Situvis: Visualising Multivariate Context Information to Evaluate Situation Specifications

Adrian K. Clear, Ross Shannon, Thomas Holland,
Simon Dobson, Aaron Quigley and Paddy Nixon
Systems Research Group
School of Computer Science & Informatics
UCD Dublin
Ireland
adrian.clear@ucd.ie

ABSTRACT

One of the many challenges faced when evaluating context-aware ubiquitous systems is to gain some understanding of the constant influx of context data into the system. Elsewhere, context has been distilled into more natural abstractions called *situations* with the aim of making these systems more understandable and intuitive to develop applications for, though codifying and manipulating these situations still presents problems. We introduce Situvis, a tool we have developed based on the Parallel Coordinates Visualisation technique, which assists users by visually representing the conditions that need to be present for a situation to be triggered in terms of the real-world context that is being recorded in their system, and allows the user to visually inspect these properties, evaluate their correctness, and change them as required. We describe the use of our tool with a small user study.

Author Keywords

Context, context-aware systems, situations, situation programming, visualisation, visual data mining

ACM Classification Keywords

D.1.7 Visual programming,
H.1.2 Human information processing

INTRODUCTION

In context-aware systems, context data is derived from multiple heterogeneous sensors. These sensors may be networked physical instruments in the environment (measuring factors like temperature, noise volume or humidity) or software sensors retrieving information from the web or various data feeds. These context data are voluminous, highly multivariate, and constantly being updated as new readings are recorded.

Situations have been proposed as a higher-level abstraction of context data [8], freeing the user from having to deal with raw context and allowing more expressive adaptations. Situations are more natural for users to work with, as they 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. 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" [10].

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 the data.

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 [2]. Interactive visualisation tools help the viewer perform visual data analysis tasks: exploring patterns and highlighting and defining filters over interesting data.

Here, we present Situvis, a scalable visualisation tool for illustrating and evaluating the makeup of situations in a context-aware ubiquitous system. Our tool is based on well-founded situation specification semantics. By incorporating real situation traces and annotations, Situvis assists system developers in constructing and evaluating sound and complete situation specifications, affording them a better understanding of the situation space, and the reliability of modelling 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 a formal description of situation specifications and a review of some challenges faced when working with context and situations. We then describe the details of the Situvis tool, including a demonstration of its utility, followed by an informal evaluation and discussion of its properties.

CONTEXTS AND SITUATIONS

Situation specifications

Based on our experience with modelling context for adaptive systems [4, 10], and from the extensive literature on the

UbiComp '08 Workshop W2 – Ubiquitous Systems Evaluation (USE '08)
September 21st, 2008
This paper is not an official publication of UbiComp '08.

subject [5, 6, 8], we can make some observations: the incoming sources of context 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 values, and may be recorded by numerous disparate positioning systems, but is modeled as a higher-level abstraction to make it easier to reason with. We have previously completed research that allows component x, y and z coordinates to be composed into a symbolic representation, given some domain information [15], and so can work with locations as readable as "Simon's office" or "Coffee Area". Our visualisation tool can accept either raw sensor data or these higher-order categorised data.

Situations are high-level abstractions that serve as a suitable model with which to develop context-aware systems. In order for a system to be able to recognise situations, they must first be specified. The semantics of situation specification can be seen in the work of Henricksen [8] and Loke [11]. Based on this work, we make some assumptions about situation specification so that situations specified using declarative languages such as these could simply be "plugged-in" to our tool.

Situation specifications are boolean expressions (sometimes called assertions in computer programming)—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 operators can be defined to complement these operators. For example, for location we could define a "distance" operator. The distance operator may take two arguments and return a numerical value of the distance between them. Domain-specific operators are also required for situation specification. For example, for many context dimensions an essential operator is one that takes a value and a range and returns true if that value exists within the range.

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.*

Properties of context-awareness

Situation specifications are essential in achieving two very important properties of context-aware systems: *soundness* and *completeness* [11]. A system is sound if it does not give false positives when determining a situation; it is complete if it contains specifications for all situations to be detected.

Related to these properties is *perceptual distinguishability*. If a system state exists that involves multiple situations, the observer should be able to distinguish between those situations. It may be the case that two situations' specifications are satisfied by a given occurrence. In the case of compatible situations, this could be due to one situation's specification subsuming another. If the specifications are incompatible, however, we must have a means to re-evaluate them. Situations are generally specific to behaviours, and, as a result, their compatibility requirements are determined by the compatibility requirements of behaviours.

The adaptive systems that we are concerned with are user-centric, and so user feedback is an important aspect of their evaluation. Situation annotation is a particularly useful mode of feedback for us, because it allows us to contrast situation specifications with actual traces of these situations. It also exposes the subjective nature of situations. However, to develop sound and complete situation specifications, it is necessary to capture two facets of reality: those situations that our specification must successfully characterise; and those situations that it should not. Therefore, the annotated situations are an important guideline, but the traces of undesired situations are also important to avoid false positives.

Situations can range from the very simple to the very complex, depending on the number of contextual components they are defined over. The more complex a situation becomes, the more difficult it is to pick out similarities or differences between multiple situations in aggregate, without the support of a visual analytics tool. This ability is important because the similarity of one situation to another determines the possibility of them occurring together or in sequence within a small period of time.

Visualisation of context data

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 [14]. 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.

An alternative to a single visualisation are co-ordinated, linked visualisation techniques employing brushing and querying. Three linked views of multi-dimensional data—using a Principal Components Analysis (PCA) view, interactive brushing, and dimensional querying with parallel bargrams—are employed in the Antenna Design Gallery [12]. Here the actions or queries in any one window or view are reflected in

all. A user selecting a range within a given dimension reduces the data cases in the PCA view, and in the element values highlighted in the other bargrams. This encourages exploration of a large, multivariate data set, as different facets of the data can be seen in each view.

PARALLEL COORDINATES

Parallel Coordinate Visualisations (PCVs) are a standard two-dimensional technique ideally suited to large, multivariate data sets [9]. The technique excels at visually clustering cases that share similar attribute values across a number of independent discrete or continuous dimensions, as they can be visually identified through the distribution of case lines within the visualisation [3]. The user can see the full range of the data's many dimensions, and the relative frequencies at which values on each axis are recorded. These features are visible in Figure 1, which shows context data from our user study, which we will describe in the next section.

PCVs 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 horizontal 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 are drawn as a set of purple *polylines* which intersect each axis at a certain point, corresponding to the value recorded for that attribute. Discrete and quantitative axes can be presented in the same view.

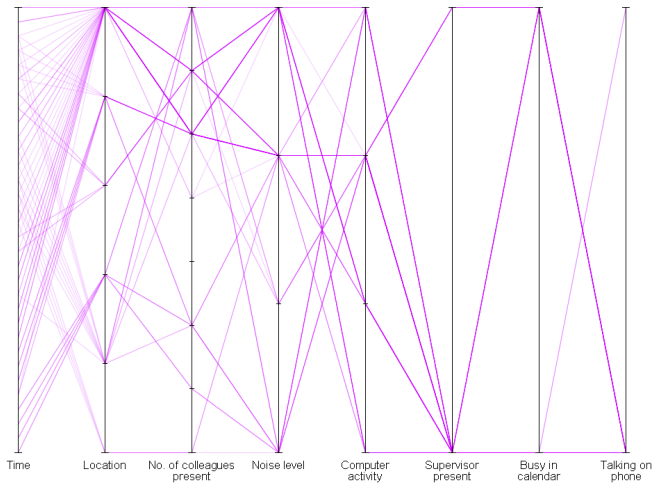

Figure 1. Our Parallel Coordinates Visualisation. This is a view of 96 overlaid context traces with 8 data dimensions gathered over 3 days. Strong correlations can be seen between the three days recorded: the subject spent the majority of all three days at their desk (the first value on the "Location" axis), with some deviations due to coffee breaks or visits to their supervisor's office at irregular times.

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.

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 [1]. Direct interaction to filter and highlight sections of the data encourages experimentation to discover additional information.

Hierarchical clustering [7] 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.

EVALUATING SITUATIONS WITH SITUVIS

Description & case-study

Situation-awareness is commonly applied to adaptive systems as a means to introduce useful cues for automatic behaviour adaptation. System developers are tasked with codifying situations that their system should respond to by tying together loose configurations of sensor readings. Because of the constant fluctuation in the values of these sensor readings (due to minute changes in the property being measured, or due to the accuracy of the sensor), situation definitions are frequently composed of a set of ranged intervals that give the developer some more latitude to cover more of the sample space than if they had to define a situation for every possible combination of sensor readings. When these ranges are all logically conjoined and the value from each sensor falls within range concurrently, the behaviour attached to this situation is invoked.

Situvis is built using Processing [13], a Java-based visualisation framework which supports rapid prototyping of visualisation techniques.[1] Each context dimension is represented in Situvis as a vertical axis. Each axis is divided equally based on the number of values that could be recorded for this dimension. For example, the axis for location contains six points representing the symbolic locations that we chose to include in our analysis. A situation trace is represented as a polyline—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 situation 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 situation trace is plotted on the axes and the result is a view of all of the situations, significant and insignificant, that occurred in the system over a period of time.

In order to carry out our case-study, we required real context data with which we could characterise situations. We chose to gather context data and situation annotations manually over a three day period. While the capabilities exist to collect these context data automatically, we chose to collect the data through manual journaling, so that we did not

[1] You can read more about Situvis and interact with a demo of the software at http://situvis.com.

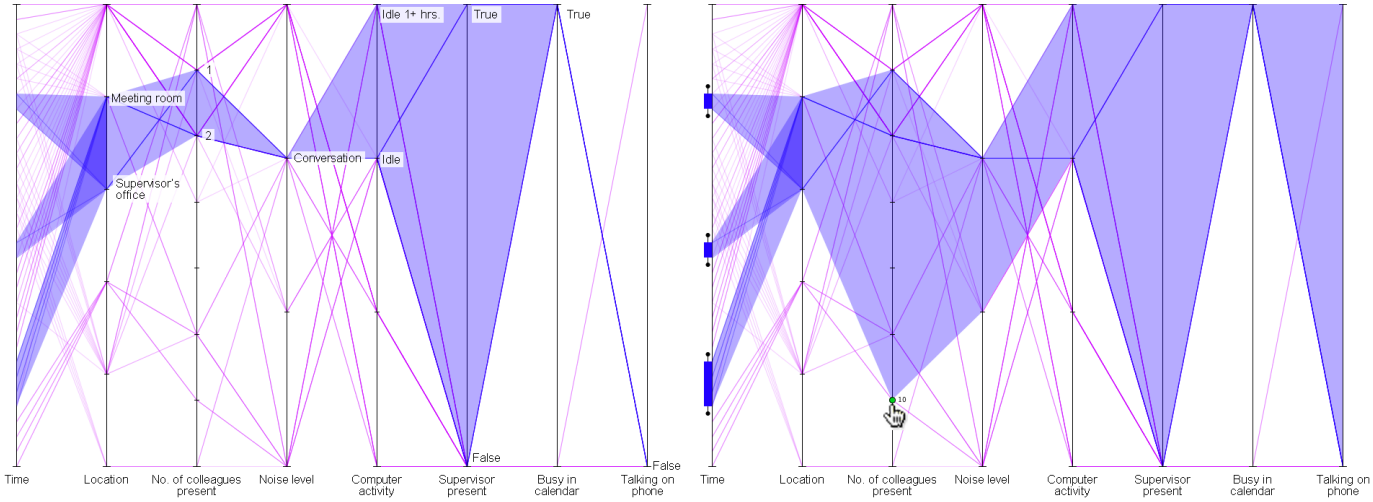

Figure 2. A view of the Situvis tool showing 3 days of a user's context, with traces annotated as being in a "meeting" situation highlighted (left). These situations occurred at many different times throughout the day in two different locations, with a range of values for the other contexts. The user can interactively expand or contract the situation definition along any of the axes. In this case, they have chosen to modify the situation specification to allow for more colleagues to be present, the noise level to be greater and the possibility of talking on the phone (right).

need to factor in issues with the aggregation, uncertainty or provenance of the context data.

Our trial subject recorded their context every fifteen minutes (10am–6pm) for three consecutive weekdays. They captured context in the form of time, location, noise-level, number of colleagues present, their supervisor's presence (true or false), their phone use (either taking a call or not), calendar data (being busy or having no appointments), and computer activity. For simplicity, the noise-level was recorded on a 4-point scale of quiet, conversation, chatty, and noisy. Likewise, computer activity level was scaled as idle for an hour or more, idle for less than an hour, active, and highly active. We defined six symbolic locations: meeting room, canteen, sports center, supervisor's office, subject's desk, and a lecture theatre. Figure 1 shows a view of the Situvis tool with all of these traces plotted together in one view.

The subject also annotated what, if any, situation they were in at the time of data capture. These annotations are used in Situvis to identify situations that require specification in the system.

Specifying situations with context

Situation specifications are structured according to the definition we discussed previously. Situvis enables a developer to select all occurrences of a given annotated situation, and add further cases to this definition using interactive brushing of polylines, or by dragging a range indicator on the left of the axis to expand or contract the range of values covered by this specification. The user can evaluate existing situation specifications overlaid against actual trace data and see where they need to be modified.

An example of this process can be seen in Figure 2. The trial subject annotated multiple occurrences of a "Meeting" situation. By selecting these traces, it is evident what context dimensions characterise them. We can see that "Time" and "Supervisor presence" are not useful due to the multiple split lines on their axes. Hence they are ineffective when defining constraints. The specification is clear from the other dimensions, however, and could be expressed as:

$$
\begin{aligned}
&\{1 \geq \text{No.colleagues} \leq 2\} \wedge \\
&\{\text{Location} = (\text{meeting room} \\
&\qquad \vee \text{supervisor's office})\} \wedge \\
&\{\text{Phone use} = \text{none}\} \wedge \\
&\{\text{Computer activity} \geq \text{idle}\} \wedge \\
&\{\text{Noise-level} = \text{conversation}\} \wedge \\
&\{\text{Calendar} = \text{busy}\}
\end{aligned}
$$

None of these values alone can characterise "Meeting", as the trace data illustrates. Furthermore, each dimension may not always be available. Situvis allows one to identify combinations of dimensions which, when taken together can provide a good estimation of the situation. For example, "Location" taken with "No. of colleagues" is a good indication of "Meeting", as the interval that they create does not contain polylines that characterise different situations. This can also give system developers an insight into which sensors in their system are the most useful, and which types of sensors they should invest in to gain the most added benefit in terms of the expressiveness of their system.

Situation evolution

When existing specifications are overlaid on the trace polylines, the developer can see where specifications 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. When the overlaid situation encompasses traces that are not relevant, the user can strengthen the constraints. Similarly, the user can weaken

constraints to include traces that happen to fall outside the existing specification.

We hypothesise that as more trace data is added and annotated, the constraints that we have defined for "Meeting" may be too strong. 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.

Situation evaluation

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 compatibility requirements. By harnessing this factor, we can address another key aspect of situation evaluation.

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.

Our tool 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 scope 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 Figure 3.

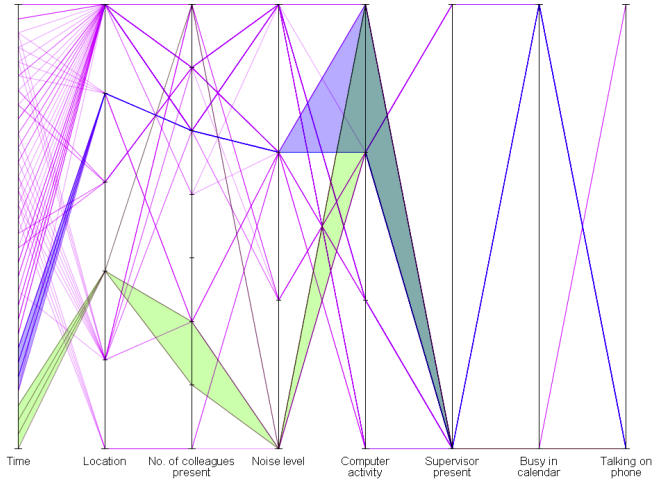

Figure 3. A view of two distinct situations. The higher blue range is a meeting situation, whereas the lower green range is a seminar that occurred after normal work hours. The dissimilarities between these situations are clear from the tool, and the specifications can be further teased apart if required.

DISCUSSION

We have shown, using a case-study, the utility of Situvis in the situation specification and evaluation processes. The traditional approach to situation specification is subject to limitations: context constraints are based on static definitions of these concepts in knowledge representation structures (e.g., ontologies); they are derived from a developer's conceptual understanding of certain situations; and do not contain methodologies for ensuring soundness and completeness. To address these limitations, Situvis presents a developer with a reference point for situation specification and evaluation through the display of actual trace data and situation annotations. The relevance of context to a specification is made clear, and contrasting situation traces can be used as a guide for specification.

Context-aware systems are dynamic—sensors, users and habits are constantly changing. Hence, we cannot expect situation specifications to remain static. It must be possible to re-evaluate them accordingly. Current approaches entail the modification of constraints based on data logs or experience. Situvis allows developers to visually overlay specifications on traces, and tailor their constraints as a result. Unlike traditional methods, Situvis clearly depicts cases where constraints are too strong or too weak.

In addition, we have identified a scenario that a tool like Situvis could address in the future. It is concerned with the notion of *closeness* of situations. Two situation specifications are close if small changes in context can cause an evolution from one to the other, a property easily identified from context constraints. Close situations may be significant as the transition step from one to the other is small in terms of probable context changes. Visualising these relationships in Situvis will allow a developer to identify the following: suggestions of areas where the situation associated with a behaviour may be incomplete; and points where the system behaviour may be unstable. The former is used to increase developer awareness of situation-behaviour associations that they may have omitted. The latter is useful for highlighting obtrusive behaviours associated with close situations—points where a see-saw-like cycle from one to another may occur in a short time frame, resulting in an erratic user experience. One can thus introduce inertia by strengthening constraints, making the transition step between them larger.

Some contexts may be relevant only when combined with other dimensions. Ideally, we would display all of the context information that is available in the system for a particular situation annotation. A developer could then eliminate contexts that are not useful based on visual analysis. However, we have yet to evaluate the feasibility of this approach in large-scale systems.

Some other context dimensions are also not easily represented on a line. In particular, Location, with its domain

relations like subsumption, is difficult to represent in two dimensions. We are researching techniques to flatten hierarchies for a more intuitive representation.

CONCLUSIONS AND FUTURE WORK

We have presented Situvis, a tool in development which uses a Parallel Coordinate Visualisation to illustrate situation traces and specifications. The tool assists developers in describing situations through direct interaction, providing a natural interface for a developer of context-aware systems. By stacking many instances of context together in one view, it becomes simple to inspect the correlation between situation specifications and the actual situations that occur during deployment. Situvis enables a developer to evaluate the soundness and completeness of situation specifications within the framework of real data.

By visually analysing the overlap of situation specifications within their system, the developer can identify where multiple situations require similar context values to be activated. Such overlaps may imply problems in the situation specifications, as conflicting behaviours may be triggered by conceptually similar situations. Thus, the developer can compare situations against others, and change the situation's specifications to become stronger or weaker as necessary.

We are developing a metric of the closeness of situations for use in evaluating soundness and completeness. Close situations may frequently occur one after the other, which may lead to unpredictable system behaviour from a user's perspective. We hope that the Situvis tool will prove useful in helping to avoid this oscillation.

A weakness of the current version of the Situvis tool is that it does not explicitly support probabilities in situation specifications. In many context-aware applications, robust probabilistic inference is a requirement to handle the naturally fuzzy data in the system. We are considering the addition of an overlay which will allow users to set up a probability distribution, though this requires a more in-depth study of the treatment of uncertainty in situations.

We are ongoing in our investigation of properties of situations that can be exploited for further evaluation. For example, Situvis could also be used by users of the context-aware system as a gateway to user programming: helping them to unroll the cause of a situation activation, so that they can gain insight into why the system began to behave as it did.

Acknowledgements: This work is partially supported by an EMBARK Scholarship from the Irish Research Council for Science, Engineering and Technology and by Science Foundation Ireland under grant number 03/CE2/I303-1, "LERO: the Irish Software Engineering Research Centre."

REFERENCES

1. A. Artero, M. de Oliveira, and H. Levkowitz. Uncovering clusters in crowded parallel coordinates visualizations. *Information Visualization, 2004. INFOVIS 2004.*, pages 81–88, 2004.
2. P. Ball. Data visualization: Picture this. *Nature*, 418(6893):11–13, 2002.
3. S. Card, J. Mackinlay, and B. Schneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999.
4. A. K. Clear, S. Knox, J. Ye, L. Coyle, S. Dobson, and P. Nixon. Integrating multiple contexts and ontologies in a pervasive computing framework. In *Contexts and Ontologies: Theory, Practice and Applications*, pages 20–25, Riva Del Garda, Italy, August 2006.
5. J. Coutaz and G. Rey. Foundations for a theory of contextors. *Computer Aided Design of User Interfaces*, 2002.
6. A. K. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5(1):4–7, 2001.
7. Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *VIS '99: Proceedings of the conference on Visualization '99*, pages 43–50, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.
8. K. Henricksen. *A Framework for Context-Aware Pervasive Computing Applications*. PhD thesis, The School of Information Technology and Electrical Engineering, University of Queensland, September 2003.
9. A. Inselberg and B. Dimsdale. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *VIS '90: Proceedings of the 1st conference on Visualization '90*, pages 361–378, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.
10. S. Knox, A. K. Clear, R. Shannon, L. Coyle, S. Dobson, A. J. Quigley, and 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*, pages 13–24, Roskilde, Denmark, August 2007.
11. S. W. Loke. Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective. *The Knowledge Engineering Review*, 19(3):213–233, 2005.
12. A. Quigley, D. Leigh, N. Lesh, J. Marks, K. Ryall, and K. Wittenburg. Semi-automatic antenna design via sampling and visualization. *Antennas and Propagation Society International Symposium, 2002. IEEE*, 2, 2002.
13. C. Reas and B. Fry. Processing: a learning environment for creating interactive web graphics. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Sketches & Applications*, pages 1–1, New York, NY, USA, 2003. ACM.
14. T. Tenev and R. Rao. Managing multiple focal levels in table lens. *Infovis*, 00:59, 1997.
15. J. Ye, A. K. Clear, and S. Dobson. Towards a formal semantics for pervasive adaptive systems. *Under revision for The Computer Journal*, 2008.