Telecare Service Challenge:

Conflict Detection

Jesse Blum
Institute of Computing Science and Mathematics,
School of Natural Sciences, University of Stirling
Stirling, Scotland
jmb@cs.stir.ac.uk

Evan Magill
Institute of Computing Science and Mathematics,
School of Natural Sciences, University of Stirling
Stirling, Scotland
ehm@cs.stir.ac.uk

***Abstract*—Telecare and telehealth system services can be dynamically configured to collect, analyse, store, and adapt to multimodal data about people as they go about their activities of daily life. These services need to be able to personalise to subjects and adapt to changes in lifestyles, environments and technology. Such dynamic adaptability may be well supported by a low-level rule programming approach; however measures may need to be taken to limit the emergence of conflicts between the distributed rulesets owing to differing programmatic assumptions and unexpected changes.**

Here, we consider types of conflict that might arise when a variety of care devices are brought together and begin to rely on each others' services. This paper describes a distributed rule-based conflict detection approach for use with heterogeneous mobile and home care devices. We propose methods that make it possible to detect certain forms of rule conflict. To do so, we introduce Event Calculus based logic for writing device rules and an analytical framework for conflict detection.

Keywords-service conflict detection; rule-based sensor networks; ambulatory assessment

I. INTRODUCTION

Mobile and sensor technologies are used for care and healthcare purposes across a wide range of settings including the home, care homes, and hospitals. They are increasingly being used to monitor individuals on the move and indeed such ambulatory assessment can be used in preference to hospital visits.

Regardless of the setting however, often such telecare systems cannot be altered, and so all individuals will experience the same system for the duration of its use. Ambulatory assessment systems could gain considerably from an ability to be personalised to the subjects being monitored [1]. A significant motivating factor for personalisation is that diseases can manifest differences both between individuals (inter-individual) and over time for a particular individual (within-individual). This is particularly true for long term assessment. The types of sensors used and their patterns of usage must, therefore match these transient subject states. So the system must be both personalised for an individual and altered for that individual over time.

Statically designed solutions are insufficient at handling the level of change that long term telecare implies since the designers would have to have a priori knowledge of the changes to the behaviour of their subjects and the deployment environments. An alternative approach is to program semi-autonomous devices to form into dynamic ad hoc coalitions that provide each other with services. The rules governing each part of the telecare system can be managed separately to personalise the telecare solution for each subject. The rules of the overall system behaviour will therefore be distributed amongst the network components and dynamically change in time, but not necessarily known to or programmed by any particular individual.

Our research into such a rules-based approach is part of the Personalised Ambient Monitoring (PAM) project, which is investigating the feasibility of reducing the incidence of debilitating episodes through personalised ambient monitoring of affective disorder patients in their homes. We are attempting to collect patient activity signatures in an ambient and unobtrusive manner. System personalisation is a core issue for PAM since activity signatures differ amongst patients and can change over the course of patient lifetime. The types of sensors used, and their patterns of usage, must be personalised in order to match patient states and be accepted by users. System personalisation requires a dynamic and flexible programming method but it must also be easy to program, represent domain information and above all result in correct system behaviour. One of our goals was to collect micro-data from long term repeated sampling of people going about their lives at home and on the move using a changing set of heterogeneous worn and environmental devices. We acknowledge that people change in time and that individuals vary with respect to their concerns and disease manifestations, and therefore we placed a great deal of value on the ability for the system to be personalised.

We evaluated the effectiveness of the technology in technical trials with control participants [2]. Reliability concerns arose from our desire to use such a dynamic network. Changes could lead to different device rules interfering with each others' operations. Device rules can conflict such that the functionality of one device may modify another in unexpected ways.

The main contribution of this paper is to show how a collection of devices can be used for telecare and to show that it is possible to detect various types of conflict in a rule-based

PervasiveHealth 2011, May 23-26, Dublin, Republic of Ireland

DOI 10.4108/pervasivehealth.2011.246080

approach to programming telecare solutions. Detecting conflicts should aid in reducing reliability concerns and thereby increase adoption of rule-based telecare. We have developed rule conflict investigation tools. These tools permit dynamic and straightforward personalisation of network behaviour. They also support additional equipment as and when they become available.

II. Related Work

A. Ambulatory Assessment

This work focuses on mental health assessment. Although it is not a new concept, ambulatory psychological/psychiatric assessment has recently begun to emerge as an important tool for clinicians and researchers as a result of methodological and technological trends. Questions have arisen in psychology/psychiatry concerning how closely subjective reports from questionnaires and laboratory findings match real world in-context behaviour [3]. Concurrently, sensors, computers, and communication devices are becoming smaller, more reliable, less expensive and easier to use.

Ambulatory assessment promises to provide clinicians, researchers, and individuals with real-time collected, ecologically valid, unbiased contextualised data about symptoms, physiology, activity, behaviour and mental/emotional state [4]. It has been suggested that individualised interactive moment-specific real-world treatment could be provided based on ambulatory assessment systems, and longitudinal continuous data could greatly enhance social science research. Ambulatory assessment systems are still in their infancy and much work remains to be addressed. This paper proposes techniques to ensure that personalisation results in a stable system. In particular this work addresses rule-based systems that support dynamic real time system behavioural changes by allowing the rules to be changed at run-time. It is crucial that the changing rules remain consistent and do not degrade system performance.

B. Rule-based Sensor Networks

Rule-based middleware for sensor networks has been used in a number of projects such as [5] - [8]. These studies show that the programming and concurrency models are simplified compared with other approaches. Furthermore they indicate that program correctness is easier to prove, and that rule-based systems remain sufficiently expressive at high conceptual levels. Also rule notations that employ an event driven paradigm find favour in sensor networks; whereas an imperative paradigm does not.

```
dstp(T1) :-
    T2 is T1+1 ... T7 is T1 + 6,
    initiallyN(connection),
    initiallyP(message),
    happens(listen_for_connection,T1),
    happens(transfer_data,T3),
    happens(process_data,T4),
    happens(store_data,T5),
    happens(dstp(T7),T7),
    initiates(listen_for_connection,
    connection,T2),
    terminates(transfer_data,connection,T4),
    terminates(process_data,message,T6).
```

Figure 1. Example Data Storage Through Processing feature rule

More generally rule-orientation is seen as a more natural way to express programs for sensor networks. It was pointed out by [8] that application developers using rule-oriented middleware are protected from complexities arising from tight real-world integration, network dynamics, and resource limitations. Rule based systems have been built that allow the rules to be changed at run time [8]. This is very attractive for personalised systems that must change over time. Maintaining a consistent set of rules across the system, however, is challenging.

C. Rule consistency

In rule-based systems where rules may originate from a number of sources and end up being executed across a number of destinations, there is a strong possibility of the rules being inconsistent and causing behavioural conflict. This has been noted in [8], where they discuss the importance of detecting and resolving such conflicts. However that paper did not address a method to do it; rather by not employing rules between nodes but only accepting them from a single trusted server, they avoided this requirement. The trusted server employed meta-rules [9] to ensure conflict was resolved within the server and so conflicting rules were not distributed.

In this paper we draw on a wider literature of programming conflict frequently described as Feature Interaction [10]. This topic was initially addressed in telephony, but has expanded to a wide range of domains experiencing program or control conflict; such as cars, lifts, internet services, and building control. Here the focus is on rule conflict in telecare sensor networks.

III. Our Methods

Attempting to personalise devices and networks to monitor subjects in-situ exposes the need to use adaptable programming approaches. Reliability concerns arise, however, from having the need for a dynamic network, with features that change in time. These dynamic changes could lead to different features interfering with each others' operations.

We have been considering whether a rule-based approach to programming devices provides a natural way to express device behaviour whilst limiting the risks of rule conflict. Here we present Event Calculus based feature rule descriptions and conflict analysis rules.

A. Event Calculus Based Rules

The Event Calculus was designed by Kowalski & Sergot as a way of representing and reasoning about actions and their effects in time [11]. It is expressed using the Horn clause subset of first-order predicate logic and its ontology contains three main concepts: fluents, actions (or events) and time points. Fluents are properties of the universe of discourse that can change in time. These properties may either take a propositional form such as "the subject is in the house" or a quantifiable form, for instance the level of ambient sound in a

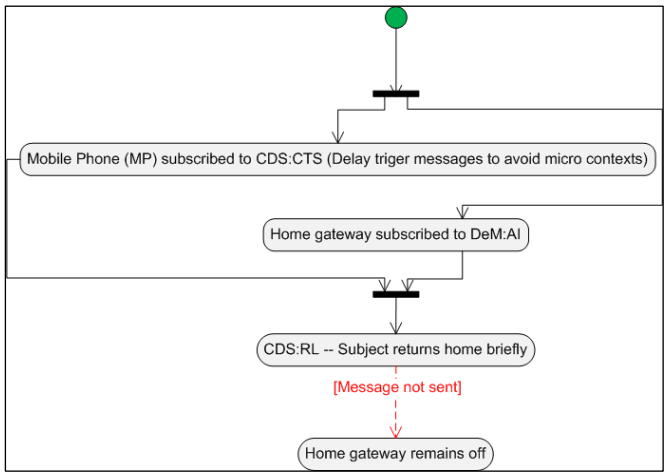

Figure 2. Missed Trigger Interaction occurs when the Context Triggering rules delay the activation of a home gateway.

room. A fluent can hold at a given point in time, if it was previously initiated by an action and has not been subsequently terminated. Actions occur at points in time and can modify fluents. Time points provide a narrative based structure independent of any particular action.

We identified device control and knowledge management service feature rules from a literature review of state of the art ambulatory assessment systems. We encoded each of these rules into Event Calculus based rules in Prolog. For instance, figure 1 shows the encoding of a *Data Storage Through Processing* rule that could be used by a node to handle incoming data by processing them, then storing the processed data. Such a rule might describe that the recipient begins in a state listening for a connection. When a triggering message arrives, data is streamed and collected. The data are processed using appropriate algorithms once the streaming has completed. The processed data are stored and the device goes back to listening for more connections. If a connection is established the data is uploaded and the connection is terminated upon data transfer completion. The recipient should then return to a state where it can repeat the process for new inbound data.

B. *Conflict Detection*

A networked environment with a dynamic collection of sensing and processing nodes that attempt to detect for unusual subject behaviour could be a recipe for network device conflicts. For instance, features operating within and across devices could rely on synchronisation and concurrency patterns that may not actually arise owing to interactions between the devices and the rest of the network. Device conflicts reduce the levels of certainty that we can have in the care assessment data and conclusions to actuate based on them. We used the Event Calculus to look for conflicts by analysing the device rules.

This work is inspired by research on the feature interaction problem as there are many similarities between this problem and rule conflict. A classic telephony example from the feature interaction literature involves the user Alice subscribed to the feature Originating Call Screening (OCS), screening out calls to the user Charlie. The user Bob is subscribed to the feature Call Forwarding when Busy (CFB), forwarding calls to Charlie when busy. A conflict can occur if Alice calls Bob when he is busy, because either the call from Alice would be forwarded to Charlie, thereby invalidating OCS, or else the call would be blocked, thereby invalidating CFB. In either case, the operation of one of the two features would be invalidated by the presence of the other.

Searching for such conflicts in Event Calculus forms of the narratives and service specifications can lead to the discovery of conflicts amongst them. In order to detect conflicts between rules, we developed an analytical rule system that can be used to understand what happens when multiple feature rules are triggered. The system analyses rule execution sequences to determine whether the rules lead to conflict. The framework ignores the contents of the triggering messages, the actions that arise from being triggered and the semantic meanings of the features. Prolog programs based on the framework resolve goals by loading the feature rules and then proceeding to check for interactions between every possible pair of features (including checking features against themselves).

Checking a pair of features involves two phases: initialisation and detection. The initialisation phase resets the Prolog environment by removing all assertions from it. It then adds a number of time points (establishing a linear order amongst them) and initialises a message fluent that can be sent to the features. The detection phase involves passing feature rules, time points and messages to conflict detection rules. The conflict detection rules are then used to evaluate whether the feature rules are concordant or conflict, and to record evaluation results.

For this work we studied feature rules looking for instances of different types of conflict: Shared Trigger Interaction (STI), Sequential Action Interaction (SAI), Sequential Action Interaction (SAI), and Missed Trigger Interaction (MTI). STIs occur when the antecedents of multiple features are satisfied such that they each perform actions in response to the same triggering event, and the operation of one or more of the features is different from how it would have reacted had it been the sole responder. SAIs occur when the operation of a feature is triggered in response to the actions of another feature. LIs are special cases of SAI whereby the operation of the chained features leads to redundant cycles. MTIs arise when the operation of a feature prevents the triggering of the operation of another one. The second feature may get stuck awaiting its trigger which is delayed, thereby causing the feature to operate incorrectly or not at all.

To contextualise sensor networks for behaviour monitoring, we considered case studies based on a scenario involving researchers interested in studying the impact of Bipolar Disorder on subjects conducting their usual activities of daily living. Bipolar disorder is a severe psychiatric disorder characterised by patients being in patterned (possibly cyclic and/or recursive) affective states, including mania, hypomania, eurythmia, depression and mixed states.

The scenario case studies depict how features could conflict by a particular conflict type. For example, an example MTI case study is shown in figure 2. It shows rules for a case study in which a mobile phone is subscribed to a feature rule that delays the transmission of a message that would activate a

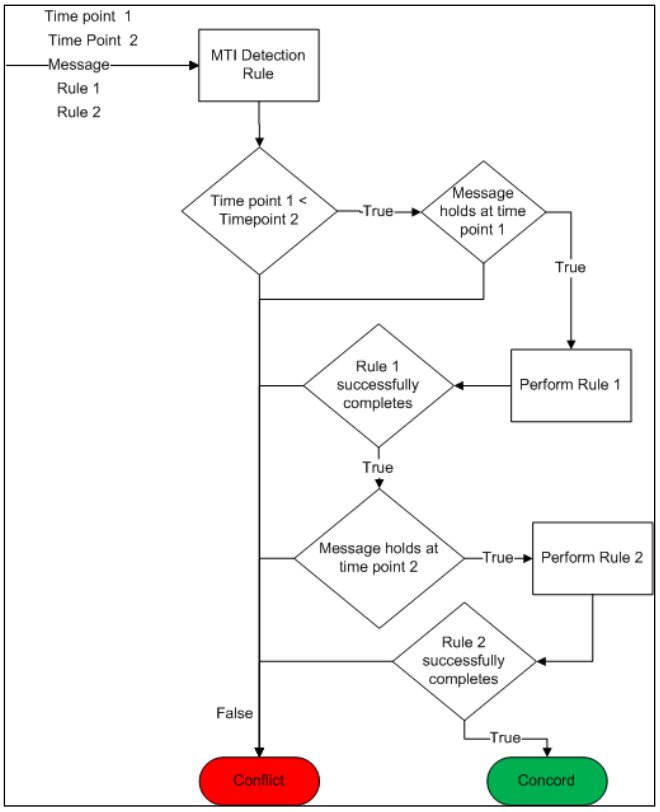

Figure 3. MTI conflict detection rule.

home monitoring system. Such delays may be reasonable from a phone programmer's point of view to minimise bandwidth usage and maximise battery life. If the subject travels from home, returns home briefly, then sets off again the home monitor would remains off because a trigger to turn it on would not be sent by the phone. This would lead to not capturing any abnormal behaviour about the brief return. Section 4 shows results of five case studies (one for each interaction type).

C. *Conflict Detection Rules*

Each of the conflict types were encoded as detection algorithms. These detection rules can be loaded into the analysis engine to check different rules for conflict.

Detecting MTI can be accomplished by testing features sequentially to ensure that a common fluent holds before being passed to each of the tested features. The fluent can be considered as a type of triggering message that should remain in a consistent state between features. Such an approach need not make any assumptions about the contents of the message, nor about the actions that should be performed by the features, nor also about what the rules do upon receiving a message. The algorithm for detecting MTI is shown in figure 3. The analytical framework evaluates a MTI concordance rule with arguments that consist of a pair of feature description rules, time points for the start times of each of the features, and the message fluent. The fluent initially holds prior to being passed to the first feature. Features conflict if the fluent becomes clipped prior to the execution of a feature.

STI detection, shown in figure 4, begins by loading arguments that consist of a pair of feature description rules, but

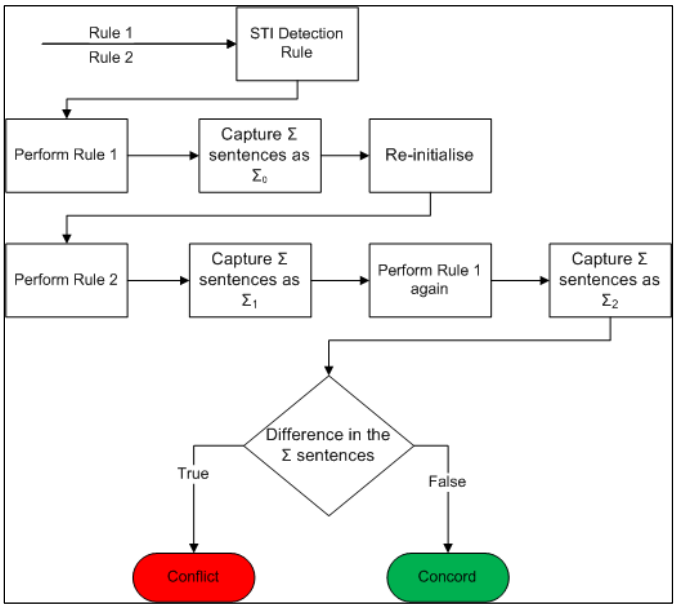

Figure 4. STI conflict detection algorithm.

ignores the time points and the message fluent arguments. The fluent initially holds prior to being passed to the first feature and the second feature. Features conflict if a check of the initiated actions from the first instance of the first rule does not match the second instance's initiated actions.

SAI can be detected by testing to determine if a feature rule performs an action that leads to actions being performed by a second rule. This can be accomplished by running rules sequentially within the framework and checking for α sentences that describe actions that will be performed as a result of the firing of the two rules. The analytical framework

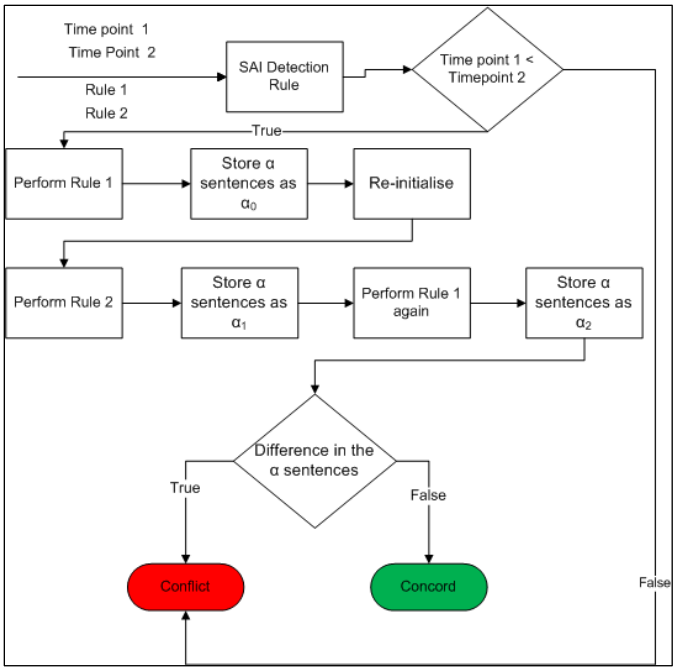

Figure 5. SAI conflict detection algorithm.

TABLE I. MTI CASE STUDY DETECTION RESULTS

Feature 1	Feature 2	Result
Notification suppression	Notification suppression	MTI
Notification suppression	Response prompting	MTI
Response prompting	Notification suppression	MTI
Response prompting	Response prompting	MTI

performs the procedures shown in figure 4. It uses the standard initialisation phase and then loads the SAI detection rule. This begins by ensuring the correct ordering of the time points. It then performs the first feature rule, stores its α sentences, and then re-initialises the world. Then it performs the second feature rule and stores its α sentences whereupon it re-runs the first rule and subtracts the second rule's actions from its actions. The remaining α sentences are compared with the actions from the initial run of the first rule. If they are the same then the second rule results in no additional actions, therefore the rules concord; otherwise they conflict by SAI.

LI occurs when one rule triggers another which in turn causes the first one to be re-triggered. LI, therefore, is a special case of SAI that can be defined as SAI leading to the triggering of the first rule's actions. This can be detected by performing SAI checks on the features and examining the output for cases where two features have SAI regardless of whether they are the first or second rule.

IV. EXPERIMENTAL RESULTS

Case studies were used to validate the detection algorithms against device rules. Here, we describe the results from one case study per conflict type.

A. MTI Case study

This is an MTI example of potentially conflicting rules. The rules describe features for subject response prompting and notification suppression (which might be enabled if a subject were in a meeting for instance). These rules each receive variables for the triggering message fluent and the time points for when they are respectively executed. Conclusions and Future Work

The results of analysing these features using the analysis engine for MTI are shown in table 1. The table shows that if the notification suppression feature is used prior to a second usage of the feature or the use of the response prompting feature then MTI occurs. In addition two instances of the response prompting feature will conflict if they are used together, as will response prompting when it is called before notification suppression.

B. STI Case study

This case study is characterised by the use of features from different services that run on the same device. In this case a mobile phone has features for Context Trigger to determine what activity the subject is engaged in when leaving home.

TABLE II. STI CASE STUDY DETECTION RESULTS

Feature 1	Feature 2	Result
Context Trigger	Context Trigger	STI
Context Trigger	State Trigger A	STI
Context Trigger	State Trigger B	Concordance
State Trigger A	Context Triggering	STI
State Trigger A	State Trigger A	STI
State Trigger A	State Trigger B	Concordance
State Trigger B	Context Trigger	Concordance
State Trigger B	State Trigger A	Concordance
State Trigger B	State Trigger B	Concordance

Similarly, rules have been set up using State Trigger to determine the emotional state that the subject is in when the subject's behavioural state changes from sitting to walking. An interaction can occur when the subject walks away from home. Here, both features may be triggered, however only one can elicit a response at a time, therefore a conflict occurs and the other feature will not be fulfilled.

Table 2 shows the analysis results for this case study. It shows the results of the use of two different versions of the state trigger rule. Normally, only one form of a feature would be used; however, in this case it is interesting to see what would happen depending on the form used. Form A responds to changes upon receiving state information in a similar manner as the Context Trigger rule. They both trigger an action in response to being triggered. Form B of the state triggering rule however is inert.

C. SAI Case Study

The case study features the rules Data Transfer and Redirect Data Stream. Data Transfer causes an action to occur, but Redirect Data Stream initiates a fluent change in response to the action. The fluent change in turn leads to the performance of another action. It is this connection which characterises a sequential interaction.

The results of testing the feature rules in the analytical framework using the SAI detection rule are shown in table 3.

TABLE III. SAI CASE STUDY DETECTION RESULTS

Feature 1	Feature 2	Result	Looping Case
Data Transfer|	Data Transfer|	SAI	Yes
Data Transfer|	Data Redirect	SAI	Yes
Data Redirect	Data Transfer|	SAI	Yes
Data Redirect	Data Redirect	Concordance	No

SAI was detected when Data Redirect was the first feature rule and Data Transfer was the second. This situation resulted in Data Transfer preceding Data Redirect and thereby initiating an action that leads to the initiation of an action within Data Redirect. SAI also resulted when Data Transfer was the first feature followed by either another Data Transfer or Data Redirect.

D. LI Case study

This case study considered a case where Data Transfer and Data Redirect are used on two devices, each directing the stream to the other. The results are also presented in table 3. A check of Data Transfer as both the first and second features resulted in LI in this case.

V. CONCLUSIONS AND FUTURE WORK

A vision of telecare is one of continuous collection, storage, analysis and reaction to multimodal data streams from a variety of sources. These processes will automatic, adaptive and personalised. The fusion of objectively measured and subjectively reported data on physical activity, location, interactions with others, psychological state and context will provide a wealth of knowledge from which we can tailor appropriate care.

With adaptation, however, comes the concern of minimising device conflicts. Like other complex adaptive systems such as call control systems, interactions can emerge that degrade the integrity of the network. These must be guarded against in future telecare systems.

We have begun this process by developing and testing algorithms that can be used to detect conflicts between telecare service features. In this paper we have reported the detection of four kinds of conflict that emerged in distributed telecare service rule descriptions. Situations that involve the distributed operation of a variety of devices are likely to occur. It is important to guard against conflicts within them to ensure the reliability of data collection and analysis procedures that will underpin the delivery of care.

The next step in this work will be a more thorough exploration of our initial results employing a broader range of scenarios and exercises. Also, the technique has a strong potential to be embedded more effectively within a future version of the PAM network architecture to provide a more responsive coverage. This will allow personalised telecare networks to self-heal when conflicts are detected and resolve problems in such a way as to maximise the integrity of the data. Resolving conflicts in real-time will be an important aspect of such work. Future technical trials of embedded conflict detection will allow us to collect real-world data which could be compared with trial data from our system with conflict detection switched off. These steps are being actively pursued.

ACKNOWLEDGMENT

This work was carried with the support of the EPSRC funded project EP/F003684/1 (Enabling health, independence and wellbeing for psychiatric patients through Personalised Ambient Monitoring (PAM)). The PAM project is a collaborative project between the Universities of Stirling, Nottingham and Southampton.

REFERENCES

[1] S. Intille et al., "Tools for studying behavior and technology in natural settings", Proc. of UbiComp 2003: Ubiquitous Computing, pp. 157-174, 2003

[2] J. Blum and E. Magill, “The Design and Evaluation of Personalised Ambient Mental Health Monitors,” Proc. Of IEEE CCNC, 2010.

[3] U.W. Ebner-Priemer and T.J. Trull, "Ambulatory assessment - an innovative and promising approach for clinical psychology", European Psychologist, vol. 14, pp. 109–119, 2009.

[4] J.B.J. Bussmann, U.W. Ebner-Priemer, and J. Fahrenberg, "Ambulatory behavior monitoring:Progress in measurement of activity, posture, and specific motion patterns in daily life", European Psychologist, vol. 14(2), pp.142–152, 2009.

[5] M. Zoumboulakis, G. Roussos, and Poulovassilis, A., “Active rules for sensor databases,” ACM International Conference Proceeding Series, vol. 72, pp. 98-103, 2004.

[6] K. Terfloth, G. Wittenburg, and J. Schiller, “FACTS - A Rule-Based Middleware Architecture for Wireless Sensor Networks,” Proceedings of the First International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE ’06), 2006.

[7] S. Sen, and R. Cardell-Oliver, “A rule-based language for programming wireless sensor actuator networks using frequence and communication,” Proceedings of Third Workshop on Embedded Networked Sensors (EMNETS), 2006.

[8] X. Fei, and E. Magill, “Rule Execution and Event Distribution Middleware for PROSEN WSN,” Second International Conference on Sensor Technologies and Applications, SENSORCOMM’08, pp. 529–551, 2008.

[9] G.A. Campbell and K.J. Turner, "Goals and Policies for Sensor Network Management", Proc. 2nd Int. Conf. on Sensor Technologies and Applications, pp. 354-359, 2008.

[10] M. Calder, M. Kolberg, E.H. Magill, and S. Reiff-Marganiec, "Feature interaction: a critical review and considered forecast", Computer Networks, vol. 41(1), pp. 115–141, 2003.

[11] R Kowalski and M. Sergot, "A logic-based calculus of events", New generation computing, vol. 4, no. 1, pp. 67–95, 1986.