Situvis: A sensor data analysis and abstraction tool for pervasive computing systems

Adrian K. Clear [a,*], Thomas Holland [b], Simon Dobson [c], Aaron Quigley [c], Ross Shannon [b], Paddy Nixon [b]

[a] *Orange Labs R&D, Meylan, France*
[b] *Systems Research Group, UCD Dublin, Ireland*
[c] *School of Computer Science, University of St Andrews, UK*

ARTICLE INFO

ABSTRACT

Pervasive systems are large-scale systems consisting of many sensors capturing numerous types of information. As this data is highly voluminous and dimensional, data analysis tasks can be extremely cumbersome and time-consuming. Enabling computers to recognise real-world situations is an even more difficult problem, involving not only data analysis, but also consistency checking. Here we present Situvis, an interactive visualisation tool for representing sensor data and creating higher-level abstractions from the data. This paper builds on previous work, Clear et al. (2009) [8] through evolved tool functionality and an evaluation of Situvis. A user-trial consisting of 10 participants shows that Situvis can be used to complete the key tasks in the development process of situation specifications in over 50% less time than an improvised alternative toolset.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Context-aware pervasive systems are designed to support a user's goals by making adaptations to their behaviours in response to the user's activities or circumstances. An example of such a system is that of Miele et al., where context can be associated with user preferences for information retrieval, meaning that those preferences are only applied in the specified contexts [1]. The accuracy and utility of these adaptations is predicated on the system's ability to capture and recognise these circumstances as they occur. To achieve this, a system designer must characterise these adaptation opportunities by collecting context data from multiple heterogeneous sensors, which may be networked physical instruments in the environment (measuring factors like temperature, noise volume or humidity), software sensors retrieving information from the web or various data feeds, or wearable sensors measuring factors such as acceleration or object use. These context data are voluminous, highly multivariate, and constantly being updated as new readings are recorded.

To better manage this complexity, we can use data abstraction to shield developers from having to deal with raw sensor data—data that often requires a steep learning curve for interpretation, such as accelerometer data or 3D-coordinate location data. One active research area in this direction involves using machine learning techniques to perform activity recognition—activities being higher-level interpretations of raw sensor data, representing objective actions such as cooking and walking. *Situations* have been proposed as another high-level abstraction of context data [2]. They symbolically define commonly-experienced occurrences such as a user "taking a coffee break", or being "in a research meeting", without requiring the user to understand any of the dozens of distinct sensor readings which may have gone into making up these situations.

---

* Corresponding author.
E-mail address: adrian.clear@orange-ftgroup.com (A.K. Clear).

Situations are thus a natural view of a context-aware system, whereas the individual pieces of context are each "a measurable component of a given situation" [3]. From a software engineering perspective, we define situations in terms of high-level context: data that has been encapsulated to a level of understanding appropriate for a developer specifying a situation (e.g., symbolic locations), as opposed to a physics expert (e.g., 3D-coordinates), for example.

Thomson et al. observe that there are two approaches to situation determination: manual specification and machine learning-based approaches [4]. The manual specification approach suffers from complexity. As the context information available to a context-aware system at any moment is so extensive, dynamic and highly dimensional, it is a significant challenge for a system observer to ascribe significance to changes in the data or identify emergent trends, much less capture the transient situations that are occurring amid the churn of data.

Machine learning-based approaches are insufficient due to the extensive training data required. Many situations are subjective and hence require a degree of personalisation. We believe that it is unrealistic to assume that every user of a context-aware system will go through the long and tedious training process required for supervised learning techniques. Here, we propose a hybrid approach that utilises minimal training data to frame a situation specification, combined with relevant visualisations that simplify the manual process of fine-tuning.

Existing work has applied the coupling of data- and user-driven processes to carry out difficult tasks. In particular, the general Interactive Machine Learning (IML) model consists of iterations of classical machine learning followed by refinement through interactive use—in the Crayons project [5], users can build classifiers for image-based perceptual user interfaces using a novel IML model that involves iterative user interaction in order to minimise the feature set and build a decision tree. Moreover, Dey's *a CAPpella* is a prototyping environment, aimed at end-users, for context-aware applications [6]. It uses a *programming by demonstration* approach, through a combination of machine-learning and user interaction, to allow end-users to build complex context-aware applications without having to write any code.

The visualisation of large and complex multivariate data sets, such as those that context-aware system developers work with, is becoming increasingly crucial in helping those developers to organise and distill data into usable information [7]. Interactive visualisation tools help the viewer perform visual data analysis tasks: exploring patterns and highlighting and defining filters over interesting data.

Situvis is our scalable interactive visualisation tool for pervasive systems [8]. By illustrating sensor data using effective and intuitive visualisations, combined with simple, intuitive interactive functionalities, Situvis affords users the ability to quickly identify interesting features of the data. By incorporating real situation traces and annotations as ground truth, Situvis assists system developers in constructing and evaluating accurate situation specifications by essentially bootstrapping the manual process, hence affording them a better understanding of the situation space, and the reliability of modeling with situations based on real, recorded sensor data. It is a framework that allows developers to understand, at a high level, how their system will behave given certain inputs.

The following section provides some background for our work. In Section 3, we describe the details of the Situvis tool. Section 4 features an evaluation of Situvis through a user-study in which it is compared to an improvised alternative toolset. Finally, we conclude in Section 5 and present some potential future work.

2. Background

2.1. Abstract models and activity recognition

Abstract models are often constructed in order to generalise concrete concepts by capturing their common properties and structure. Besides abstracting situations from context data, other abstract modelling research occurs in the literature. Penta et al. [9] introduce ontologies for representing and reasoning about the high-level content of multimedia data, particularly images. High-level knowledge, such as the fact that an image illustrates a jockey riding a racehorse, can be inferred from lower level concepts such as colour, texture and shapes, and spatial relationships between these. We infer situations from unrelated dimensions. However, future work is planned to investigate the potential of modelling situations as temporally related sub-events. Ghezzi et al. [10] describe an approach to creating behaviour models of software components given a black-box view of them. A finite state machine is first created which models partial behaviour of the component. This graph is then generalised through graph transformation rules which describe the abstract behaviour of the component. In comparison to this work, we are more concerned with abstracting states (situations) rather than state transitions (situation changes).

Techniques for activity recognition use machine learning techniques – both supervised and unsupervised – to infer high-level activities from low-level sensor data. Logan et al. present a long-term experiment of activity recognition in a home setting using a semi-supervised technique [11]. Like the majority of activity recognition, the focus is on concepts that can be described and recognised by body movements and object use. 104 h of video data was manually annotated following a period of data collection. Many activities, such as dishwashing and meal preparation, were accurately classified to a high degree. However, the study showed that even with this large amount of data and annotation, some activities, such as drying dishes, could not be learned effectively due to lack of training data caused by their infrequent occurrence, even over a 104-hour period.

Krause et al. describe an approach to learning context-dependent personal preferences using machine learning techniques to refine the behaviour of Sensay, a context-aware mobile phone [12]. The behaviour modifications, such as

changing the state of the ringer volume from loud to silent, are known in advance. The task is to find the user's "state" (or contextual circumstances) that corresponds to them modifying the behaviour of their phone so that in the future it can be done automatically. Machine learning of personalised states is favoured over manual specification of general states as a result of a study showing that states and desired phone behaviour differed among individuals. Because the behaviour modifications are known, this method requires no supervision. Essentially, the behaviour modifications serve as labels for the recorded sensor values.

2.1.1. Recognising higher-level abstractions

Recent work has aimed to recognise high-level abstractions of context called routines [13]. Routines are structured as compositions of several activities that may be influenced by time, location and the individual performing them. Examples include "commuting" or "working". In contrast to activities, routines cannot be identified through their local physical structure alone: they consist of variable patterns of multiple activities; they range over longer periods of time; and they often vary significantly between instances. Moreover, they are subjective. As a result, the authors chose topic maps as an alternative approach for recognition. Topic maps are a family of probabilistic models often used by the text processing community and enable the recognition of daily routines as a composition of activity patterns.

We too aim to be able to recognise high-level abstractions, and our approach is designed to achieve this with minimal annotation. Situations and routines are similar in that they are subjective, making long periods of annotation unscalable; and they require more factors to recognise them than simply body posture, body movement, or object use. Therefore, accurate situation determination cannot rely completely on data-driven techniques. Situations are generally short term and hence are logically more complex than routines—they can be partially described in terms of individual activities but they are not lengthy nor activity-rich enough to be represented as the most probable activities that are occurring over a long time window. As a result, we are taking a hybrid approach to recognition that includes a short ground truth collection period followed by manual fine-tuning by a domain expert.

2.2. Situation specifications

We have reviewed related literature on abstract models to put situation modelling into context. Here, we will explore the structure of situations, as it is an important factor in how they are represented in Situvis. Based on the extensive literature on the subject of modeling context for adaptive systems [2,3,14–17], we can make some observations: the incoming sources of context into a pervasive application are viewed as a finite number of variables: either nominal or categorical values, e.g., activity levels {idle, active, highly active … }; or quantitative ordinal values which may be defined over some known interval, e.g., noise level in decibels {0, 140}.

Location information will typically arrive as individual values for an object's x, y and z coordinates in a space, and may be recorded by numerous disparate positioning systems, but is modeled as a higher-level abstraction to make it easier to reason with. Previously conducted research allows component x, y and z coordinates to be composed into a symbolic representation, given some domain information [18], and so we can work with locations as readable as "Simon's office" or "Coffee Area". Our visualisation tool works equally well with simple quantitative data or these higher-order categorised data.

Numerous work in the literature focuses on programming abstractions for representing structures like situations. Bacon et al. [19] introduce Composite Events which are similar to Context Widgets of the Context Toolkit [20]. They are components that subscribe to low-level events and fire higher-level events themselves when a set of constraints, expressed in first-order logic, are met. Abstract Events [21,22] extend the notion of Composite Events by adding further semantics such as Temporal First-Order Logic reasoning. Bolchini et al. [23] describe a similar, but more application-specific, model called the Context Dimension Tree that is used to create context configurations for tailoring an information space for differing user information needs. Relevant contexts can be specified by traversing the tree resulting in a set of logically conjoined assertions of differing granularities. Further theory on the semantics of situation specification can be seen in the work of Henricksen [2] and Loke [24]. Based on this work, we also model situations using declarative languages, which can simply be plugged in to our tool.

Situation specifications are boolean expressions (or assertions)—they are either true or false, denoting occurrence and non-occurrence, respectively. Assertions may be composed using the logical operators AND ($\wedge$), OR ($\vee$), and NOT ($\neg$), resulting in richer expressions. Domain-specific functions can also be defined to enrich specification semantics (e.g., a distance operator could return a numerical value of the distance between two locations). We can thus define a situation specification as a concatenation of one or more assertions about contexts, which leads us to the following formal definition:

A situation specification consists of one or more assertions about context that are conjoined using the logical operators AND ($\wedge$), OR ($\vee$), *and* NOT ($\neg$). *Assertions may comprise further domain-specific expressions on context, given that the required semantics are available.*

2.3. Visualisation of context data

The field of visual analytics uses interactive visual interfaces to aid end-users in analysing and understanding large and complex multivariate data sets. Interactive visualisation tools help the viewer perform visual data analysis tasks: exploring

patterns and highlighting and defining filters over interesting data. For example, Andrienko et al. developed a toolset for analysing and reasoning about movement data (e.g., GPS coordinates). Following some preprocessing steps, the data can be clustered according to different properties, such as start and end points of trips, or similar behaviour over time [25]. Such properties can be portrayed using different types of visualisations to increase user understanding.

There exist myriad visualisation techniques, from time-series to multi-dimensional scatter plot methods, which can be adapted to the exploration of multidimensional context data. Our focus here is not only on the exploration of such context data, but also the scope of the higher order situations, their specification, and data cases which fall outside the set boundaries. The Table Lens, a focus + context visualisation, supports the interactive exploration of many data values in a semi-familiar spreadsheet format [26]. In practice, due to the distortion techniques employed, users can see 100 times as many data items within the same screen space as compared with a standard spreadsheet layout. Rather than showing the detailed numeric values in each cell, a single row of pixels, relating to the value in the cell, is shown instead. The Table Lens affords users the ability to easily study quantitative data sets, but categorical values are not well supported.

2.4. The context-aware application development process

Creating a context-aware application is a difficult process due to the lack of tool support and programming methodologies for handling context data and evaluating the completeness and correctness of situation specifications. Here, we wish to outline the key tasks of the development cycle, and they will be referenced later in the design of our evaluation.

Situation specification is concerned with providing a link from the context data to the application behaviour. It involves specifying the constraints that characterise a situation. This is a difficult task when there are many dimensions of context available and a user must conceptually aggregate and constrain these in order to capture real-world situations such as `in a meeting`.

Once a situation has been specified, its correctness must be evaluated. Evaluation encapsulates two key tasks. The first, *evaluating a specification in relation to the data*, is necessary in order to make the situation specifications consistent with annotated sensor data. Annotations provide ground truth that can be used as a guide for a developer when the task of specifying a situation from scratch is too challenging.

In order to *evaluate a specification in relation to other specifications*, a developer must be able to test the consistency of their situation specifications in relation to the other situation specifications in the system. Some situations should naturally never co-occur, for example, and this relationship must be reflected in the set of situations that applications adapt to. This is a difficult task when there are a large number of specifications that must be consistent with one another, and when the number of dimensions that they are constrained over is high.

Analysis tasks are important to completing both *situation specification* and *evaluating a specification in relation to the data* tasks. Tasks in this category cover cases where the user must interact with the data in order to find the answer to a particular question. Traditionally, this is done by making database queries or sorting tables of data according to some attribute. Other than for the above tasks, a developer may wish to carry out analysis task for fault detection, sensor coverage, or statistical analysis.

2.5. Data collection

Situvis is largely a data-oriented tool and, as a result, cannot be demonstrated or evaluated adequately without a dataset. For this reason, we built an infrastructure to capture the following data about a person in our research lab environment: their computer activity, their calendar entries, their instant messenger (IM) status, the number of their colleagues in their vicinity, their physical activity, the noise level in their environment, the selected profile on their mobile phone, and their location. All sensors were synchronised and some data was abstracted from low-level sensors as described below.

The `Computer activity` sensor runs on the participant's desktop PC and monitors the rate of key presses and mouse clicks, along with the length of time since the last activity. The data from this sensor is abstracted to the values of `Idle 1+ hours`, `Idle`, `Active` and `High activity`. The `Calendar status` sensor *scrapes* Google Calendar and collects information about events such as a title, start and end time, and participants. The `IM status` sensor obtains its readings from Gtalk through the Google Talk API.

The [number of] `Colleagues present`, `Physical activity`, `Noise level` and `Phone profile` sensor data were all obtained from the participant's mobile phone. We created a PyS60[1] *sensing platform* for the Nokia N95 that recorded the results of a Bluetooth scan and the phone profile information every minute; recorded data from the 3-axis accelerometer; and finally recorded microphone data. We asked our research lab colleagues to register their Bluetooth MAC addresses with us so that we could extract the number of colleagues present from the Bluetooth scan data. We extracted an 8-point scale of noise level from the microphone data and we extracted the physical activities of `Walking`, `Running` and `Idle` from the accelerometer data.

For `Location`, we have a Ubisense[2] deployment on the third and fourth floor of our research lab. We found that although Ubisense could give us precise results, our deployment was unreliable and sometimes gave no readings at all. Therefore, we supported this infrastructure with Bluetooth technology—we deployed two Bluetooth beacons, one on the third floor of our research lab and one on the fourth floor, and used the mobile phone Bluetooth scan information to search for the MAC addresses of these beacons. We could then reinforce our location data as a result.

We recruited a participant to gather a dataset over a 4-day period. We decided on a period of 4 days because we felt that this was a sufficient amount of time to capture a person's regular working-day routine (i.e., some instances of them going for lunch, having meetings, working at their desk, etc.). The goal was to capture a range of situations, from those that occur infrequently (once per week), to those that occur one or more times per day. Data collection involved sharing a calendar with us where the participant would put events titled 'Busy' into the calendar corresponding to events where she intended to be busy (e.g., if she intended to attend a meeting).[3] The participant was asked to use the calendar in the same manner that she would use her personal calendar to record actual events. In order to share her `IM status` with us, the participant befriended an account that we set up for the data collection period, allowing us to retrieve her status' through the Google Talk APIs. We asked the participant to turn the sensors on at the beginning of her work day and to turn them off at the end.

We were only concerned with data from the participant's working days for this study. There are two main reasons for this: Firstly, the N95 only lasts approximately 7 h (excluding talktime) when the sensors are running so we felt it would be too much of a disruption to the participant to have to charge her phone so frequently for 96 h. Secondly, our location infrastructure exists in our research lab alone, and we felt that this was an important context dimension to capture as often as possible. We could have captured GPS data in the outdoor environment but we felt that it would be too much of a drain on the battery, especially over a 96-hour period.

The participant was also asked to annotate the situations that she encountered throughout her days using a pen & paper annotation technique—she carried a notebook with her throughout the day and wrote down the start and end times of different situations as accurately as she could.

3. Situvis

The first version of Situvis [8] was developed in Processing [27], a Java-based visualisation framework that supports rapid prototyping of visualisation techniques. The current version was re-developed in Java to make it more extensible. Situvis is open-source software and can be downloaded from `situvis.com`.[4]

Situvis uses two visualisation techniques, a Parallel Coordinates visualisation and a time-series visualisation, to support the development of situation specifications. The PCV facilitates the display of a large amount of data traces, along with annotations where available, in a single view; and allows the user to clearly and easily explore the data for patterns. A user can program a set of constraints for a situation in the same view as the data, and adjust the resulting set of ranged intervals over significant parts of the data to create a more complete and accurate situation specification. The time-series visualisation allows the user to efficiently select parts of a dataset for in-depth analysis by observing correlations between classifications and annotations of the data traces.

3.1. Situvis views

The Situvis tool contains two separate interfaces named according to the principal visualisations used in each: the *Time-series (TS) view* and the *Parallel Coordinates (PC) view*. A user can move between these views to carry out different tasks from the situation specification development cycle, as we will see in the following sections.

The *TS view* consists of a time-series representation of the data and a panel for selection of annotated traces and classified traces. A screenshot of this interface can be seen in Fig. 1. The time-series visualisation of the data appears on the left side of the figure. Each data trace is timestamped, and the traces are temporally ordered from top to bottom. Two parameters of the visualisation can be manipulated as required using the buttons at the bottom of the time-series visualisation: the thickness of the lines representing a data trace, and the space between the lines. A user can use the lowest parameter values to see as much data as possible on the screen, or increase the parameters in order to distinguish clearly between individual traces.

The lines representing the data traces are divided into two segments. The left side of the line represents the annotation for the trace in question, and the right side represents its classification. Each segment is coloured according to the values that these attributes have. The right side of the interface contains a panel of labels for both annotations and situation specifications. There is a box beside each one that illustrates the colour that it will be represented as in the time-series visualisation. If a trace has more than one annotation or classification, the portion of the line representing that attribute is divided evenly, and the resulting segments are coloured appropriately.

---

[2] The Ubisense ultra-wideband location system: http://www.ubisense.net/.

[3] The reason that we did not simply "scrape" the participant's personal calendar was because we wanted to be able to distinguish between actual events and events that serve as reminders, such as "Bruce's birthday".

[4] Situvis is freely-available software, which you are encouraged to download from our website at http://situvis.com.
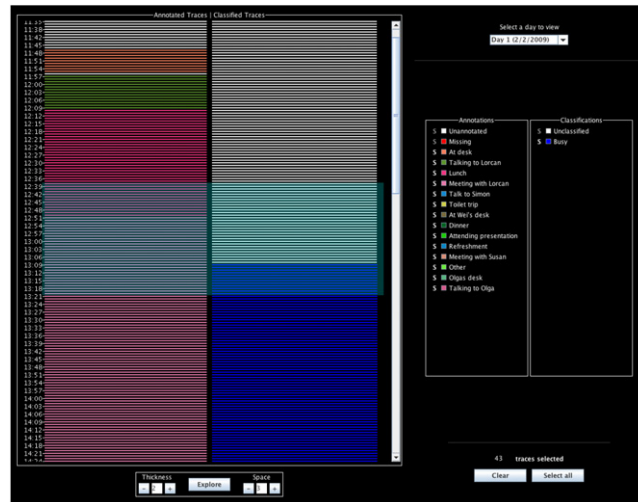
Fig. 1. The time-series view interface in Situvis. The time-series visualisation appears on the left side and the annotations and specifications on the right side. The traces are coloured according the their annotation and the situation that they are classified as. In the diagram, a Busy situation has been specified as Calendar status:Busy and Computer activity:Idle. The user has brushed a selection of traces (the semi-opaque cyan rectangle). By clicking the Explore button, this selection would be drawn in the PC view.

A user can interact with the traces in the time-series view by selecting a subset of them, resulting in them being highlighted. They can be selected in a number of ways: a select all button; select specification/classification buttons; or manually. An *interactive brushing* technique is employed to realise the manual selection functionality. In Fig. 1, a selection of lines have been brushed between the times 12:38 and 13:20, and a semi-opaque cyan rectangle has been overlaid on them as a result. Selected traces can be viewed in the PC view by clicking the Explore button below the time-series visualisation. Only these traces will be drawn.

The *PC view* consists of two main components: a Parallel Coordinates visualisation (PCV) and a *situation panel*. These can be seen on the left and right of Fig. 4, respectively.

PCVs (Fig. 2) give users a global view of trends in the data while allowing direct interaction to filter the data set as desired. A set of parallel vertical axes are drawn, which correspond to attributes of the readings in the system. In our case, the readings are records of context data at a certain time, with each axis representing a sensor in the system. Then, a set of n-dimensional tuples, corresponding to data traces in our case, are drawn as a set of *polylines*—a line drawn starting at the leftmost axis and continuing rightwards to the next adjacent and so on, intersecting each axis at the point that represents the value that the context has in that data trace. For example if, in a given situation, a user's computer activity level is "idle", and their location is "canteen", and these two axes are adjacent, then a line will be drawn between those two points. Each axis has a "No value" point which represents that this dimension is missing from any data trace whose polyline passes through it. Polylines are not drawn for times when no data traces were recorded. Each data trace is plotted on the axes and the result is a view of all of the traces, significant and insignificant, that occurred in the system over a period of time. Discrete and quantitative axes can be presented in the same view.

As all the polylines are being drawn within the same area, the technique scales well to large data sets with arbitrary numbers of attributes, presenting a compact view of the entire data set. Axes can be easily appended or removed from the visualisation as required by the dimensions of the data. Situvis also supports reordering of the PCV axes to assist in the identification of correlations between data dimensions.

As Parallel Coordinates have a tendency to become crowded as the size of the data set grows larger, techniques have been designed to cluster or elide sub-sets of the data to allow the dominant patterns to be seen [28]. Direct interaction by interactive brushing to filter and highlight sections of the data encourages experimentation to discover additional information, as seen in Fig. 3. Situvis also includes an approach to reducing clutter using semi-opaque polylines—the default opacity is 0.2, meaning that five lines can be stacked on top of each other before reaching full opacity. This value can be modified as required. Different numbers of overlap will be illustrated using different shades of the colour of the polylines.

Hierarchical clustering [29] uses colour to visually distinguish cases that share a certain range of values into a number of sets, increasing the readability of the diagram. We use a similar technique to group case lines that are assigned to a certain situation, colour-coding these as a group. Different situations can be colour-coded so that the interplay of the context traces that correspond to them can be easily seen. The *situation panel* contains a list of all situations that have been specified in the system. By checking the box beside a situation, the specification will be illustrated on the PCV. The PC view also contains a zoom panel that depicts the area under the mouse cursor magnified to three times its original size. This feature is useful for extracting detailed information from cluttered visualisations.
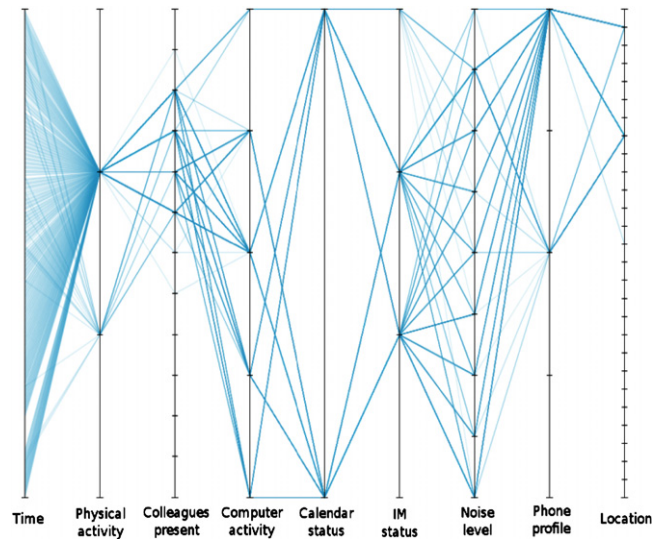
Fig. 2. Part of the main Situvis window showing our Parallel Coordinates Visualisation. This is a view of 420 overlaid context traces with 9 data dimensions gathered over one day. Strong correlations can be seen between the data recorded: the subject spent the majority of the time Idle (the second value on the "Physical activity" axis), with some deviations due to changes in location throughout the day.
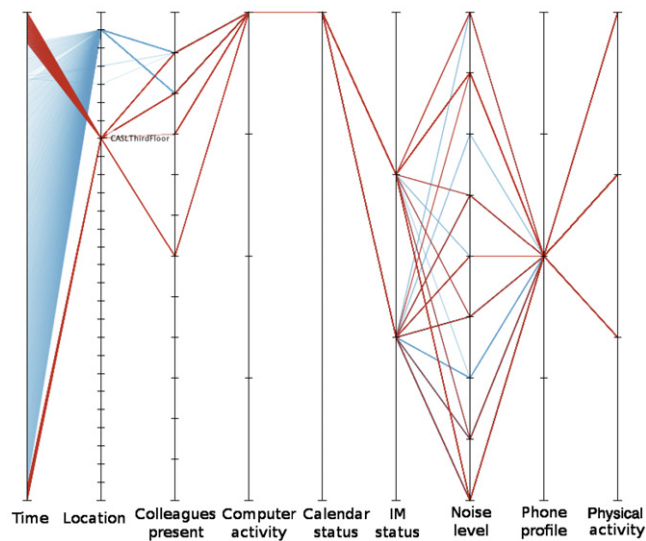

Fig. 3. Here the user has *brushed* over a set of case lines (those that correspond to the Location CASLThirdFloor) by right clicking and dragging a line across them between the first and second axes. This highlights these polylines throughout the diagram, allowing the patterns that occurred at this location to be seen. In this case, the brushed polylines are coloured in red, whereas the unbrushed polylines are coloured in blue. This same operation can be performed on any axis to select any subset of the polylines. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The PC view consists of two separate modes of operation, *edit mode* and *analysis mode*. Both modes are based on the PCV but contain different interactive functionality to fulfill different purposes. The Situvis modes are orthogonal to the development cycle. *Analysis mode* is used for data exploration and contains simple functionality to highlight subsets of the data to get a view of traces across all dimensions in the context of the rest of the data. *Edit mode* is the programming mode, and contains functionality for constructing situation specifications by conjoining assertions about different dimensions of the data.

3.2. Specifying situations with context

The potential for pervasive applications increases dramatically with our ability to aggregate and abstract the many sources of sensor data. Of particular interest to the authors is personal information, or context, and how this information
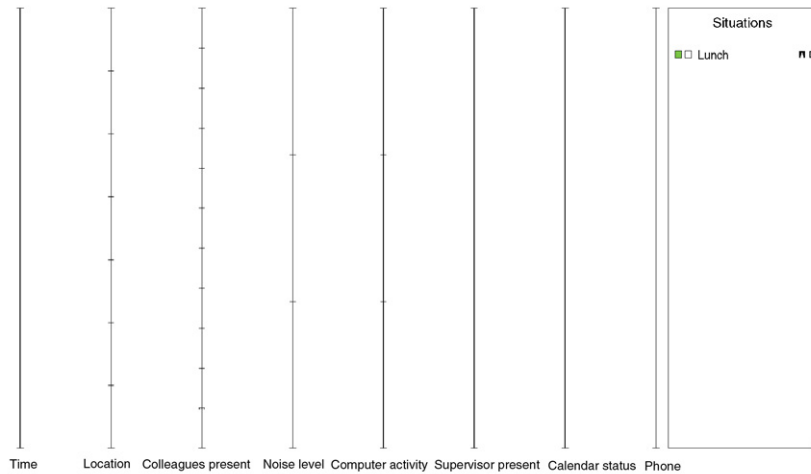
Fig. 4. A view of Situvis with an empty PCV. The situation specification panel on the right of the diagram illustrates that there is one situation specification, `Lunch`, already defined.

can be used to infer the situation of the user. The semantics of situation specifications are described in Section 2.2, and the development process tasks in Section 2.4.

An empty Parallel Coordinates visualisation (i.e., without any polylines plotted) provides a blank canvas for a developer that needs to create a situation specification. The parallel dimensions illustrate the set of context dimensions that are available for the characterisation, and the points on their axes illustrate the scope of the values that these dimensions can acquire. An example of Situvis in this state can be seen in Fig. 4. The developer can create a new, blank specification by clicking the add button ('+' in the bottom right of the figure) and entering a label for it. They can then enter *edit mode* where they are then free to constrain the context dimensions appropriately.

In edit mode, axes representing discrete context dimensions can be constrained by selecting different points in a traffic light system: green signifies that the point is included in the assertion, red signifies otherwise. The quantitative axes can be constrained by creating intervals of one or more points along the axis. If an axis is constrained through multiple points or multiple intervals, $x_1, x_2, \ldots, x_n$, this expresses the assertion semantics $x_1 \vee x_2 \vee \ldots x_n$. Constraining multiple context dimensions (axes) with the assertions $A_1, A_2, \ldots, A_m$, expresses the specification semantics $A_1 \wedge A_2 \wedge \ldots, A_m$.

In order to facilitate the specification of complex situations, Situvis provides three key components: the ability to plot a dataset of sensor readings on the Parallel Coordinates visualisation; a visual representation of a situation specification as a semi-opaque shaded region across the PCV; and support for inputting and representing situation annotations. As we mentioned earlier, it can be impossible to characterise complex situations without a dataset to reference. The required information from such a dataset are the attribute-values around which data from a given situation occurrence clusters, and the correlations between different attributes of the data. The PCV is a technique that allows this information to be easily captured visually.

Annotations of a situation in a dataset can be input into Situvis, and the underlying data traces that they encapsulate can be used as an initial specification of the situation that they represent. Specifications can be overlaid on the trace data as semi-opaque shaded regions, allowing a developer to analyse the traces that fall both inside and outside of the specification constraints.

An example of the situation specification process can be seen in Fig. 5. The user annotated multiple occurrences of a `Lunch` situation. They selected one of these by manually brushing the traces in the TS view, entered the PC view by pressing the `Explore` button with those traces selected, and brushed them in edit mode to create constraints for a new situation. These traces are indicated by the polylines that are completely subsumed by the semi-opaque region. The extra polylines are other traces annotated as `Lunch` that were selected afterwards. By selecting the annotated traces, it is evident what context dimensions characterise them. By comparing the specification to the extra annotated trace lines it can be seen that this initial specification will have to be tweaked as it is too specific in constraining some of the context dimensions.

3.3. Evaluating specifications in relation to data

So far we have described the functionality that allows a user to create a first-cut situation specification from annotated sensor data. Very often, this specification will not be generalised enough because it will be based on a limited amount of situation occurrences, and will have to be tweaked. *Situation refinement* is the process of altering the assertions that make up a situation specification. A developer may have reason to question the correctness of an existing specification (e.g., it caused a situation to be inferred erroneously) and to do this will want to *evaluate the specification in relation to the data* that the sensors produced, as described in Section 2.4.
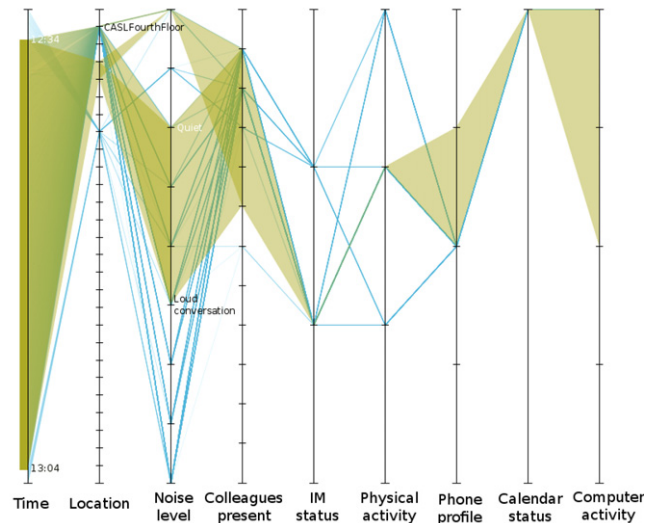
Fig. 5. A view of the Situvis tool with a sample of the Lunch data. Each trace was annotated as the Lunch situation. The highlighted traces illustrate the traces that are part of the current specification of the situation. Labels have been added to the axes for clarity. They are normally shown when the user hovers over the axis.

Both the TS view and the PC view play important roles in completing situation refinement tasks in Situvis. As we mentioned above, the TS view contains a list of existing specifications in the system and the colours assigned to them. The data traces are classified using this set of specifications, and the appropriate segment of the lines in the time-series visualisation are coloured according to the situations that they are classified as. In Fig. 1, a situation called Busy was created and the traces classified by it are coloured in blue in the time-series visualisation. A user can use this method to compare annotated traces to their classifications, free from the details of the attribute-values of the individual traces. From this, they can efficiently identify regions of the dataset that they wish to explore in more detail. On selecting these traces, they can then analyse them in the PC view.

Interactively constraining axes is an important functionality for situation refinement. When existing specifications are overlaid on the trace polylines, the developer can see where they are too strong or weak. Constraints that are too strong will cause the system to sometimes fail in determining when that situation is occurring. Constraints that are too weak may be wrongly interpreted as an occurrence of the specified situation, when in fact a different situation is occurring. By overlaying our specification on top of the polylines, it will be obvious where constraints need to be strengthened, weakened or even excluded altogether. Situvis enables a developer to drag the boundaries of specifications to change the polylines that they cover, essentially changing the constraints of the situation. When the overlaid situation encompasses traces that are not relevant, the user can strengthen the constraints by narrowing the range of values covered by this situation specification (the shaded area in Fig. 5). Similarly, the user can weaken constraints to include traces that happen to fall outside the existing specification by widening the specification, as we have done in Fig. 6.

Sometimes a developer may wish to go beyond simply changing constraints on the axes, and instead encapsulate a further number of polylines by the situation specification. It can be difficult and cumbersome to follow the traces throughout the different dimensions and constrain each in turn. To simplify this task, a user can *brush* trace lines while in `edit mode` and have the situation specification adjusted in order to encapsulate them, as was done in the previous section to create an initial specification from annotated traces. The evolution process is simpler and more intuitive as a result.

3.4. Evaluating specifications in relation to other specifications

Context-aware adaptive systems are very sensitive to incompatible behaviours. These are behaviours that conflict, either due to device restrictions, such as access to a public display, or due to user experiences, such as activating music playback while a meeting is taking place. Situations are closely tied to behaviours—they define envelopes in which behaviour occurs. As a result, their specifications are directly responsible for adherence to compatibility requirements. By harnessing this factor, we can address another key aspect of situation refinement: *evaluating specifications in relation to other specifications*.

Conceptually relating situations to each other from a behaviour compatibility standpoint is an overwhelming task for a developer. We recognise that there are two situation relationships that may lead to incompatibility:

subsumption　if *a* subsumes *b*, and *b* occurs, then *a* will certainly occur.
overlap　if *a* overlaps *b*, then *a* and *b* may co-occur.
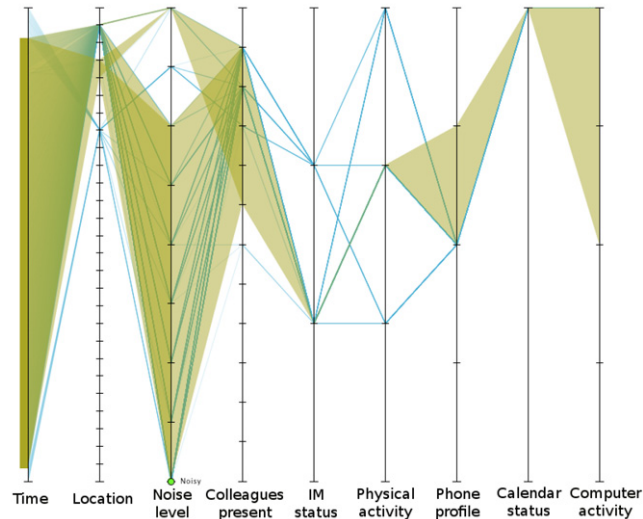
Fig. 6. The user can interactively expand or contract the situation specification along any of the axes. In this case, they have chosen to modify the situation specification to allow for a higher level of noise.

Inconsistencies between situation specifications are commonly not caught until the deployment phase when incompatible behaviours are triggered under the same circumstances. The developer will then have to revise the sensor data produced as in Section 3.3 and adjust one or more situation specifications and maybe even add more sensor infrastructure in order to distinguish between the problematic situations. Reasoning about such inconsistencies can be done before the deployment stage. This process could be automated and the user could be presented with a list of specifications that their specification may co-occur with. However, even given such a list, it may not be intuitive to derive the reason for inconsistencies or to conceptualise ways to avoid them.

Situvis allows multiple situation specifications to each be coloured distinctly. When two or more situations are shown together, the overlap in their constituent contexts is clear, as well as the extent of their dissimilarities. This view allows the developer to alter constraints where necessary, while the overlap and subsumption relationships are refreshed and displayed on-the-fly. A screenshot of this scenario is seen in Fig. 7, which also shows the specification selection panel on the right-hand side. This area allows the user to toggle specifications on and off so that they can be compared and manipulated.

4. Experiments & results

To the best of the authors' knowledge, no tools exist that are purpose-built to assist users in analysing and creating abstractions of pervasive system sensor data. We are aware of tools that allow data to be played back in real-time, such as PlaceLab's Handlense tool and Dey's *a CAPpella* [6], however, these tools are purpose-built for other tasks and do not allow the user to explore the data set as a whole in an efficient manner. An improvised toolset could consist of the following: a tabular representation of the data (e.g., a spreadsheet or database), and a textual programming interface to a logic-based language. For evaluation purposes, we choose the spreadsheet application to represent the data, as we believe that the learning-curve is less steep for an application like Microsoft Excel than it is for learning the syntax of a database query language such as SQL, so we will focus on it here also.

The tabular representation has a number of limitations when it comes to exploring sensor data. Firstly, the maximum amount of information that can be displayed on screen is very low, particularly relative to the potential size of the data set. One effect of this is that it can be time-consuming to view sections of the dataset, not to mention the whole thing. To get a complete perspective of the data, the user must scroll through many *screens* and use human memory to remember patterns that they have seen. Not only must the user scroll vertically through the many data traces, but they may also have to scroll horizontally across the columns if the dimensionality of the data is high.

Patterns can be very difficult to recognise in a tabular representation of sensor data when they exist across multiple columns of data that may not be adjacent. It involves filtering and correlating across multiple columns using human sight and memory alone without any visual aids. Again, this is particularly accentuated as the dimensionality of the data increases.

Comparison of multiple traces of sensor data in spreadsheet format can be very cumbersome, especially when the traces are scattered throughout different screens of the data. One possibility is to paste the traces for comparison into a new document or table, but this can be time consuming and, again, becomes difficult as the magnitude of data exceeds a single screen in size.
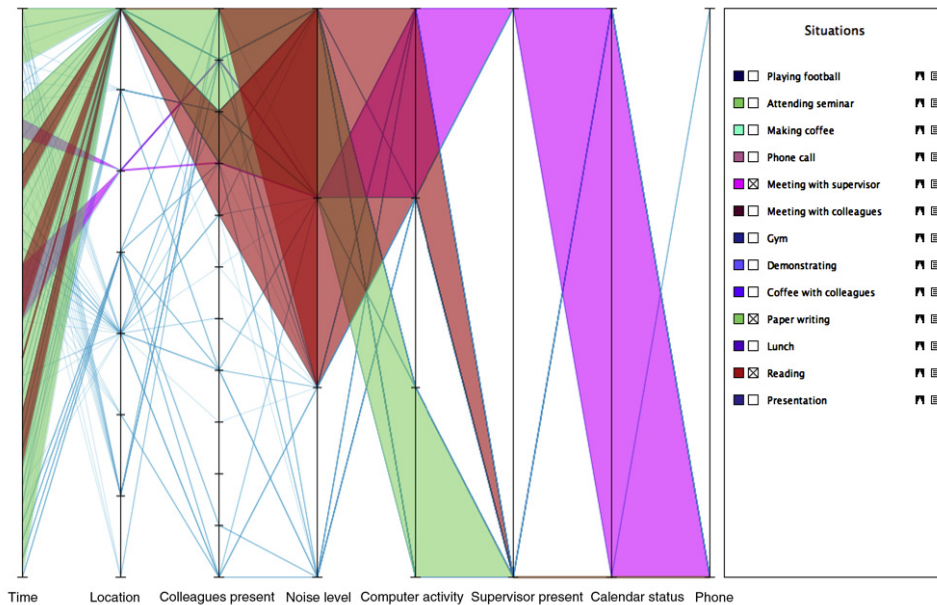
Fig. 7. A view of three distinct situations. Here we are showing the specifications for a meeting with supervisors, paper writing time, and time spent reading. The dissimilarities between these situations are clear from the tool, and the specifications can be further teased apart if required.

4.1. Experiments

Ten participants were recruited to take part in the user study. This number was chosen in order to achieve statistically significant results, however, it is also limited by the number of candidates that were at our disposal. The group consisted of eight Ph.D. students and two postdoctoral researchers, all in the Computer Science domain. There were nine male and one female participants between the ages of 22 and 35. None of the participants were collaborators on the Situvis project, and none of them were used in the pilot study. Seven of the participants were researchers in pervasive computing-related areas such as Human Computer Interaction and distributed systems. The other three were researchers in the areas of formal methods and complex adaptive systems.

We conducted a single factor, between-subjects experiment. Our independent variable was the interface, i.e., Situvis or the improvised alternative (IA), and our dependent variables were Time and Accuracy. These allow us to, firstly, gain a measure of efficiency and, secondly, to report when one tool is more effective than the other for completing a task. The time to complete each task was recorded by the conductor of the experiment. The participant told the conductor when they were beginning a new task, and when they had finished writing the answers for a task. The inclusion of the time taken to record answers in the task-time may make the time taken to complete tasks seem longer than they should be. However, it was necessary to do so because some participants recorded their answers in parts throughout the tasks. The accuracy metric is determined by the task being carried out, as we will explain later.

We chose a between-subjects design, as opposed to a within-subjects one, for two main reasons: we could not randomise the data—we had to use the same dataset for both conditions (Situvis and Excel), and we wanted to avoid fatigue effects caused by lengthy studies. We used a repeated-measures design because of the small number of participants that were available for the study. For each task, the participants were given a number of instances of the task to perform. This number depended on the estimated length of time that it would take to complete a task of this type. The tasks were chosen according to the categories of tasks described in Section 2.4. Participants were given four analysis tasks, two situation specification tasks, two evaluation of specifications in relation to the data tasks, and two evaluation of specifications in relation to other specifications tasks. Each participant was given the same set of tasks to complete. The task order and repeated measures' order were randomised for each participant to avoid learning effects influencing the time and accuracy measures of later tasks.

Both sets of participants were given a brief tutorial of the data representation and tool functionality of the appropriate interface before the experiment. They were also given an opportunity to ask any questions prior to the experiment.

The average time to complete all tasks using Situvis was 44 min and 15 s. The average time to complete all tasks using Excel was 105 min and 25 s.

Tests of significance were carried out following the experiments using two-sample t-tests and Wilcoxon rank sum tests, where appropriate.

Table 1

An average of the results for the analysis tasks for both Situvis and IA participants. The cells in the bottom row of the table contain an average of the figures in the corresponding column. The Situvis group completed this task in 51% less time than the IA group, on average.

Situvis			IA		
Participant	Time	Accuracy (%)	Participant	Time	Accuracy (%)
1	106	100	1	99	100
2	78.75	100	2	258	75
3	50.5	100	3	127	100
4	75.75	100	4	139.5	87.5
5	47	100	5	100.5	100
	71.6	100		144.8	92.5

Analysis task

The first key development task that we evaluated was data analysis. Participants were given four instances of analysis tasks. It took the Situvis group an average time of 4 min 46 s to complete all analysis tasks, leading to an average of 71.6 s for each task. The IA group took an average time of 9 min 39 s to complete all analysis tasks, or 144.8 s for an individual task.

The Situvis group completed the tasks by sometimes filtering the data using combinations of the TS view and brushing in the PC view. They reordered axes when correlations were not apparent, and often visually extracted required information by analysing polylines passing through axis points on the PCV. The IA group sometimes sorted columns of the data, often performed a lot of scrolling, and identified correlations by sequentially scanning row cells.

Table 1 shows the average time and accuracy of all participants for each condition. The Situvis group completed the analysis tasks in 51% less time than the IA group. A one-way analysis of variance determines that there is significant difference between both conditions for the time measurements at the 5% significance level. In case the distributions are not normal,[5] a Wilcoxon rank sum test[6] also indicates a significant difference at the 5% significance level.

4.1.1. Situation specification task

The participants were given two situation specification tasks. For each task, they were given a number of data traces that were annotated by the data collection participant as being a particular situation. Their task was to analyse the data and arrive at a situation specification for that situation. Ideally, the user who collected the data would be the one to specify the situation as it is something that they have experienced and hence they would have the greatest domain knowledge about. However, because we had to compare two approaches we did not want to bias the results by getting the user to complete one study after the other, hence having been exposed to the data before completing the second one.

Each specification was evaluated on annotated test data to arrive at a measure of accuracy. The accuracy is measured as a percentage of annotated traces that the specification classifies. In order to measure false positives, the specifications were also evaluated against the other situations that were annotated and are also measured as a percentage of traces that they cover.

Due to some issues with our data collection process our results are somewhat skewed. However, being aware of these issues allows us to interpret them to some extent. First of all, we evaluate the accuracy of situations against data that the data collection participant annotated. The annotated start and end times of these situations can be quite imprecise. The data collection participant indicated that they estimated a window of error of 2 min on average in a questionnaire given to them after the collection process. We have also found places where the participant mis-annotated situations. For this reason, our measure of accuracy is skewed.

Secondly, our false positive results are biased by the other situations that the participant annotated. In particular, the Lunch situation was one of the few situations that took place in the location CASLFourthFloor. For this reason, we can falsely arrive at very low false positive results for specifications of this situation (i.e., if a participant constrained the Location sensor to be CASLFourthFloor, there were very few data traces, other than Lunch ones, that could be classified by this specification). These situation specification tasks took an average total time of 6 min 36 s and individual time of 196.3 s for the Situvis group. The IA group took an average total time of 16 min 4 s and individual time of 482.4 s.

Table 2 summarises the results for the Situvis and IA participants in completing the situation specification tasks. The Situvis group produced specifications in 59% less time than the IA group. Their specifications were 0.74% less accurate when classifying a test set of traces, but had a 31.8% decrease in false positives. A Wilcoxon rank sum test indicates that the times to complete these tasks under the different conditions are significantly different at the 5.56% level of significance, just 0.56% above the conventional 5% level. There is no significant difference between the accuracies or false positives according to a one-way analysis of variance and a Wilcoxon rank sum test.

---

[5] A Lilliefors test, which is a statistical test to determine whether a sample comes from a normal distribution, was unable to determine that the sample of times for the Excel and Situvis groups were not from normal distributions.

[6] A Wilcoxon rank sum test is a non-parametric test for assessing whether two independent samples of observations are from the same distribution.

Table 2
An average of the results for the situation specification tasks for Situvis and IA participants. The Situvis group produced specifications in 59% less time than the IA group. Their specifications were 0.74% less accurate but had 31.8% less false positives.

Situvis				IA			
Participant	Time	Accuracy (%)	False pos. (%)	Participant	Time	Accuracy (%)	False pos. (%)
1	234.5	69.97	40.66	1	422.5	72.02	39.92
2	51	59.7	24.19	2	952.5	61.82	31.59
3	179.5	57.07	10.69	3	431.5	25.36	5.39
4	267	55.39	14.71	4	188.5	84.3	55.36
5	240.5	65.35	22.21	5	417	66.32	32.62
	196.3	61.5	22.5		482.4	61.96	32.98

Table 3
An average of the results for the specification evaluation tasks for Situvis and IA participants. The Situvis group completed these tasks in 64% less time than the IA group, on average. Their answers were 48% more accurate.

Situvis			IA		
Participant	Time	Accuracy (%)	Participant	Time	Accuracy (%)
1	228	100	1	533	45
2	160.5	100	2	542	70
3	113	100	3	311.5	67.5
4	154	100	4	488	73.5
5	164	100	5	419.5	81.5
	163.9	100		458.8	67.5

Evaluating specifications in relation to the data

Participants were given two tasks where they had to evaluate an existing situation specification against traces that were annotated as that situation. Both groups were provided with the whole four-day dataset that included the annotations. The Situvis group were presented with a visual representation of a situation specification in Situvis and their task was to find where the traces that were annotated as that situation were inconsistent with the specification. The IA group were presented with a set of constraints, represented logically.

The Situvis group took an average time of 5 min 27 s to complete both tasks, or 163.9 s for each task. Their average accuracy in completing the task was 100%. The IA group completed both tasks in an average time of 15 min 17 s, or 7 min 38 s for an individual task. Their answers were 67.5% accurate on average.

To complete this task, the Situvis group used the TS view to select all traces annotated as the situation in question. They then moved to the PC view where they overlaid the specification provided on the selected traces. Finally, they recorded areas where inconsistencies occurred. The IA participants scrolled through the dataset until they found the traces annotated as the given situation. They then analysed whether the traces satisfied the constraints provided, and noted where inconsistencies occurred.

A summary of the participants' results from both groups can be seen in Table 3. The Situvis group performed the task in 64% less time than the IA group. A Wilcoxon rank sum test indicates that this difference is significant to the 1% significance level. The Situvis users provided correct answers 100% of the time, while the IA users provided 67.5% correct answers. Again, a Wilcoxon rank sum test indicates that there is significant difference in accuracy at the 1% significance level.

Evaluating specifications in relation to other specifications

Participants were given two tasks where they had to evaluate a situation specification in relation to other specifications defined in the system. In these tasks, they were given a set of situation specifications and their task was to determine whether some combinations of the situations would definitely co-occur, could possibly co-occur, or would never co-occur, based on the constraints of the specifications. The Situvis group carried out both tasks in an average time of 3 min 18 s, or 99.3 s for each task. Their answers were, on average, 76.7% correct. The IA group completed both tasks in an average time of 5 min 58 s, or 179 s for each task. They arrived at answers that were 93.4% correct on average.

The IA participants completed the task by analysing the constraints that made up the situation specifications and identifying areas where the constraints were distinct, partially overlapped, or completely overlapped. The Situvis group overlaid the relevant specifications on each other in the PC view and identified regions where their semi-opaque areas did or did not overlap.

The Situvis group completed the tasks of evaluating a specification in relation to other specifications in 45% less time than the IA group. A Wilcoxon rank sum test indicates that this difference is significant at the 5% significance level. The Situvis group provided answers that were 18% less accurate than those provided by the IA group. A Wilcoxon rank sum test indicates that there is no significant difference between the accuracies of answers produced by the two groups (Table 4).

Table 4

An average of the results for the tasks of evaluating a specification in relation to other specifications for Situvis and IA participants. The Situvis group completed the tasks, on average, in 45% less time than the IA group. Their answers were 18% less accurate.

Situvis			IA		
Participant	Time	Accuracy (%)	Participant	Time	Accuracy (%)
1	162.5	83.5	1	217	83.5
2	82.5	33.5	2	194	83.5
3	41.5	66.5	3	146.5	100
4	140.5	100	4	161.5	100
5	69.5	100	5	176	100
	99.3	76.7		179	93.4

5. Conclusion and future work

Situvis provides context-aware application developers with a means to understand vast datasets of sensor data in order to efficiently construct specifications of situations to be used as cues for context-aware applications. An integrated programming environment and sensor data representation means that a developer can create specifications by constraining sensors in response to real data traces produced by the system. Results show, with statistical significance, that Situvis can be used to complete the key tasks in the development process in over 50% less time. Although we have demonstrated the power and success of Situvis in supporting the end-to-end development of cues for context-aware application behaviour, limitations also exist which lead to areas for potential future work.

If the dimensionality of the data is very high, it will not be possible to fit all of the required axes onto a typical computer screen. One approach to overcoming this issue would be to stack the lesser significant axes at the edge of the PCV with very little space between them. The more significant ones could be displayed as normal using the rest of the screen. The user could swap axes in and out as required. Another approach would be to project the Parallel Coordinates visualisation onto a sphere that could be rotated to explore more axes than would fit on the computer screen.

Moreover, the number of values that an attribute can acquire could be very high. The minimum horizontal space that can be assigned to a value is a single pixel. If there are more attribute-values than the number of pixels available on the axis, it would be necessary to summarise the data values. Situvis would then require a zooming feature for exploring more detailed attribute-values.

Data analysis using Situvis has shown us that our situation semantics are quite naive. For example, in Fig. 3, a simple interactive brushing of some of the polylines illustrates the temporal nature of the events that the Lunch situation is composed of. We believe that many other situations are constructed of sub-events like these, and if we are to realise the full potential of user behaviour determination, we must expand on our situation semantics to include such temporality.

A feature missing from the current version of the Situvis tool is explicit support for probabilities in situation specifications. In many context-aware applications, robust probabilistic inference is a requirement to handle the naturally fuzzy data in the system. The addition of an overlay which would allow users to set up a probability distribution may be one approach to solving this problem, though this requires a more in-depth study of the treatment of uncertainty in situations.

Acknowledgements

References

[1] A. Miele, E. Quintarelli, L. Tanca, A methodology for preference-based personalization of contextual data, in: EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology, ACM, New York, NY, USA, 2009, pp. 287–298.

[2] K. Henricksen, A framework for context-aware pervasive computing applications, Ph.D. thesis, The School of Information Technology and Electrical Engineering, University of Queensland, 2003.

[3] S. Knox, A.K. Clear, R. Shannon, L. Coyle, S. Dobson, A. Quigley, P. Nixon, Towards Scatterbox: a context-aware message forwarding platform, in: Fourth International Workshop on Modeling and Reasoning in Context in conjunction with Context '07. Roskilde, Denmark, 2007, pp. 13–24.

[4] G. Thomson, G. Stevenson, S. Terzis, P. Nixon, A self-managing infrastructure for ad-hoc situation determination, in: Smart Homes and Beyond - ICOST2006 4th International Conference On Smart Homes and Health Telematics. Assistive Technology Research Series, Amsterdam, The Netherlands, IOS Press, 2006, pp. 157–164.

[5] J. Fails, D. Olsen, A design tool for camera-based interaction, in: CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, New York, NY, USA, 2003, pp. 449–456.

[6] A.K. Dey, R. Hamid, C. Beckmann, I. Li, D. Hsu, A cappella: programming by demonstration of context-aware applications, in: CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, New York, NY USA, 2004, pp. 33–40.

[7] P. Ball, Data visualization: picture this, Nature 418 (6893) (2002) 11–13.

[8] A.K. Clear, R. Shannon, T. Holland, A. Quigley, S. Dobson, P. Nixon, Situvis: a visual tool for modeling a user's behaviour patterns in a pervasive environment, in: Pervasive 2009, the Seventh International Conference on Pervasive Computing. Nara, Japan, 2009, pp. 327–341.

[9] A. Penta, A. Picariello, L. Tanca, Multimedia knowledge management using ontologies, in: MS '08: Proceeding of the 2nd ACM workshop on Multimedia semantics, ACM, New York, NY, USA, 2008, pp. 24–31.

[10] C. Ghezzi, A. Mocci, M. Monga, Synthesizing intensional behavior models by graph transformation, in: ICSE'09, 2009, pp. 430–440.

[11] B. Logan, J. Healey, M. Philipose, E.M. Tapia, S. Intille, A long-term evaluation of sensing modalities for activity recognition, in: UbiComp 2007: Ubiquitous Computing, 9th International Conference, Innsbruck, Austria, 2007, pp. 483–501.

[12] A. Krause, A. Smailagic, D.P. Siewiorek, Context-aware mobile computing: learning context-dependent personal preferences from a wearable sensor array, IEEE Transactions on Mobile Computing 5 (2) (2006) 113–127.

[13] T. Huýnh, M. Fritz, B. Schiele, Discovery of activity patterns using topic models, in: UbiComp 2008: Ubiquitous Computing, 10th International Conference. Seoul, South Korea, 2008, pp. 1–10.

[14] A.K. Clear, S. Knox, J. Ye, L. Coyle, S. Dobson, P. Nixon, Integrating multiple contexts and ontologies in a pervasive computing framework, in: C&O 2006: ECAI 2006 Workshop on Contexts and Ontologies: Theory, Practice and Applications, Riva Del Garda, Italy, 2006, pp. 20–25.

[15] J. Coutaz, G. Rey, Foundations for a theory of contextors, in: CADUI: 4th International Conference on Computer-Aided Design of User Interfaces, Kluwer, Valenciennes, France, 2002, pp. 13–34.

[16] A.K. Dey, Understanding and using context, Personal Ubiquitous Computing 5 (1) (2001) 4–7.

[17] J. Coutaz, J. Crowley, S. Dobson, D. Garlan, Context is key, Communications of the ACM 48 (3) (2005) 49–53.

[18] J. Ye, L. Coyle, S. Dobson, P. Nixon, A unified semantics space model, in: Location- and Context-Awareness, in: LNCS, vol. 4718, Springer, 2007, pp. 103–120.

[19] J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, M. Spiteri, Generic support for distributed applications, Computer 33 (3) (2000) 68–76.

[20] D. Salber, A.K. Dey, G.D. Abowd, The context toolkit: aiding the development of context-enabled applications, in: CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, New York, NY, USA, 1999, pp. 434–441.

[21] E. Katsiri, J. Bacon, A. Mycroft, An extended publish/subscribe protocol for transparent subscriptions to distributed abstract state in sensor-driven systems using abstract events, IEE Seminar Digests 918 (2004) 68–73.

[22] E. Katsiri, Middleware support for context-awareness in distributed sensor-driven systems. Ph.D. thesis, Computer Laboratory, University of Cambridge, 2005.

[23] C. Bolchini, C.A. Curino, E. Quintarelli, F.A. Schreiber, L. Tanca, Context information for knowledge reshaping, International Journal of Web Engineering and Technology 5 (1) (2009) 88–103.

[24] S.W. Loke, Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective, The Knowledge Engineering Review 19 (3) (2004) 213–233.

[25] G. Andrienko, N. Andrienko, S. Wrobel, Visual analytics tools for analysis of movement data, SIGKDD Explorations Newsletter 9 (2) (2007) 38–46. Special issue on visual analytics.

[26] T. Tenev, R. Rao, Managing multiple focal levels in table lens, in: Infovis 1997: IEEE Symposium on Information Visualization, IEEE Computer Society, Washington, DC, USA, 1997, p. 59.

[27] C. Reas, B. Fry, Processing: a learning environment for creating interactive web graphics, in: SIGGRAPH '03: ACM SIGGRAPH 2003 Sketches & Applications, ACM, New York, NY, USA, 2003, p. 1.

[28] A. Artero, M. de Oliveira, H. Levkowitz, Uncovering clusters in crowded parallel coordinates visualizations, IEEE Symposium on Information Visualization (2004) 81–88.

[29] Y.-H. Fua, M.O. Ward, E.A. Rundensteiner, Hierarchical parallel coordinates for exploration of large datasets, in: VIS '99: Proceedings of the conference on Visualization '99, IEEE Computer Society Press, Los Alamitos, CA, USA, 1999, pp. 43–50.