations, including planning, at compile time. *Compile-time planning* may be useful if, for instance, a pregeneration of action sequences, or an early identification of modelling problems, such as deadlocks, is needed [Krüger et al. 2011]. *Runtime planning* is the conventional way of performing planning during system's runtime.

Design and Implementation. The designs of planning systems of primary studies share many similarities. A typical design consists of five components. A *Problem Generator* component accepts requests and context information, and generates problem specifications interpretable by a *Planner* component. The Planner, which implements a specific planning technique, takes problem specifications and acquires appropriate domain specifications from a *Knowledge Base* component. Given these specifications, the Planner finds a plan, if one exists, and passes it to an *Executor* component, which executes plan actions in the environment. The action execution is observed by a *Monitor* component, and upon deviations from the expected flow, it triggers a recovery process.

With respect to the implementation of planning systems, we identify three relevant pieces of information. First, the majority of studies claim to have implemented a prototype software of their proposal, however, with limited details on actual implementations. Second, most studies employ or extend existing planners for the Planner component, and third, only few studies implement new planners. For information on the planners employed by the primary studies, we refer to Appendix C.

Multiagent Systems. The classified primary studies typically design and implement multiagent systems by considering devices and other objects as agents, software components as agents, and a single planning agent. Amigoni et al. [2005] regard devices as simple agents that neither support context reasoning nor participate in planning. The other classified studies make a similar assumption: besides device agents, other agents can only extract and provide context information and domain knowledge to the planning agent that is solely responsible for reasoning and achieving the desired objectives. Pajares Ferrando and Onaindia [2013] employ multiagent planning: agents

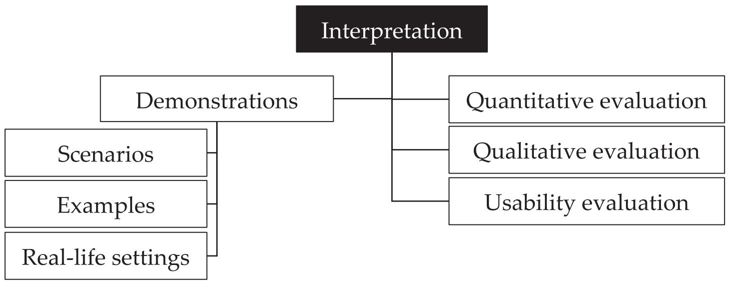

Fig. 3. Hierarchy of dimensions for interpretation.

exchange and support their decisions, interact with each other, and derive a joint plan as a solution to some planning problem [Weerdt and Clement 2009].

Modular Architectures. There are several cases in which modular architectures integrate planning as a separate module. Planning modules may be supported by the functionalities of other modules, such as a diagnosis module [Courtemanche et al. 2008] or actuator [Garro et al. 2008]. Modules communicate with each other through interfaces, which describe objects that are required and provided by the modules. This interoperability and modularity reduce the effort needed for modification and reuse of existing modules in new ubiquitous computing systems [Sánchez-Garzón et al. 2012].

Service-Oriented Architectures. We identify two types of services in the primary studies, depending on their role in ubiquitous computing systems. *Ubiquitous services* are domain specific and represent the functionalities of behavioural outputs (e.g., devices to sense and act). The primary studies often implement ubiquitous services as OWL-S services. *Application services* represent and encapsulate the capabilities of ubiquitous computing systems, and are used to implement system workflows. Among these, *planning services* hide the implementation details of the capabilities of planning systems. For example, Georgievski et al. [2013] implement a limited set of planning services as Representational State Transfer (REST) resources [Fielding and Taylor 2002]. Planning services may be used to create domain and problem specifications, to solve planning problems, etc. [Kaldeli et al. 2012; Georgievski et al. 2013].

6. INTERPRETATION

For the interpretation dimension, we catalogue four subdimensions, namely, demonstrations, quantitative evaluation, qualitative evaluation, and usability evaluation. Only the demonstrations dimension is further split into subdimensions. Figure 3 gives an overview of the hierarchy of these dimensions.

6.1. Demonstrations

Demonstrations encompass the ways used to illustrate planning problems under consideration, and to evaluate the feasibility of adopted planning techniques. The common ways of demonstration are scenarios, examples, and real-life settings.

—*Scenarios* are synoptic descriptions of people's and system's actions and events in ubiquitous computing environments. Scenarios are powerful illustrations of the difficulty of problems and their solutions. Scenarios should help people have a sufficiently wide view about proposed ideas so as to avoid missing important attributes of corresponding planning problems [Alexander and Maiden 2004]. Nevertheless, scenarios are the starting point of all modelling and design [Sutcliffe 2003].

—*Examples* are short descriptions used to support and clarify what is introduced and meant. Examples can be descriptive (included in the text), or examples can be represented in a chosen syntax. The latter may include excerpts from state representations, goal examples, parts of domain knowledge, and examples of plans. Examples are the most common way used to demonstrate planning problems in ubiquitous computing.

—*Real-life settings* extend research beyond the limits of scenarios and examples into the real world. Real-life settings provide details on the conditions relevant for the application and use of planning in actual ubiquitous computing environments, and used to evaluate the feasibility of approaches.

6.2. Quantitative Evaluation

Quantitative evaluation focuses on characterising the feasibility of approaches by evaluating the performance of adopted planners. Evaluations need to be carried out using transparent strategies in order to understand how planners meet the requirements of ubiquitous computing environments. One can think of a strategy for quantitative evaluation in terms of the following elements derived from the primary studies.

—*Testing configuration* defines the setup of hardware and software on which planners are tested. The setup consists of a physical setup, which involves one or more computers together with a description of their specifications, sensors, and other hardware required for the testing; and a logical setup, which includes the operating system, database servers, libraries, and other software components.

—*Algorithmic configuration* defines the setup of algorithms used to evaluate planners. Algorithms are often configured with different runtime properties (some specific feature or a certain mode). The cases in which such configurations are being used must be explicitly noted as evaluation results depend on those configurations.

—*Problem base* requires basic details on planning problems used to test planners. One can understand better the testing by having insights into the domain specification, such as domain types, predicates, actions, and their respective numbers, and the problem specification, including objects, initial predicates, and the goal.

—*Computational factors* focus on the properties of planning problems that influence the computational efficiency of planners. In general, the computational complexity is a function of the number of domain actions, predicates, and other planning-related entities. Additional computational factors are related to the assumptions made about planning problems. The restrictive assumptions of classical planning problems create *ideal conditions*, while uncertainty interferes with these assumptions and produces *faulty conditions* of ubiquitous computing environments, which make planning an even more computationally expensive task.

—*Scalability* is related to the computational factors and is often defined with respect to the *size of a domain* and the *size of a plan*. Worst-case analysis is often useful for characterising the performance of planners.

6.3. Qualitative Evaluation

Qualitative evaluation assesses the quality of plans. We identify two general ways of qualitative evaluation: one answering the question of *how well a plan is created in relation to some specific parameters*, and the other one seeking an answer to the question of *how well a plan is created in comparison to plans of other approaches*. Qualitative evaluation might be subjective as it may be based primarily on opinions drawn from subjective choices of evaluation parameters or from observations.

6.4. Usability Evaluation

Ubiquitous computing systems, including adopted planners, can bring benefits to environments only if they are actually used. According to the ISO 9241-11 standard on ergonomics of human-system interaction, *usability* is "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use."[5] The accuracy and completeness with which users achieve goals while not experiencing any negative consequences define the effectiveness. Efficiency is defined by the relation between what has been achieved and what resources have been consumed. The satisfaction of users refers to their positive attitudes resulting from the use of some product. According to the Technology Acceptance Model, users will have positive attitudes if they perceive the product as easy to use and helpful at the same time [Davis 1986]. Thus, positive attitudes lead to ubiquitous computing systems being actually used.

Usability testing provides a fundamental method to evaluate the usability of any system [Wichansky 2000], and it should be an essential phase in the development of ubiquitous computing environments [Kim et al. 2003]. *Usability evaluation* thus deals with how the usability of planners for ubiquitous computing is tested and evaluated.

6.5. Review of Primary Studies

Table V shows the arrangement of the primary studies according to the subdimensions of the interpretation dimension.

Demonstrations. Almost half of the primary studies use scenarios to illustrate their problem of interest, more than half use examples to describe the problem of interest and planning problems, while only nine studies employ their systems in real-life settings. Several studies combine all three ways.

—The scenarios are often from the domain of homes, others being from offices, hospitals, and infotainment systems. Scenarios focus on either a specific use case and solve a single planning problem, or various use cases and address multiple problems. All scenarios are human centric, that is, characterise situations from the perspective of persons involved explicitly or implicitly.
—Textual and syntactic examples are both prevalent among the primary studies. Textual examples are used to explain concepts from both ubiquitous computing and automated planning. Syntactic examples are generally used to introduce planning concepts. Actions are the most exemplified concepts in a chosen syntax. Numerous are the syntactic examples of plans, both totally ordered and partially ordered. While states and goals are also exemplified, the syntactic examples are often partial only.
—Table VI shows details on real-life settings used mainly for exploratory purposes. The most frequent environments are homes, while a hospital has the largest number of participants (30 patients and 6 nurses). We identify the duration of experimentation in five studies, the longest being 13 months. Almost all studies provide clear descriptions of involved entities, such as devices and locations. Interestingly, Fraile et al. [2013] consult experts to create the domain knowledge for the real-life experiment.

Quantitative Evaluation. Some primary studies reveal their testing configuration, which typically includes the processing power and RAM of the computer used for the evaluation. Additionally, the testing configuration may specify the operating system and runtime environment, such as the Java virtual machine. Few studies provide details on the algorithmic configuration by including the version of a specific tool [De Giacomo et al. 2012], type of a search mechanism [Kotsovinos and Vukovic 2005; Kaldeli

[5]https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:dis:ed-2:v1:en.

Table V. The Dimensions of Primary Studies within the Interpretation Dimension

Study	Demonstrations			Quantitative eval.	Qualitative eval.	Usability eval.	Study	Demonstrations			Quantitative eval.	Qualitative eval.	Usability eval.
	Scenarios	Examples	Real-life settings					Scenarios	Examples	Real-life settings			
S1							S28	✘		✘			
S2	✘	✘		✘			S29	✘	✘	✘			
S3	✘	✘		✘			S30	✘	✘				
S4	✘	✘					S31		✘				
S5	✘	✘					S32		✘		✘	✘	
S6	✘	✘		✘	✘		S33		✘		✘	✘	
S7	✘	✘					S34		✘				
S8		✘			✘		S35	✘	✘		✘		
S9			✘	✘	✘	✘	S36	✘	✘				
S10		✘					S37						
S11		✘		✘			S38	✘					
S12		✘		✘			S39	✘	✘				
S13		✘					S40						
S14	✘			✘			S41						
S15		✘					S42			✘		✘	✘
S16			✘			✘	S43			✘			✘
S17							S44	✘					
S18	✘	✘		✘		✘	S45		✘		✘		
S19	✘	✘					S46	✘					
S20		✘		✘			S47	✘	✘				
S21	✘	✘		✘	✘		S48	✘	✘		✘		
S22		✘	✘		✘		S49				✘		
S23		✘	✘				S50			✘			
S24		✘					S51		✘				✘
S25	✘	✘					S52	✘			✘		
S26		✘					S53	✘			✘	✘	
S27		✘											

Table VI. Real-Life Settings of Ubiquitous Computing Environments in Some Primary Studies

Study	Environment	Number of participants	Experiment duration
S9	Hospital	36 people	
S16	University laboratory	10 people	3 partial days
S22	Home	1 inhabitant	1 month
S23	Home	1 inhabitant, 1 robot	
S25	Care centre	7 persons with dementia	
S28	Home	2 robots	
S29	Home	1 person, 1 robot	3 hours
S42	Shopping mall	30 people	13 months
S43	College	13	1 day
S50	Home	1 inhabitant, 1 person	

et al. 2012; Chen et al. 2016], and other specific features (e.g., the SUM heuristic in Pajares Ferrando and Onaindia [2013]). For the problem base, the primary studies provide only partial information on domain specifications often including examples of types, predicates, and actions. A domain size is commonly expressed by the number of actions (e.g., Pajares Ferrando and Onaindia [2013]), and the number of predicates or variables (e.g., Kotsovinos and Vukovic [2005]). Regarding problem specifications, the classified studies provide limited details about the constituents and structure of initial states and goals. The computational complexity of planning problems depends on the (1) size of initial states, including the number of devices, robots, people, and rooms; (2) goal size in terms of the number of literals or tasks; (3) number of actions in plans; (4) number and type of failures during plan execution; (5) structure of domain and problem specifications; and (6) number of concurrent goals. Scalability is commonly analysed by varying the number of state constituents and/or goal constituents within some intuitive ranges. For example, Vukovic et al. [2007] vary the number of service instances between 80 and 640, and Georgievski et al. [2013] vary the number of goal tasks between 5 and 50. The problem difficulty is often illustrated by the number of actions in resulting plans. Considering the example of Vukovic et al. [2007], the size of resulting plans varies between 9 and 80. Naturally, the performance of planners degrades with the scaling factor. Some primary studies focus further on the worst-case scenario and analyse the relationship between the result and the complexity factor (e.g., Mastrogiovanni et al. [2010]).

Qualitative Evaluation. To evaluate their approaches qualitatively, the primary studies use various parameters, such as goal specification [Vukovic et al. 2007], duration of plans [Courtemanche et al. 2008], success of plans [Bajo et al. 2009; Corchado et al. 2009; Chen et al. 2016], length of plans [Pajares Ferrando and Onaindia 2013], some reasoning patterns [Milani and Poggioni 2007], cost of plans [Pajares Ferrando and Onaindia 2013; Fraile et al. 2013; Georgievski et al. 2013], etc. For example, the goal specification parameter takes one of three values, namely, manual, semiautomated, and automated from a development perspective. On the other hand, some studies make qualitative comparisons between their approaches and others. Other approaches include legacy systems [Vukovic et al. 2007], state-of-the-art solutions [Pajares Ferrando and Onaindia 2013; Chen et al. 2016], or systems already in place [Georgievski et al. 2013]. While some studies elaborate their comparison methodology and results, others leave undefined the way of assigning values to the evaluation parameters.

Usability Evaluation. A small number of primary studies consider and evaluate their approaches with respect to usability. The most common steps taken by these studies to test and evaluate usability are as follows.

—*Determine users*. Planning systems may have several distinct user groups each of which has its own goals and varying levels of effectiveness, efficiency, and satisfaction. For example, Kaldeli et al. [2012] determine a group of elderly and disabled people, and a group of young, technologically savvy people; Bajo et al. [2009] identify nurses as a targeted user group; Corchado et al. [2009] select users that use Wi-Fi and Bluetooth on their mobile devices; and Bacon et al. [2013] target three groups based on training experiences of participating users.

—*Determine user goals*. The selection of user goals important for given situations is a difficult problem in itself. Kaldeli et al. [2012] define their user goals using acceptability, learnability, system effectiveness, and efficiency. Acceptability comprises the attitude of users towards the importance of domotic technology, automation of tasks, and privacy. Sando and Hishiyama [2011] allow users to score importance levels for involved items. Kaldeli et al. [2012] assess learnability by the amount of effort users must expend to understand the functionalities of their system, and to be able to use it. Effectiveness is an aggregate of virtual environment effectiveness, user interface effectiveness, and the support for complex goals. In Bajo et al. [2009] and Corchado et al. [2009] effectiveness is measured using the relation between the satisfaction of users with plans and the quality of plans being improved over time. Sando and

Hishiyama [2011] evaluate effectiveness using a correlation between the level of importance of items and the level of user satisfaction with those items. In Kaldeli et al. [2012], the efficiency is measured according to the user's assessment of the time required to complete simple operations and complex goals.

—*Determine the context of use*. The context of use is determined by the "diverse requirements, abilities and technological knowledge" of users within homes [Kaldeli et al. 2012]; diverse profiles of nurses, patient needs, and their technological knowledge [Bajo et al. 2009]; ingredients [Sando and Hishiyama 2011]; products of interest, time, and technological knowledge [Corchado et al. 2009]; crisis situations, behavioural profiles of trainees, and technological knowledge [Bacon et al. 2013].

—*Determine the levels of importance, effectiveness, efficiency, and satisfaction*. This is also challenging as it requires one to determine "right" levels, but also a crucial step as it defines the actual usability. The studies use scales with different levels: 0–4 [Kaldeli et al. 2012], 0–5 [Sando and Hishiyama 2011], and 0–6 [Bacon et al. 2013].

7. DIRECTIONS FOR FUTURE RESEARCH

The framework of dimensions foregrounds the possibility to evaluate the progress of automated planning for ubiquitous computing. The analysis of primary studies per dimension leads us to the following possible directions for future research.

—*Extended forms of procedural goals*: Enhancing the conventional representation of procedural goals as tasks of HTN planning to allow people to specify additional or personal information, such as maintainability properties and preferences over tasks.

—*Preferences*: While preferences can be a powerful tool to empower people to customise their environments, they are one of the least treated topics.

—*Human-computer interaction*: Though central to ubiquitous computing, the topic of human-computer interaction is not mentioned at all in two-thirds of the primary studies. Less than one-third of the studies mention that the support for human-computer interaction is provided in the form of user interfaces implemented on PDAs, smartphones, and Web browsers. These studies, however, do not report on details whether and how such support is related to planning. Only a few studies specify their approach to human-computer interaction. (From their qualitative analysis, we could not identify shared attributes or patterns that would allow us to organise these studies into a separate dimension. Therefore, a few aspects of human interactions can be seen from a perspective of other dimensions.)

—*Pure spatial representations*: Abstract spatial representations, as a typical way of representing spatial properties, are associated with the issue of realisability (see Section 4.3). Realisability can be achieved by pure spatial representations.

—*Fine-grained spatial properties for humans*: Besides the information on locations of humans, many situations in ubiquitous computing environments (e.g., emergencies) require other more fine-grained properties, such as human posture and orientation.

—*Categories of unexpected events*: Our recognition of dynamic goals and context changes as two categories of unexpected events can serve as a basis for further identification and definition of categories, which can be used to improve the capabilities of planning systems when dealing with dynamic ubiquitous computing environments.

—*Reasons for action contingencies*: Finding out why action contingencies occur can lead to improved characterisation and definition of processes for planning and plan execution. For example, the main reason for action timeouts often lies in the disconnection or failure of networks that action providers belong to. Knowing this, the model of these timeouts and semantics of action (re-)execution can be defined appropriately.

—*Conditions for partial observability*: Define and extend the conditions under which the information sensed from ubiquitous computing environments is valid.

—*Limitations of planning techniques*: Several planning techniques used for ubiquitous computing are characterised by some well-known or less-known limitations. Kaldeli et al. [2012] criticise HTN planning due to the requirement for predefined methods that cannot be easily reconfigured when changes in the context or domain occur. Marquardt et al. [2008] recognise a critical point in the use of HTN planning due to its need for domain-specific knowledge. This may come to light in reality when the responsibility for providing knowledge is transferred to manufacturers of devices or ubiquitous computing systems. The main difficulty with POMDP planning is scaling up: probabilities create belief states that are continuous and infinite. It is also known that complete CSP algorithms may take a very long time to solve the inconsistencies in constraint networks [Barták et al. 2010]. Similarly, partial-order planners require substantial computational power to search for plans due to the inherent complexity of their algorithms. Finally, case-based reasoning can be "highly affected" by context changes [Bajo et al. 2009]. The success of planning depends, among other things, on the environment changes that happen during plan execution. Unexpected changes may lead to replanning.

—*Human-aware plans*: Ubiquitous computing envisions that devices, robots, and humans will coexist side by side and interact among each other in one way or another. Planning systems should therefore be able to produce plans that satisfy the given goals while taking into account the activities and constraints under which people perform those activities. These constraints can involve, for example, safety conditions, comfort conditions, or activity-related conditions [Cirillo et al. 2012].

—*Knowledge engineering*: The formulation of domain knowledge for planning in ubiquitous computing (and in general) is a strenuous, tedious, and error-prone task that also requires expertise from the designer of the respective domain. Ideally, this task should be performed automatically by acquiring information from ubiquitous computing environments themselves and translate it into domain knowledge. While this is an existing research topic in automated planning (see Shah et al. [2013]), the support for engineering ubiquitous computing knowledge for planning is scarce.

—*Semantics and algorithms for plan execution*: What is interesting, but currently missing is a formalisation of execution semantics such that defines valid repairs of plans at execution time, and sound and complete algorithms for monitoring and plan repairs.

—*Standardisation of planning systems*: If planning systems are going to be integrated in large (ubiquitous computing) systems, they need to satisfy some standard requirements of such systems, including interoperability, reusability, evolution, scalability, distribution, and fault tolerance [Degeler et al. 2013]. For example, interoperability of a planning system is only possible under the assumption that the system offers a standard interface or a set of planning services to other components of potential ubiquitous computing systems. The current situation witnesses planning systems with no standardisation of interfaces and services, while the obvious consequence is the strong need for familiarity with details of planning systems.

—*Planning in real-life settings*: The limited use of planning systems in actual environments narrows the understanding about usefulness, benefits, and possible open issues in ubiquitous computing. In addition, with the exception of Corchado et al. [2009], the duration of deployments of planning systems in actual environments is usually very short and without indications of whether planning can be put into long-term use.

—*Evaluating planning systems*: Understanding the behaviour of planning systems in terms of their performance, comparative analysis, and usability is crucial for determining their effectiveness, usefulness, and acceptance by people and in ubiquitous computing; and for identifying possible directions for improvement.

More generally, our framework of dimensions can be used to further characterise planning for ubiquitous computing on a conceptual level. Besides a set of concepts, which our framework provides through its dimensions, conceptualisation gathers and models the relationships among those concepts [Thalheim 2010]. A conceptual model may provide more and new insights into planning for ubiquitous computing, and can be used as an artefact to be further analysed and communicated among different audiences. Additionally, little is known about how difficult it is to solve planning problems in ubiquitous computing, or the amount of resources the computation for solutions requires. Complexity analysis may provide insights into the answers to these questions.

We also identify two points of attention for when defining, designing, and developing future approaches based on planning for ubiquitous computing.

—*Definition of problems being solved*: Planning problems are intended to model problems originating from ubiquitous computing. On the one hand, the current situation witnesses unclear and ambiguous descriptions of ubiquitous computing problems being addressed. On the other hand, their corresponding planning problems are also often undefined (more that 50% of primary studies only mention and just a few define the planning problems). Being formalised, both ubiquitous computing problems and planning problems provide means for developing well-defined ubiquitous computing systems, allowing for improved maintainability, ability to evolve, reusability, consistency, and sound reasoning [Bettini et al. 2010].

—*Formal correspondence between problems*: A formal correspondence between ubiquitous computing problems and planning problems would allow for a sound use of plans computed for planning problems as solutions to the ubiquitous computing problems.

8. CONCLUSION

Automated planning will have an increasing role as ubiquitous computing is becoming a reality for a great deal of human-based environments. In the present treatment, we focus on systematising and evaluating the current state of the art. We adopted a rigorous methodology to find a full range of relevant literature and to extract qualitative information from the literature selected. We used that information to develop a framework of dimensions around which we articulated our analysis. This analysis facilitated the evaluation of the current progress in terms of recognising directions and points of attention for future research.

With respect to our introductory questions, our work indicates that there is a general challenge inherent in the current literature that should be considered by future research, most notably, the field needs precise definitions of planning problems being solved and transparent modelling of those problems. Another finding is that the majority of literature uses planning systems off the shelf, and that when using and integrating planning systems into ubiquitous computing systems no particular design considerations are enforced. The theoretical progress of automated planning for ubiquitous computing tends to predominate the practical progress, though several studies already demonstrate the use of planning in actual ubiquitous computing environments.

APPENDIXES

B. THEODORE'S HOME THROUGH PDDL ENCODINGS

We provide samples of encodings of some aspects of the scenarios in Theodore's home introduced at the beginning of the article. Most of the encodings are adopted from the primary studies and modified to fit our purposes and be compliant with PDDL.

B.1. Domain Types and Predicates

The home of Theodore has various entities that can be classified by some shared characteristics. For example, the TV, lamps, ventilator, and computer are all devices. PDDL enables representing this kind of information in the domain definition under the `:types` tag. Figure 4 shows PDDL types encoding some of the entities in Theodore's home.

```
(:types ingredient device room slide surface restaurant ... - object
        persons time ... - number
        TV lamp computer door window blinds ventilator ... - device
        oil water salt egg vegetable ... - ingredient
        onion cucumber tomato ... - vegetable
        ...
)
```

Fig. 4. PDDL types representing classes of some of the entities in Theodore's home.

The relations between different entities and the properties of the home itself are described by predicates. A basic PDDL predicate has the form `(NAME ?A1 ... ?An)`, where the arguments beginning with a question mark are parameters. The predicate name defines the relation or property of arguments. For example, `(adjacent ?r1 ?r2)` indicates that rooms `?r1` and `?r2` are adjacent and connected by a door. Additionally, in its definition, a PDDL predicate may contain the type of each argument. For example, `(adjacent ?r1 ?r2 - room)`. Figure 5 shows definitions of PDDL predicates for some of the relations and properties of the home of Theodore.

```
(:predicates (adjacent ?r1 ?r2 - room)
             (in ?d - device ?r - room)
             (at ?b - blinds ?w - window)
             (in ?p - person ?r - room)
             (on ?o - object ?s - surface)
             (turned ?d - device)
             (pulled-up ?b - blinds)
             (opened ?d -  door)
             (holding ?o - object)
             (quantity ?v - ingredient ?n - number)
             (channel ?tv - TV ?ch - channel)
             (dirt ?r - room)
             (ppt ?id - string ?mac - computer)
             (started ?id - string ?f - pptfile - ?s- slide)
             (found ?r - restaurant)
             (booking-online ?r - restaurant)
             (has-space ?r - restaurant ?p - persons)
             (booking-made ?r - restaurant ?t - time)
             (booked ?r - restaurant)
             ...
)
```

Fig. 5. PDDL predicates representing some of the properties of Theodore's home.

B.2. Domain Actions

A single home like Theodore's one may contain a large number of behavioural outputs: device operations, human actions, robot actions, application services, and information services. All these are modelled as actions in PDDL. In the following sections, we provide examples of each type of behavioural output.

B.2.1. Device Operations. Figure 6 shows a template action for turning on any device in Theodore's home that supports this operation. Examples include TV, lamps, radio, boiler, computer, etc. Figure 7 shows a simple encoding of an action for closing any of the doors in Theodore's home. The preconditions can include additional checks on whether there are any observable obstacles that may prevent a door from closing. Figure 8 illustrates a PDDL action for opening a window given that its blinds are pulled up so that the window can be opened inward. Finally, Figure 9 shows a PDDL action for setting up a specific channel to the TV in Theodore's living room, if that is the only one.

```
(:action turn-on-device
 :parameters (?d - device)
 :precondition (and (not (turned ?d)) (other_cond))
 :effect (turned ?d))
```

Fig. 6. PDDL action for a device operation that turns on a device.

```
(:action close-door
 :parameters (?d - door)
 :precondition (and (opened ?d) (other_cond))
 :effect (not (opened ?d)))
```

Fig. 7. PDDL action for closing a door adopted and adapted from De Giacomo et al. [2012].

```
(:action open-window
 :parameters (?w - window)
 :precondition (and (not (opened ?w)) (blinds ?b) (at ?b ?w)
                    (pulled-up ?b) (other_cond))
 :effect (opened ?w))
```

Fig. 8. PDDL action for opening a window.

```
(:action set-tv-channel
 :parameters (?tv - tv ?ch - channel)
 :precondition (and (turned ?tv) (channel ?tv ?c) (not (= ?c ?ch)))
 :effect (not (channel ?tv ?c)) (channel ?tv ?ch))
```

Fig. 9. PDDL action for setting a TV channel and adapted from Kaldeli et al. [2012].

B.2.2. Human Actions. We show examples of two human actions that can be used to guide Theodore towards accomplishing some objective. Figure 10 illustrates a PDDL action that encodes picking up some object by a human. The action's preconditions require the object to be on some surface before it can be picked up and held by the human. Figure 11 shows a template action that can be used to instruct Theodore to move some object from one place to another, assuming that the places are different and he has already executed the `pick-up` action.

```
(:action pick-up
 :parameters (?o - object ?s - surface)
 :precondition (and (on ?o ?s))
 :effect (and (not (on ?o ?s)) (holding ?o)))
```

Fig. 10. PDDL action representing a human action of picking up an object adopted and adapted from Ortiz et al. [2013].

```
(:action move
:parameters (?o - object ?r1 ?r2 - room)
:precondition (and (in ?o ?r1)
                   (not (in ?o ?r2)) (not (= ?r1 ?r2)) (holding ?o))
:effect (and (in ?o ?r2) (not (in ?o ?r1))) (moved ?o)))
```

Fig. 11. PDDL action encoding a human action for moving an object adopted and adapted from Yordanova [2011].

B.2.3. Robot Action. Figure 12 shows a PDDL action encoding a robot action for cleaning some room (the original representation involves durations too) [Cirillo et al. 2012]. The action requires a room to be dirty given by the fact that dirt is quantified and greater than 0. The action's effect involves decreasing the value of dirt by 1 and increasing the cost of performing this action by 2. The dirt and cost are both domain functions, which are state variables whose values can be updated as needed (for details, see Fox and Long [2003]).

```
(:action clean
 :parameters (?r - room)
 :precondition (and (> (dirt ?r) 0))
 :effect (and (decrease (dirt ?r) 1) (increase (cost) 2))
```

Fig. 12. PDDL action representing a robot action for cleaning a room adopted and adapted from Cirillo et al. [2012].

B.2.4. Application Service. Figure 13 shows a PDDL action encoding a service that sets a presentation file on a computer using the Microsoft PowerPoint application. Preconditions specify the computer and that the presentation file displayed on that computer should be different from the one in the input parameter. The effect ensures that the current file is started on the first slide. This action can be used, for example, to show the recipe that Samantha has computed for the dish chosen by Theodore.

```
(:action set-pptfile
 :parameters (?id - string ?f - pptfile)
 :precondition (and (computer ?mac) (slide ?s) (pptfile ?f1)
                    (ppt ?id ?mac) (started ?id ?f1 ?s) (not (= ?f ?f1)))
 :effect (and (ppt ?id ?mac) (started ?id ?f 1))
```

Fig. 13. Example of an application service for setting up a presentation adopted and adapted from Ranganathan and Campbell [2004].

B.2.5. Information Service. Figure 14 shows a PDDL action encoding an information service that can be used by Samantha to book a restaurant for Theodore's dinner at a particular time slot of the day.

```
(:action book-restaurant
 :parameters (?r - restaurant ?p - persons ?t - time)
 :precondition (and (found ?r) (booking-online ?r) (has-space ?r ?p)
                    (not (booking-made ?p ?t)) (not (booked ?r)))
 :effect (and (booking-made ?p ?t)(booked ?r))
```

Fig. 14. PDDL action representing an information service for booking a restaurant adopted and adapted from Vukovic et al. [2007].

B.3. Initial State

The current state of Theodore's home gives information on the actual layout of the home, available devices, their arrangement within the home, the statuses of all devices, and other relations and properties. This snapshot of Theodore's home can be described in the initial state of a PDDL problem definition, particularly under the `:init` tag. The initial state consists of ground predicates, that is, predicates whose parameters have assigined specific values. Figure 15 shows a PDDL description of an initial state of Theodore's home at some point in time.

```
(:init (room livingRoom)
       (room kitchen)
       (room bedroom)
       (room bathroom)
       (room toilet)
       (door d1)
       (door d2)
       (door d3)
       (door d4)
       (door d5)
       (window w1)
       (window w2)
       (window w3)
       ...
       (blinds b1)
       (blinds b2)
       ...
       (TV tv1)
       (TV tv2)
       (ventilator v1)
       ...
       (vegetable onion)
       (vegetable tomato)
       ...
       (adjacent livingRoom kitchen)
       (adjacent bedroom bathroom)
       ...
       (in w1 kitchen)
       (at b2 w1)
       (in v1 kitchen)
       ...
       (opened d1)
       (opened d3)
       (opened w1)
       (pulled-up b1)
       (turned tv1)
       ...
       (channel tv1 CH5)
       (quantity onion 3)
       ...
)
```

Fig. 15. PDDL ground predicates representing an initial state of Theodore's home.

B.4. Extended Goal

Recall the gas leakage scenario in Theodore's home. An extended goal for such a situation is shown in Figure 16. The `achieve-maint` indicates that the kitchen ventilator must be turned on, the TV must show a warning, and the window in the kitchen needs to be opened in some intermediate state and stay satisfied until the final state. `achieve-final` defines that the respective expression may hold or not in intermediate states, but it has to be satisfied in the final state. Finally, `under_cond_or_not` indicates that the goal under `achieve-final` will be satisfied if Theodore is not in the kitchen; however, if he is in the kitchen, then only the rest of the conjunction will be dealt with.

```
achieve-maint(and (turned ventilator) (in ventilator kitchen)
                  (alarm TV)(opened window)(in window kitchen)) and
achieve-final(and (not (opened door))(adjacent livingRoom kitchen)))
under_cond_or_not achieve-maint(not (in THEODORE kitchen))
```

Fig. 16. Extended goal for dealing with gas leakage adopted and adapted to PDDL (goal constructs are not part of PDDL) from Kaldeli et al. [2012].

C. PLANNERS EMPLOYED BY PRIMARY STUDIES

Many of the primary studies use already existing planners to implement their approaches, while only a few studies develop their own planners. Table VII shows the planners employed by the primary studies together with planners' references.

Table VII. Planners Employed by Primary Studies Together with Planners' References

Study	Planner	Planner's reference
S2	Blackbox	Kautz and Selman [1999]
S3	TLPlan	Bacchus and Kabanza [1996]
S4	NOAH	Sacerdoti [1975]
S5	JSHOP2	Nau et al. [2003]
S6	TLPlan	Bacchus and Kabanza [1996]
S9	CBPMP	S9
S10	JSHOP2	Nau et al. [2003]
S15	FF	Hoffmann and Nebel [2001]
S17	SIADEX	Castillo et al. [2006]
S18	RuGPlanner	S18
S19	SHOP2	Nau et al. [2003]
S20	SIADEX	Castillo et al. [2006]
S21	CAMAP	S21
S23	SHOP2	Nau et al. [2003]
S26	SHOP2	Nau et al. [2003]
S27	FF	Hoffmann and Nebel [2001]
	LPG	Gerevini and Serina [2002]
	MIPS	Edelkamp and Helmert [2001]
S28	Configuration	Di Rocco et al. [2013]
S33	SH	S33
S35	LPG	[Gerevini and Serina 2002]
S36	GraphPlan	Blum and Furst [1997]
S44	JSHOP2	Nau et al. [2003]
S47	FD	Helmert [2006]
S50	Configuration	Di Rocco et al. [2013]
S52	OPTIC	Benton et al. [2012]

ACKNOWLEDGMENTS

We thank Ian-Pratt Hartmann for the feedback on spatial representation and reasoning.

REFERENCES

Marco Aiello, Ian Pratt-Hartmann, and Johan van Benthem. 2007a. *Handbook of Spatial Logics*. Springer.

Marco Aiello, Ian Pratt-Hartmann, and Johan Van Benthem. 2007b. What is spatial logic? In *Handbook of Spatial Logics*. Springer, 1–11.

Ian Alexander and Neil Maiden. 2004. *Scenarios, Stories, use Cases: Through the Systems Development Life-cycle*. John Wiley & Sons.

James F. Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26, 11 (1983), 832–843.

Giuseppe Amato, Davide Bacciu, Mathias Broxvall, Stefano Chessa, Sonya A. Coleman, Maurizio Di Rocco, Mauro Dragone, Claudio Gallicchio, Claudio Gennaro, Héctor Lozano Peiteado, T. Martin McGinnity, Alessio Micheli, A. K. Ray, Arantxa Rentería, Alessandro Saffiotti, David Swords, Claudio Vairo, and Philip J. Vance. 2015. Robotic ubiquitous cognitive ecology for smart homes. *Journal of Intelligent and Robotic Systems* 80, Supplement-1 (2015), 57–81.

Francesco Amigoni, Nicola Gatti, Carlo Pinciroli, and Manuel Roveri. 2005. What planner for ambient intelligence applications? *IEEE Transactions on Systems, Man and Cybernetics, Part A* 35, 1 (2005), 7–21.

Hajnal Andréka, Judit X. Madarász, and István Németi. 2007. Logic of space-time and relativity theory. In *Handbook of Spatial Logics*. Springer, 607–711.

Fahiem Bacchus and Froduald Kabanza. 1996. Using temporal logic to control search in a forward chaining planner. In *New Directions in AI Planning*, Malik Ghallab and Alfredo Milani (Eds.). IOS Press, 141–153.

Liz Bacon, Lachlan M. MacKinnon, Amedeo Cesta, and Gabriella Cortellessa. 2013. Developing a smart environment for crisis management training. *Journal of Ambient Intelligence and Humanized Computing* 4, 5 (2013), 581–590.

Javier Bajo, Juan F. de Paz, Yanira de Paz, and Juan M. Corchado. 2009. Integrating case-based planning and RPTW neural networks to construct an intelligent environment for health care. *Expert Systems with Applications* 36, 3 (2009), 5844–5858.

Roman Barták, Miguel A. Salido, and Francesca Rossi. 2010. Constraint satisfaction techniques in planning and scheduling. *Journal of Intelligent Manufacturing* 21, 1 (2010), 5–15.

Sandrine Beauche and Pascal Poizat. 2008. Automated service composition with adaptive planning. In *International Conference on Service-Oriented Computing (ICSOC'08)*. Springer-Verlag, 530–537.

Mounir Beggas, Lionel Médini, Frederique Laforest, and Mohamed Tayeb Laskri. 2013. Fuzzy logic based utility function for context-aware adaptation planning. In *Modeling Approaches and Algorithms for Advanced Computer Applications*, Abdelmalek Amine, Ait Mohamed Otmane, and Ladjel Bellatreche (Eds.). Studies in Computational Intelligence, Vol. 488. Springer, 227–236.

Johan van Benthem. 1983. *The Logic of Time: A Model-theoretic Investigation into the Varieties of Temporal Ontology and Temporal Discourse*. Springer.

J. Benton, Amanda J. Coles, and Andrew Coles. 2012. Temporal planning with preferences and time-dependent continuous costs. In *International Conference on Automated Planning and Scheduling*. 2–10.

Piergiorgio Bertoli, Raman Kazhamiakin, Massimo Paolucci, Marco Pistore, Heorhi Raik, and Matthias Wagner. 2009. Continuous orchestration of Web services via planning. In *International Conference on Automated Planning and Scheduling (ICAPS'09)*. 18–25.

Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, and Daniele Riboni. 2010. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing* 6, 2 (2010), 161–180.

Claudio Bettini and Daniele Riboni. 2015. Privacy protection in pervasive systems: State of the art and technical challenges. *Pervasive and Mobile Computing* 17, Part B, 0 (2015), 159–174.

Julien Bidot and Susanne Biundo. 2011. Artificial intelligence planning for ambient environments. In *Next Generation Intelligent Environments*, Wolfgang Minker and Tobias Heinroth (Eds.). Springer, 195–225.

Julien Bidot, Christos Goumopoulos, and Ioannis Calemis. 2011. Using AI planning and late binding for managing service workflows in intelligent environments. In *International Conference on Pervasive Computing and Communications*. IEEE, 156–163.

Zeungnam Zenn Bien, Hyong-Euk Lee, Jun-Hyeong Do, Yong-Hwi Kim, Kwang-Hyun Park, and Seung-Eun Yang. 2008. Intelligent interaction for human-friendly service robot in smart house environment. *International Journal of Computational Intelligence Systems* 1, 1 (2008), 77–94.

Avrim L. Blum and Merrick L. Furst. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90, 12 (1997), 281–300.

Blai Bonet and Héctor Geffner. 2001. Planning as heuristic search. *Artificial Intelligence* 129, 12 (2001), 5–33.

Craig Boutilier, Thomas Dean, and Steve Hanks. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 11 (1999), 1–94.

Michael Brenner and Bernhard Nebel. 2009. Continual planning and acting in dynamic multiagent environments. *Autonomous Agents and Multi-Agent Systems* 19, 3 (2009), 297–331.

Kevin Carey, Dave Lewis, Steffen Higel, and Vincent Wade. 2004. Adaptive composite service plans for ubiquitous computing. In *International Workshop on Managing Ubiquitous Communications and Services*.

Berardina Carolis and Giovanni Cozzolongo. 2007. Planning the behaviour of a social robot acting as a majordomo in public environments. In *Congress of the Italian Association for Artificial Intelligence on AI*IA 2007: Artificial Intelligence and Human-Oriented Computing*. 805–812.

Mario Caruso, Çağri Ilban, Francesco Leotta, Massimo Mecella, and Stavros Vassos. 2013. Synthesizing daily life logs through gaming and simulation. In *ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*. 451–460.

Roberto Casati and Achille C. Varzi. 1999. *Parts and Places: The Structures of Spatial Representation*. MIT Press.

Luis A. Castillo, Juan Fernández-Olivares, Óscar García-Pérez, and Francisco Palao. 2006. Efficiently handling temporal knowledge in an HTN planner. In *International Conference on Automated Planning and Scheduling*. 63–72.

Ahmed-Chawki Chaouche, Amal El Fallah-Seghrouchni, Jean-Michel Ilié, and Djamel-Eddine Saïdouni. 2015. Improving the contextual selection of BDI plans by incorporating situated experiments. In *International Conference on Artificial Intelligence Applications and Innovations*. 266–281.

Chia-Hung Chen, Alan Liu, and Pei-Chuan Zhou. 2014. Controlling a service robot in a smart home with behavior planning and learning. In *International Conference on Systems, Man and Cybernetics*. 2821–2826.

Nanxi Chen, Nicolás Cardozo, and Siobhán Clarke. 2016. Goal-driven service composition in mobile and pervasive computing. *IEEE Transactions on Services Computing* PP, 99 (2016), 1–1.

Sehyeong Cho and Chulan Ren. 2008. Using goal-oriented paradigm for community computing. In *IEEE International Conference on Industrial Informatics*. 1031–1035.

Marcello Cirillo, Lars Karlsson, and Alessandro Saffiotti. 2008. A framework for human-aware robot planning. In *Scandinavian Conference on Artificial Intelligence (SCAI 2008)*. 52–59.

Marcello Cirillo, Lars Karlsson, and Alessandro Saffiotti. 2012. Human-aware planning for robots embedded in ambient ecologies. *Pervasive and Mobile Computing* 8, 4 (2012), 542–561.

Juliet Corbin and Anselm Strauss. 2008. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Sage.

John M. Corchado, Javier Bajo, and Ajith Abraham. 2008a. GerAmi: Improving healthcare delivery in geriatric residences. *IEEE Intelligent Systems* 23, 2 (2008), 19–25.

Juan M. Corchado, Javier Bajo, Juan F. De Paz, and Sara Rodrguez. 2009. An execution time neural-CBR guidance assistant. *Neurocomputing* 72, 1315 (2009), 2743–2753.

Juan M. Corchado, Javier Bajo, Yanira de Paz, and Dante I. Tapia. 2008b. Intelligent environment for monitoring alzheimer patients, agent technology for health care. *Decision. Support Systems* 44, 2 (2008), 382–396.

François Courtemanche, Mehdi Najjar, Blandine Paccoud, and André Mayers. 2008. Assisting elders via dynamic multi-tasks planning: A Markov decision processes based approach. In *International Conference on Ambient Media and Systems*. 1–8.

Oleg Davidyuk, Nikolaos Georgantas, Valérie Issarny, and Jukka Riekki. 2011. MEDUSA: Middleware for end-user composition of ubiquitous applications. In *Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspectives*, F. Mastrogiovanni and N. Y. Chong (Eds.). Vol. 11. IGI Global, 197–219.

Fred D. Davis. 1986. *A Technology Acceptance Model for Empirically Testing New End-user Information Systems : Theory and Results*. Ph.D. dissertation. Massachusetts Institute of Technology.

Giuseppe De Giacomo, Claudio Ciccio, Paolo Felli, Yuxiao Hu, and Massimo Mecella. 2012. *On the Move to Meaningful Internet Systems: OTM 2012: Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE*. Springer, 194–211.

Viktoriya Degeler, Luis I. Lopera Gonzalez, Mariano Leva, Paul Shrubsole, Silvia Bonomi, Oliver Amft, and Alexander Lazovik. 2013. Service-oriented architecture for smart environments. In *IEEE International Conference on Service Oriented Computing and Applications (SOCA'13)*. 99–104.

Maurizio Di Rocco, Federico Pecora, and Alessandro Saffiotti. 2013. When robots are late: Configuration planning for multiple robots with dynamic goals. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 5915–5922.

Maurizio Di Rocco, Subhash Sathyakeerthy, Jasmin Grosinger, Federico Pecora, Alessandro Saffiotti, Filippo Cavallo, Bonaccorsi Manuele, Raffaele Limosani, Alessandro Manzi, Giancarlo Teti, and Paolo Dario. 2014. A planner for ambient assisted living: From high-level reasoning to low-level robot execution and back. In *AAAI Spring Symposium*. 10–17.

Yun Ding, Christian Elting, and Ulrich Scholz. 2006. Seamless integration of output devices in intelligent environments: Infrastructure, strategies and implementation. In *IET International Conference on Intelligent Environments*. 21–30.

Stefan Edelkamp and Malte Helmert. 2001. MIPS: The model-checking integrated planning system. *AI Magazine* 22, 3 (2001), 67–72.

Christian Elting. 2005. Orchestrating output devices: Planning multimedia presentations for home entertainment with ambient intelligence. In *Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-aware Services: Usages and Technologies*. ACM, 153–158.

Manfred Eppe and Mehul Bhatt. 2015. Approximate postdictive reasoning with answer set programming. *Journal of Applied Logic* 13, 4, Part 3 (2015), 676–719.

Thomas Erl. 2007. *SOA Principles of Service Design*. Prentice Hall PTR.

Kutluhan Erol, James Hendler, and Dana S. Nau. 1994. HTN planning: Complexity and expressivity. In *National Conference on Artificial Intelligence - Volume 2*. AAAI, 1123–1128.

Fang Fang, Zhou Bo, Qian Kun, Ma Xudong, and Dai Xianzhong. 2015. Mobile robot task planning system design in intelligent environments. In *Chinese Control Conference*. 5815–5818.

Roy T. Fielding and Richard N. Taylor. 2002. Principled design of the modern Web architecture. *ACM Transactions on Internet Technology* 2, 2 (2002), 115–150.

Tarik Fissaa, Hatim Guermah, Hatim Hafiddi, Mahmoud Nassar, and Abdelaziz Kriouile. 2014. Towards an ontology based architecture for context-aware services composition. In *International Conference on Multimedia Computing and Systems*. 990–995.

Maria Fox and Derek Long. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20, 1 (2003), 61–124.

Juan A. Fraile, Yanira Paz, Javier Bajo, Juan Francisco Paz, and Belén Pérez-Lancho. 2013. Context-aware multiagent system: Planning home care tasks. *Knowledge and Information Systems* (2013), 1–33.

David Franklin and Kristian Hammond. 2001. The intelligent classroom: Providing competent assistance. In *International Conference on Autonomous Agents (AGENTS'01)*. ACM, 161–168.

Christian Fritz and Sheila A. McIlraith. 2007. Monitoring plan optimality during execution. In *International Conference on Automated Planning and Scheduling*. 144–151.

Alfredo Garro, Sergio Greco, and Fabio Palopoli. 2008. Smart agents and smart environments: A predictive approach to replanning. In *Intelligent Agents and Services for Smart Environments as Part of the Artificial Intelligence and Simulation of Behaviour Convention*. 7–12.

Ilche Georgievski and Marco Aiello. 2015. HTN planning: Overview, comparison, and beyond. *Artificial Intelligence* 222 (2015), 124–156.

Ilche Georgievski, Tuan Anh Nguyen, and Marco Aiello. 2013. Combining activity recognition and AI planning for energy-saving offices. In *International Conference on Ubiquitous Intelligence and Computing*. IEEE, 238–245.

A. Gerevini and D. Long. 2006. Preferences and soft constraints in PDDL3. In *ICAPS Workshop on Planning with Preferences and Soft Constraints*.

Alfonso Gerevini and Ivan Serina. 2002. LPG: A planner based on local search for planning graphs with action costs. In *International Conference on Artificial Intelligence Planning Systems*. 13–22.

Malik Ghallab, Dana S. Nau, and Paolo Traverso. 2004. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc.

Barney G. Glaser and Anselm L. Strauss. 2009. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine de Gruyter.

Nahid Golafshani. 2003. Understanding reliability and validity in qualitative research. *The Qualitative Report* 8, 4 (2003), 597–607.

Marek Grześ, Jesse Hoey, Shehroz S. Khan, Alex Mihailidis, Stephen Czarnuch, Dan Jackson, and Andrew Monk. 2014. Relational approach to knowledge engineering for POMDP-based assistance systems as a translation of a psychological model. *International Journal of Approximate Reasoning* 55, 1, Part 1 (2014), 36–58.

Young-Guk Ha, Joo-Chan Sohn, Young-Jo Cho, and Hyunsoo Yoon. 2005. Towards a ubiquitous robotic companion: Design and implementation of ubiquitous robotic service framework. *Electronics and Telecommunications Research Institute Journal* 27, 6 (2005), 666–676.

Kristian J. Hammond. 1989. *Case-based Planning: Viewing Planning as a Memory Task*. Academic Press Prof.

Anthony Harrington and Vinny Cahill. 2011. Model-driven engineering of planning and optimisation algorithms for pervasive computing environments. *Pervasive and Mobile Computing* 7, 6 (2011), 705–726.

Thomas Heider. 2003. Goal-oriented assistance for extended multimedia systems and dynamic technical infrastructures. In *IASTED International Conference on Internet and Multimedia Systems and Applications*.

Thomas Heider and Thomas Kirste. 2002. Supporting goal-based interaction with dynamic intelligent environments. In *European Conference on Artificial Intelligence (ECAI'02)*. 596–600.

Malte Helmert. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26, 1 (2006), 191–246.

Malte Helmert. 2009. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence* 173, 5–6 (2009), 503–535.

Eva Hidalgo, Luis Castillo, R. Ignacio Madrid, Óscar García-Pérez, Manuel R. Cabello, and Juan Fernández-Olivares. 2011. ATHENA: Smart process management for daily activity planning for cognitive impairment. In *Ambient Assisted Living*, José Bravo, Ramón Hervás, and Vladimir Villarreal (Eds.). Lecture Notes in Computer Science, Vol. 6693. Springer, 65–72.

Guy Hoffman and Cynthia Breazeal. 2007. Cost-based anticipatory action selection for human–robot fluency. *IEEE Transactions on Robotics* 23, 5 (2007), 952–961.

Jörg Hoffmann and Bernhard Nebel. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14, 1 (2001), 253–302.

Frank Honold, Pascal Bercher, Felix Richter, Florian Nothdurft, Thomas Geier, Roland Barth, Thilo Hörnle, Felix Schüssel, Stephan Reuter, Matthias Rau, Gregor Bertrand, Bastian Seegebarth, Peter Kurzok, Bernd Schattenberg, Wolfgang Minker, Michael Weber, and Susanne Biundo. 2014. Companion-technology: Towards user- and situation-adaptive functionality of technical systems. In *International Conference on Intelligent Environments*. 378–381.

Frank Honold, Felix Schüssel, Michael Weber, Florian Nothdurft, Gregor Bertrand, and Wolfgang Minker. 2013. Context models for adaptive dialogs and multimodal interaction. In *Proceedings of the 2013 9th International Conference on Intelligent Environments (IE)*. 57–64.

Bill Irwin. 2014. *Interstellar*. Directed by Christopher Nolan. Legendary Pictures, Syncopy, Lynda Obst Productions, UK and USA. Film.

Emilie M. D. Jean-Baptiste, Pia Rotshtein, and Martin J. Russell. 2015. POMDP based action planning and human error detection. In *International Conference on Artificial Intelligence Applications and Innovations*. 250–265.

Wan-rong Jih, Li-lu Chen, and Jane Yung-jen Hsu. 2007a. A context-aware service platform in a smart space. In *ACM International Workshop on Agent-Based Ubiquitous Computing*.

Wan-rong Jih, Jane Yung-jen Hsu, Tsu-Chang Lee, and Li-lu Chen. 2007b. A multi-agent context-aware service platform in a smart space. *Journal of Computers* 18, 1 (2007), 45–60.

Eirini Kaldeli, Alexander Lazovik, and Marco Aiello. 2016. Domain-independent planning for services in uncertain and dynamic environments. *Artificial Intelligence* 236, 7 (2016), 30–64.

Eirini Kaldeli, Ehsan U. Warriach, Jaap Bresser, Alexander Lazovik, and Marco Aiello. 2010. Interoperation, composition and simulation of services at home. In *International Conference on Service-Oriented Computing (ICSOC'10)*. 167–181.

Eirini Kaldeli, Ehsan U. Warriach, Alexander Lazovik, and Marco Aiello. 2012. Coordinating the Web of services for a smart home. *ACM Transactions on the Web* 7, 2, Article 10 (2012).

Dimitris N. Kalofonos and Paul Wisner. 2007. A framework for end-user programming of smart homes using mobile devices. In *IEEE Consumer Communications and Networking Conference*. 716–721.

Henry Kautz and Bart Selman. 1999. Unifying SAT-based and graph-based planning. In *International Joint Conference on Artifical Intelligence - Volume 1*. 318–325.

Sang Hwan Kim, Sung Woo Kim, and HyunMi Park. 2003. Usability challenges in ubicomp environment. In *International Ergonomics Association*.

Barbara Kitchenham and Stuart Charters. 2007. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. Technical Report EBSE 2007-001. Keele University and Durham University Joint Report.

Terry P. Klassen, Alejandro R. Jadad, and David Moher. 1998. Guides for reading and interpreting systematic reviews: I. Getting started. *Archives of Pediatrics & Adolescent Medicine* 152, 7 (1998), 700–704.

Uwe Köckemann, Federico Pecora, and Lars Karlsson. 2014. Grandpa hates robots—Interaction constraints for planning in inhabited environments. In *AAAI Conference on Artificial Intelligence*. 2293–2299.

Roman Kontchakov, Ian Pratt-Hartmann, and Michael Zakharyaschev. 2014. Spatial reasoning with RCC8 and connectedness constraints in Euclidean spaces. *Artificial Intelligence* 217 (2014), 43–75.

Evangelos Kotsovinos and Maja Vukovic. 2005. Su-chef: Adaptive coordination of intelligent home environments. In *Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services*. IEEE, 74–74.

Frank Krüger, Gernot Ruscher, Sebastian Bader, and Thomas Kirste. 2011. A context-aware proactive controller for smart environments. *I-COM* 10 (2011), 41–48.

Oliver Lemon and Ian Pratt. 1997. Spatial logic and the complexity of diagrammatic reasoning. *Machine Graphics and Vision* 6, 1 (1997), 89–108.

Huan-Ming Liang, Alan Liu, Yi-Chih Chen, and Chiung-Hon Leon Lee. 2010. Device collaboration in smarthomes as service delivery. In *SICE Annual Conference*. 30–34.

Yvonna S. Lincoln and Egon G. Guba. 1985. *Naturalistic Inquiry*. Sage Publications Inc.

Mohcine Madkour, Driss El Ghanami, and Abdelilah Maach. 2013. Context-aware service adaptation: An approach based on fuzzy sets and service composition. *Journal of Information Science and Engineering* 29, 1 (2013), 1–16.

Florian Marquardt, Christiane Reisse, Adelinde Uhrmacher, and Thomas Kirste. 2008. A two-way approach to service composition in smart device ensembles. In *Advanced Topics in Telecommunication*. 49–60.

Florian Marquardt and Adelinde Uhrmacher. 2009a. Creating AI planning domains for smart environments using PDDL. In *Intelligent Interactive Assistance and Mobile Multimedia Computing*, Djamshid Tavangarian, Thomas Kirste, Dirk Timmermann, Ulrike Lucke, and Daniel Versick (Eds.). Communications in Computer and Information Science, Vol. 53. Springer, 263–274.

Florian Marquardt and Adelinde M. Uhrmacher. 2009b. An AI-planning based service composition architecture for ambient intelligence. In *Intelligent Environments (Workshops) (Ambient Intelligence and Smart Environments)*, Vol. 4. 145–152.

David Martin, Mark Burstein, Drew Mcdermott, Sheila Mcilraith, Massimo Paolucci, Katia Sycara, Deborah L. Mcguinness, Evren Sirin, and Naveen Srinivasan. 2007. Bringing semantics to web services with OWL-S. *World Wide Web* 10, 3 (2007), 243–277.

Ricardo De Masellis, Claudio Di Ciccio, Massimo Mecella, and Fabio Patrizi. 2010. Smart home planning programs. In *International Conference on Service Systems and Service Management (ICSSSM)*. 1–6.

Fulvio Mastrogiovanni, Antonello Scalmato, Antonio Sgorbissa, and Renato Zaccaria. 2010. Affordance-based planning for assisting humans in daily activities. In *International Conference on Intelligent Environments*. 19–24.

Thomas Leo McCluskey. 2002. Knowledge engineering: Issues for the AI planning community. In *The AIPS-2002 Workshop on Knowledge Engineering Tools and Techniques for AI Planning*.

Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. 1998. *PDDL—The Planning Domain Definition Language*. Technical Report. CVC TR-98-003/DCS TR-1165. Yale Center for Computational Vision and Control.

Alfredo Milani and Valentina Poggioni. 2007. Planning in reactive environments. *Computational Intelligence* 23, 4 (2007), 439–463.

Christian Muise, J. Christopher Beck, and Sheila A. McIlraith. 2013. Flexible execution of partial order plans with temporal constraints. In *International Joint Conference on Artificial Intelligence*. 2328–2335.

Dana S. Nau, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu, and Fusun Yaman. 2003. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research* 20, 1 (2003), 379–404.

Qun Ni. 2005. Service composition in ontology enabled service oriented architecture for pervasive computing. In *Workshop on Ubiquitous Computing and e-Research*.

Qun Ni and Morris Sloman. 2005. An ontology-enabled service oriented architecture for pervasive computing. In *International Conference on Information Technology: Coding and Computing*, Vol. 2. IEEE, 797–798.

Javier Ortiz, Angel García-Olaya, and Daniel Borrajo. 2013. Using activity recognition for building planning action models. *International Journal of Distributed Sensor Networks* 9, 6 (2013).

Madhukar Pai, Michael McCulloch, Jennifer D. Gorman, Nitika Pai, Wayne Enanoria, Gail Kennedy, Prathap Tharyan, and John M. Colford. 2004. Systematic reviews and meta-analyses: An illustrated, step-by-step guide. *National Medical Journal of India* 17, 2 (2004), 89–95.

Sergio Pajares Ferrando and Eva Onaindia. 2013. Context-aware multi-agent planning in intelligent environments. *Information Sciences* 227 (2013), 22–42.

Justin Mazzola Paluska, Hubert Pham, Umar Saif, Grace Chau, Chris Terman, and Steve Ward. 2008. Structured decomposition of adaptive applications. *Pervasive and Mobile Computing* 4, 6 (2008), 791–806.

Federico Pecora, Marcello Cirillo, and Michael Brenner. 2010. A constraint-based approach for plan management in intelligent environments. In *Cognitive Robotics*.

Federico Pecora, Marcello Cirillo, Francesca Dell'Osa, Jonas Ullberg, and Alessandro Saffiotti. 2012. A constraint-based approach for proactive, context-aware human support. *Journal of Ambient Intelligence and Smart Environments* 4, 4 (2012), 347–367.

Mark Petticrew and Helen Roberts. 2006. *Systematic Reviews in the Social Sciences: A Practical Guide*. Blackwell Publishing.

Joaquin Phoenix and Scarlett Johansson. 2013. *Her*. Directed by Jonze Spike. Annapurna Pictures, LA. Film.

Christiane Plociennik, Christoph Burghardt, Florian Marquardt, Thomas Kirste, and Adelinde Uhrmacher. 2009. Modelling device actions in smart environments. In *Intelligent Interactive Assistance and Mobile Multimedia Computing*, Djamshid Tavangarian, Thomas Kirste, Dirk Timmermann, Ulrike Lucke, and Daniel Versick (Eds.). Communications in Computer and Information Science, Vol. 53. Springer. 213–224.

Abir Qasem, Jeff Heflin, and Héctor Muñoz-avila. 2004. Efficient source discovery and service composition for ubiquitous computing environments. In *Workshop on Semantic Web Technology for Mobile and Ubiquitous Applications*.

Lirong Qiu, Zhongzhi Shi, and Fen Lin. 2006. Context optimization of AI planning for services composition. In *International Conference on e-Business Engineering (ICEBE'06)*. 610–617.

Anand Ranganathan and Roy H. Campbell. 2004. Autonomic pervasive computing based on planning. In *International Conference on Autonomic Computing*. 80–87.

Earl David Sacerdoti. 1975. *A Structure for Plans and Behavior*. Ph.D. dissertation. Standford University, AI Center.

Johnny Saldana. 2009. *The Coding Manual for Qualitative Researchers*. Sage Publications Ltd.

Inmaculada Sánchez-Garzón, Gonzalo Milla-Millán, and Juan Fernández-Olivares. 2012. Context-aware generation and adaptive execution of daily living care pathways. In *International Conference on Ambient Assisted Living and Home Care*. Springer, 362–370.

Makoto Sando and Reiko Hishiyama. 2011. Human-Centered planning for adaptive user situation in ambient intelligence environment. In *International Conference on Agents in Principle, Agents in Practice*. 520–531.

María J. Santofimia, Scott E. Fahlman, Xavier del Toro, Francisco Moya, and Juan C. López. 2011. A semantic model for actions and events in ambient intelligence. *Engineering Applications of Artificial Intelligence* 24, 8 (2011), 1432–1445.

María J. Santofimia, Scott E. Fahlman, Francisco Moya, and Juan C. López. 2010. A common-sense planning strategy for ambient intelligence. In *International Conference on Knowledge-based and Intelligent Information and Engineering Systems: Part II*. 193–202.

Mohammad Munshi Shahin Shah, Lukás Chrpa, Falilat Jimoh, Diane E. Kitchin, Thomas Leo McCluskey, Simon Parkinson, and Mauro Vallati. 2013. Knowledge engineering tools in planning: State-of-the-art and future challenges. In *Workshop on Knowledge Engineering for Planning and Scheduling at ICAPS*. 53–60.

Mithun Sheshagiri, Norman M. Sadeh, and Fabien Gandon. 2004. Using semantic Web services for context-aware mobile. In *Workshop on Context Awareness Applications*.

Emrah Akin Sisbot, Luis F. Marin-Urias, Rachid Alami, and Thierry Siméon. 2007. A human aware mobile robot motion planner. *IEEE Transactions on Robotics* 23, 5 (2007), 874–883.

Carol Smidts, Chetan Mutha, Manuel Rodríguez, and Matthew J. Gerber. 2014. Software testing with an operational profile: OP definition. *ACM Computing Surveys* 46, 3, Article 39 (2014), 39:1–39:39 pages.

David E. Smith, Jeremy Frank, and Ari K. Jónsson. 2000. Bridging the gap between planning and scheduling. *Knowledge Engineering Review* 15, 1 (2000), 47–83.

Seheon Song and Minkoo Kim. 2011. A plan-based service composition for work process agent in ubiquitous computing. In *Asia-Pacific Services Computing Conference (APSCC)*. IEEE, 483–487.

Seheon Song and Seok-Won Lee. 2013. A goal-driven approach for adaptive service composition using planning. *Mathematical and Computer Modelling* 58, 1–2 (2013), 261–273.

Thanos G. Stavropoulos, Ageliki Tsioliaridou, George Koutitas, Dimitris Vrakas, and Ioannis Vlahavas. 2010. *International Conference on Artificial Neural Networks*. Springer, 477–482.

Thanos G. Stavropoulos, Dimitris Vrakas, and Ioannis Vlahavas. 2011. A survey of service composition in ambient intelligence environments. *Artificial Intelligence Review* (2011), 1–24.

Roykrong Sukkerd, David Garlan, and Reid Simmons. 2015. Task planning of cyber-human systems. In *International Conference on Software Engineering and Formal Methods*. 293–309.

Alistair Sutcliffe. 2003. Scenario-based requirements engineering. In *International Requirements Engineering Conference*. 320–329.

Kartik Talamadupula, J. Benton, Subbarao Kambhampati, Paul Schermerhorn, and Matthias Scheutz. 2010. Planning for human-robot teaming in open worlds. *ACM Transactions on Intelligent Systems and Technology* 1, 2 (2010), 14:1–14:24.

Celia Taylor, Graham R. Gibbs, and Ann Lewins. 2014. Quality of qualitative analysis. (Online Oct. 2014). http://onlineqda.hud.ac.uk/Intro_QDA/qualitative_analysis.php.

Bernhard Thalheim. 2010. Towards a theory of conceptual modelling. *Journal of Universal Computer Science* 16, 20 (2010), 3102–3137.

Aitor Urbieta, Guillermo Barrutieta, Jorge Parra, and Aitor Uribarren. 2008. A survey of dynamic service composition approaches for ambient systems. In *Ambi-Sys Workshop on Software Organisation and MonIToring of Ambient Systems*. 1:1–1:8.

Mathieu Vallée, Fano Ramparany, and Laurent Vercouter. 2005. Flexible composition of smart device services. In *International Conference on Pervasive Systems and Computing (PSC'05)*. 165–171.

Tiago Vaquero, Sharaf Mohamed, Goldie Nejat, and J. Christopher Beck. 2015. The implementation of a planning and scheduling architecture for multiple robots assisting multiple users in a retirement home setting. 47–52.

Tiago Stegun Vaquero, José Reinaldo Silva, Flavio Tonidandel, and J. Christopher Beck. 2013. itSIMPLE: Towards an integrated design system for real planning applications. *Knowledge Engineering Review* 28, 2 (2013), 215–230.

Marc Vilain and Henry Kautz. 1986. Constraint propagation algorithms for temporal reasoning. In *AAAI Conference on Artificial Intelligence*. 377–382.

Maja Vukovic, Evangelos Kotsovinos, and Peter Robinson. 2007. An architecture for rapid, on-demand service composition. *Service Oriented Computing and Applications* 1, 4 (2007), 197–212.

Maja Vukovic and Peter Robinson. 2004. Adaptive, planning based, web service composition for context awareness. In *International Conference on Pervasive Computing: Advances in Pervasive Computing*, Vol. 176. 257–252.

Wenshan Wang, Qixin Cao, XiaoXiao Zhu, and Shuang Liang. 2015. A framework for intelligent service environments based on middleware and general purpose task planner. In *International Conference on Intelligent Environments*. 184–187.

Mathijs de Weerdt and Brad Clement. 2009. Introduction to planning in multiagent systems. *Multiagent and Grid Systems* 5, 4 (2009), 345–355.

Mark Weiser. 1999. The computer for the 21st century. *SIGMOBILE Mobile Computing and Communications Review* 3, 3 (1999), 3–11.

Alfred North Whitehead. 2010. *Process and Reality*. Simon and Schuster. (1st ed., 1929).

Anna M. Wichansky. 2000. Usability testing in 2000 and beyond. *Ergonomics* 43, 7 (2000), 998–1006.

Paul Wisner. 2006. Automatic composition in service browsing environments. In *Workshop on Mobile Interaction with the Real World*. 39–42.

Michael Wooldridge. 2009. *An Introduction to Multi-agent Systems*. Wiley Publishing.

Stephen S. Yau and Arun Balaji Buduru. 2014. Intelligent planning for developing mobile IoT applications using cloud systems. In *IEEE International Conference on Mobile Services*. 55–62.

Kristina Yordanova. 2011. Modelling human behaviour using partial order planning based on atomic action templates. In *International Conference on Intelligent Environments*. 338–341.

Received August 2015; revised September 2016; accepted October 2016