On the Feasibility of Integrating Data Mining Algorithms into Self Adaptive Systems for Context Awareness and Requirements Evolution

by

Angela Rook
B.A., University of Victoria, 2005

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

On the Feasibility of Integrating Data Mining Algorithms into Self Adaptive Systems for Context Awareness and Requirements Evolution

by

Angela Rook
B.A., University of Victoria, 2005

ABSTRACT

Context is important to today's mobile and ubiquitous systems as operational requirements are only valid under certain context conditions. Detecting context and adapting automatically to that context is a key feature of many of these systems. However, when the operational context associated with a particular requirement changes drastically in a way that designers could not have anticipated, many systems are unable to effectively adapt their operating parameters to continue meeting user needs. Automatically detecting and implementing this system context evolution is highly desirable because it allows for increased uncertainty to be built into the system at design time in order to efficiently and effectively cope with these kinds of drastic changes. This thesis is an empirical investigation and discussion towards integrating data mining algorithms into self-adaptive systems to analyze and define new context relevant to specific system requirements when current system context parameters are no longer sufficient.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank the following people and organizations for their contributions to the research undertaken in this thesis:

Dr. Daniela Damian, my supervisor, for supporting me and challenging me in my pursuit of becoming a better researcher.

Dr. Alex Thomo, my graduate committee member, for his insights into Data Mining and his encouragement in exploring the research area.

OAR Northwest and the members who participated in this study including Patrick Fleming, Jordan Hanssen, Adam Kreek, Markus Pukonen, Greg Spooner, and Richard Tarbill for all their valuable contributions including extensive feedback at multiple points in the study and data coordination.

Alessia Knauss, Eirini Kalliamvakou, Dr. Eric Knauss, and Jordan Ell for their insight, guidance, constructive feedback, and friendship.

Dr. Hausi Müller for his invaluable insight and expertise in Self-Adaptive Systems.

Dr. Fritz Stahr from the University of Washington School of Oceaography for providing the environmental context data sets used and his invaluable insight in preprocessing them.

Fatigue Science for providing the Readiband biometric data sets.

Dr. Florin Diacu, my external examination committee member.

NSERC for funding this research.

DEDICATION

To Rusty, Brenna, and Gabriel.
There are no words for how grateful I am for you.

Chapter 1

Introduction

1.1 Motivation of this Research

A common problem for mobile app developers is that the context of use of the application cannot always be anticipated at design time, and therefore, an incomplete set of user requirements is a result. It is challenging to cost-effectively maintain system relevance through manually updating and evolving system requirements. Additionally, even if all contexts of use can be anticipated at design time, user requirements and their associated contexts of use are constantly evolving at runtime.

For example, while the designers of a mobile phone may make observations about the tasks being completed in an urban setting, it may not be feasible to make the same kinds of observations if the user(s) of the mobile system complete tasks in settings that are unobservable by the designers (e.g., prohibitively dangerous or remote like on the open ocean in a small craft or during a forest fire). Therefore, it may be extremely difficult to ensure that the situations a context-aware application should and should not be active/triggered in are properly defined at design time, particularly if their needs for a context-aware system function or service are significantly different from those anticipated in the urban context. In addition, if a forest-fire fighter expects that her phone should behave in one way while fighting fires (e.g., send all incoming calls to voicemail, keep map of current location and other fire fighters on screen), and then automatically changes functionality when back in an urban setting (e.g., vibrate phone whenever a new call is received, disable GPS location services for privacy). This is only compounded by the fact that the user may wish for her system to behave differently between different urban settings. For example, the system should enable the GPS location services when she's in an unfamiliar urban setting that is near to where she's fighting fires, but disable the GPS location services when she's back in her hometown.

In order to cope with evolving user requirements, *context-aware* systems need to automatically identify evolving contexts of use relevant to specific user requirements at runtime and adapt accordingly in order to reduce operational and maintenance costs. This turns them into context-aware *self-adaptive* systems.

Data mining refers to the process of applying machine learning algorithms to large data sets in order to discern patterns within the data. By using data mining algorithms applied to historical sensor data concerning the context of use collected passively at run time, we can discern patterns for when a service should be delivered to a user by a context-aware system. This means that system requirements can evolve in order to continue effectively meeting user needs. Because these contexts may be subtle and expensive to characterize manually, integrating data mining algorithms into the system in order to derive contexts that trigger requirements from collected data observations is a promising solution approach.

1.2 Research Objectives

This research aims at investigating the feasibility of integrating data mining into context-aware systems in order to facilitate automatic requirements evolution at runtime. These mechanisms will enable developers to shift much of the uncertainty about operational context for specific requirements at design time to runtime, and will allow for a more automated approach to system evolution and maintenance with fewer assumptions.

Specifically, this research focuses on using data mining algorithms to dynamically detect patterns in historical contextual data, and correlate those patterns to specific system requirements, thus producing a self-adaptive system evolution. Through this research, the following objectives are pursued:

- Data mining algorithms are applied to historical sensor data to automatically identify which context situations are relevant to a specific requirement.

- The results of applying the data mining algorithms to historical sensor data are compared against the actual context situations in which specific requirements are valid.

1.3 Research Methodology

The methodology presented in this thesis is a pragmatic approach to contextualizing requirements for an unobservable setting in order to ensure the developed system functions properly and meets user needs. It is based on passively collected sensor data from two field studies, and is supported by user log data from one of the studies and multiple interviews with the users. An iterative process of literature reviews and case studies combined with the approach to data mining presented by Kotsiantis [17] as well as empirical analysis produced the results in this thesis.

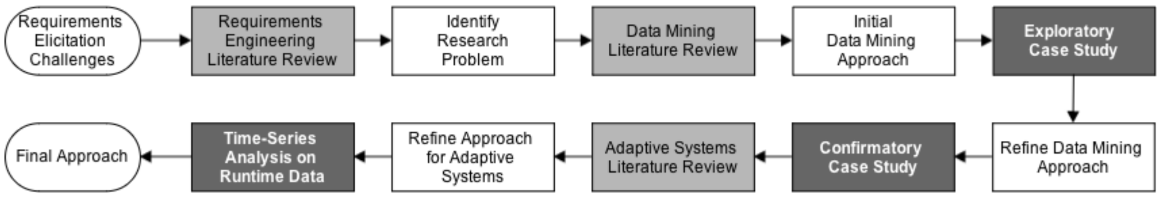
Requirements Elicitation Challenges
Requirements Engineering Literature Review
Identify Research Problem
Data Mining Literature Review
Initial Data Mining Approach
Exploratory Case Study
Refine Data Mining Approach
Confirmatory Case Study
Adaptive Systems Literature Review
Refine Approach for Adaptive Systems
Time-Series Analysis on Runtime Data
Final Approach

Figure 1.1: Research Methodology.

The realism of the case studies is high because the results are based on passive, unobtrusive data collection from two actual operating environments. An attempt to maximize precision in this study has been made through the similarity of the case studies (minimizing impacts to internal validity), maximizing the number of sensors involved, and the isolation of the operational environment (minimizing external impacts to the system). While the operational environment of the users in the case studies is relatively unique and isolated, the results reveal that the most important contextual attributes to the system involved are similar to those found in many mobile devices (location, time, user identity, motion detection). Given this, the results from this thesis may be generalized to those cases.

1.3.1 OAR Northwest

OAR Northwest is a Seattle, USA based non-profit organization who undertake long-distance rowing voyages in order to perform research and deliver science, technology, engineering, and math (STEM) curriculum to classrooms online and through school visits. The data from the two case studies presented in this thesis was collected from two open-ocean voyages, completed by OAR Northwest in the space of a year. These voyages both took place in a custom rowboat designed for long-distance, open-water journeys, and were propelled entirely by the rowers themselves.

In September 2012, OAR Northwest was collaborated with in order to develop a context-aware activity scheduling (calendar) system (ToTEM) for an open-ocean voyage from Dakar, Senegal to Miami, Florida, USA. Because of the highly isolated and dangerous nature of the system context of use, it was impossible for software developers to observe the rowers interacting with the system in order to perform typical requirements engineering activities such as ethnography. In addition, it was not possible for developers to meet with a number of the rowers in person during design time, and developers had to rely on interviews conducted through video telecommunication means such as Skype. This created a number of requirements engineering challenges detailed in Chapter 4, and prompted the exploration of how the passively collected sensor data that OAR Northwest collected during their voyages may be investigated for additional insight for the requirements engineering process.

The need for context awareness within the system developed was justified by the rowers for two reasons. The first of these was because of the extreme and dangerous conditions they often faced on the open ocean. The rowers expressed that, ideally, the system should be aware of these conditions and adapt accordingly in a non-obtrusive way. Additionally, the rowers often faced extreme fatigue and wished to use the system for cognitive offloading such that the system would 'think' for them in a number of circumstances so as to better support them in achieving their research and voyage-completion goals. Context awareness was seen as a way to support cognitive offloading in this manner.

While there is much previous work in literature on location-dependant, ubiquitous mobile context-aware systems, there is a lack of literature on those that do not depend on location or constant connectivity in order to offer context-aware services to users. In addition, there is existing work on data mining user data offline (e.g., uploading user data to servers for data mining), but there is little on concrete applications that might necessitate online data mining of user data (i.e., those that data mine directly on the mobile device).

This study investigates a non-location dependant context-aware mobile system, as well as one that is non-ubiquitous, therefore, necessitating the implementation of data mining directly on the mobile device itself. This has implications for context-aware mobile applications where privacy may be a concern (i.e., ones where the user may want to keep sensor data localized for data mining processing instead of uploading it, as in the case of health-oriented applications), as well as for systems that may not have consistent connectivity (e.g., disaster zones, or extremely isolated environments). In addition, it has implications for group context-aware mobile applications.

Two separate case studies were carried out, an exploratory case study, and a confirmatory case study.

1.3.2 Exploratory Case Study

The purpose of the exploratory case study was to discern whether or not the potential of data mining to automatically define context for contextual requirements for system evolution was worth investigating further. An initial data mining approach[1], was developed for the study and applied to the exploratory case study data set for the following purposes:

1. to investigate whether or not a requirement could be linked to contexts of use by discerning patterns in which of the sensors and specific contextual situations (represented by the readings of those sensors) were relevant by applying data mining algorithms on the given data set

2. to see which, if any, of the data mining algorithms used performed to accuracy levels greater than 80%

3. to discover which data mining algorithms produced the most accurate results.

Upon obtaining results for the exploratory case study[2], the data mining approach was refined and a confirmatory case study was undertaken in order to confirm the results.

1.3.3 Confirmatory Case Study

The refined data mining approach derived from the exploratory case study was applied to the confirmatory case study with a much larger runtime sensor data set with more contextual requirements investigated. Upon obtaining results that confirmed those of the exploratory case study[3], the potential for automatic contextual evolution for requirements at runtime was explored and an adaptive systems literature review was undertaken. This was done in order to explore where the context definition at runtime

[1]This initial methodology is based on Kotsiantis' approach, illustrated in Figure 3.1.
[2]Shown in Section 5.1.6.
[3]These results are discussed in Section 5.2.5).

for contextual requirements using data mining algorithms could fit into self-adaptive systems for runtime requirements evolution.

1.3.4 Time-Series Analysis on Runtime Data

In order to further explore the feasibility of automatic requirements evolution using data mining algorithms on contextual requirements, a time-series analysis on the historical runtime sensor data from the confirmatory case study was completed for the JRip data mining algorithm and the J48 data mining algorithm. This analysis was used to explore the following:

1. how long it took for each of the data mining algorithms to accurately predict the relevant context of use for each contextual requirement from the confirmatory case study
2. how sensor configuration changes affect the data mining algorithms' ability to accurately define the context of use for each of the requirements from the confirmatory case study.[4]

1.4 Contributions

The research carried out in this thesis makes three unique contributions in its approach to requirements engineering using data mining algorithms applied to system requirements contextualization for unobservable environments:

1. An exploration of the feasibility of integrating data mining algorithms into self adaptive systems for context awareness and requirements evolution.
2. A novel application of data mining in order to identify context of use situations for several requirements using passively collected historical sensor data in order to implement them as context-aware services in the ToTEM context-aware mobile application.
3. A time-series analysis and comparison of the performance of JRip (RIPPER) and J48 (C4.5) data mining algorithms on identifying context of use situations from runtime sensor data for several requirements.

[4]The results of this time-series analysis on runtime data can be found in Section 5.3.4.

This research has significant implications for the state of the art of data mining and self-adaptive systems, as well as requirements engineering:

1. Context of use situations for requirements (application services) for context-aware applications can be derived from passively collected sensor data, thus moving such requirements elicitation from design time to runtime,

2. Context of use situations for context-aware services can be derived for unobservable contexts of use from passively collected sensor data,

3. It is possible to derive some context of use situations from passively collected sensor data within finite time ranges to high performance levels, and

4. the context of use classifiers produced by both the JRip and J48 algorithms are robust enough in several cases to continue providing high levels of accuracy, precision, and recall even with sensor configuration changes and abrupt changes in normal user behaviour (such as user activity shift changes).

1.5 Thesis Outline

Chapter 1 contains a statement of the research area of interest as well as the investigative approach undertaken in this thesis followed by an overview of the structure of the document itself.

Chapter 2 provides a background of the practical problem explored combined with an overview of the current state of the art and the impact of the research.

Chapter 3 details the methodological approach undertaken in order to solve the research problem.

Chapter 4 gives a methodological overview of the empirical analysis undertaken to arrive at the final approach.

Chapter 5 includes the data characteristics of the exploratory and confirmatory case studies and the results obtained from the empirical analysis. It also includes details of the time-step analysis of the runtime data from the confirmatory case study and the results obtained.

Chapter 6 discusses the final approach and the results obtained with respect to the original research problem, and demonstrates how the outcomes can be implemented using existing adaptive system models.

Chapter 7 summarizes the problem addressed in this thesis and the approach taken to solve it.

Chapter 2

Literature Review

The definition of *context* as it relates to software engineering and computer science has gone through several revisions. It has been defined as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves," with primary context types that can be used to situate an entity being *location*, *identity*, *time*, and *behaviour* [7]. A more recent definition further refines context into the following three categories:

1. *computing context* refers to the actual hardware (and its configuration) that the user(s) have access to for interacting with the system such as input/output devices, memory capacity, and wireless bandwidth,

2. *user context*, which pertains to the users of the system and the individual application context associated with them such as their system profiles, calendars, and preferences, and

3. *physical context* which is the non-computing, real-world setting that the system is interacted with by the user(s) in. Examples include location, time, weather conditions, noise, directional heading, and wave pitch. [3, 14]

There may be contextual information relevant to the system above and beyond that which the system can detect; however, it is only that which the system can process that can be used in practice [9]. While this may be relatively straightforward for computing context, it may be more difficult to capture relevant user context, and physical context. However, the recent surge in the availability and application of sensors has made gathering and reifying user context and physical context much more accessible for system integration. Mobile devices now include location, environment, and motion sensors, and bluetooth enables mobile systems to incorporate additional sensors such as those for health monitoring. A system that is able to adapt itself to a changing context is called a *context-aware* system. A system is considered context-aware if it can process and apply context information to adapt its functionality to the current context of use [9, 14].

The majority of prior work on context awareness using sensor data has focused on location physical context and ubiquitous computing, for example map and direction applications, and location-sensitive applications as in the case of [2, 16]. However, the applicability of context-aware systems to unobservable settings, with no connectivity, and where location data is not relevant to the context-aware services being provided is underrepresented in literature.

2.1 Context-Aware Applications for Mobile Systems

Context-aware applications can reduce the amount of effort a user has to put into interacting with an application, and can automatically deliver desired services [13]. For example, if an application can sense the current context, then it can efficiently adapt automatically without the user having to take any action. Additionally, sensor technology calibrated to a specific user's profile can enable them to leverage that context for a wide range of services [19]. There is currently great interest in context-aware mobile applications in health-related fields where users are empowered to make health-conscious decisions based upon the activity-related sensor input the context-aware system receives and interprets for them [19, 23, 28, 25].

The UbiFit Garden project is an example of a health-oriented context-aware application that displays continuous, ambient updates about the activity levels of a user so that they know how much physical exercise they have completed during the day [19]. There are currently many sophisticated health-monitoring bluetooth-enabled wristband sensors designed to monitor activity in particular. This is useful for monitoring sleep and fatigue levels [28], and exercise and activity levels [19, 25]. More commercially-available sensors are currently in development with a wide range of signals in addition to motion sensors including pulse, temperature, and blood oxygen level. Sensors like these would be capable of capturing information for context-aware services such as:

- workout tracking and sleep monitoring,
- facilitation of monitoring of circadian rhythms such as fertility cycles,
- childcare health applications, such as fever or asthma monitoring, heart monitoring, and
- historical data gathering to be shared with healthcare professionals for interpretation and informed treatment recommendations.

However, risks come associated with the collection of personal context sensor data. A major issue with ubiquitous systems is privacy, and while there are a number of context-aware applications and studies that focus on leveraging location contextual data [20, 16], there are inherent risks associated with the leaking of private contexual information (e.g. location) [19, 8, 16, 18, 15]. Certainly these same risks apply to the collection and transmission of health-related contextual information. Given these concerns, it may be preferable to keep contextual information as localized as possible and to process and delete it as soon as is feasible. The resource cost of connectivity is another motivator for keeping all context-aware processes directly on the mobile device instead of uploading contextual data for processing elsewhere [11].

While there have been a number of empirical studies using accelerometry sensor data for recognizing activity, these have been largely focused on laboratory settings, and over a period of time that usually spans under a day [25]. Longer-term studies involving users completing activities outside the lab appear to be less prevalent. Additonally, the studies do not appear to take into consideration sensor data from other physical context, such as weather conditions, for example.

2.2 Data Mining and Context-Aware Mobile Systems

One of the ways that context awareness is supported is by integrating *data mining classifiers*. *Data mining* is a form of machine learning that generally uses historical data to form statistical predictions about future context [17, 30]. That is, data mining classifiers are used to identify real-life *situations* such as “at home”, “running”, or "imminent danger" [11]. Data mining algorithms are often applied when human analysis is not feasible (e.g., very large amounts of data), and also to discern subtle and non-obvious patterns in data.

A data mining *classifier* is derived by applying a data mining algorithm (such as the RIPPER [34, 5] or C4.5 [26] algorithms) to a *training set* of representative data. The resulting classifier is then applied to new data in order to classify it. In the case of context awareness, data mining classifier inputs include context sensor readings (singly or aggregated) as well as other contextual data, and outputs include a classification (often represented as a binary value) that the context-aware system interprets as a service to be delivered [2, 6, 8].

The classification depends on a *target attribute* (also called an *indicator attribute*) included with the *instances* (e.g. rows in a table) of the training set that indicates the classification of each of the instances. In this thesis, all the target attributes are binary values, and the binary classification pertains to whether or not the context aware application should or should not deliver a particular service. The data mining algorithms use the target attribute and sensor data to derive a classifier by identifying patterns from the correlations between the target attribute and the training data. The method of deriving a data mining classifier (including testing and evaluation) will be discussed in Chapter 3.

Much work has been undertaken to produce classifiers by data mining historical context data for context-aware applications implemented on mobile systems [25]. This includes deriving and performing empirical comparative analyses on classifiers from different movement patterns from sensors (e.g. sleeping, walking, running) in order to keep track of daily activities for health monitoring applications [19, 25, 1]. There is also current work in mobile context awareness focusing on monitoring computing context in mobile devices in order to make better use of mobile system resources while data mining streaming context [11].

There is, however a need for further studies on real-life applications with comparative analysis performed between data mining algorithms in real-life settings [10]. Additionally challenges such as *concept drift* (or *data expiry*) [32] need to be explored for mobile systems. Concept drift is when data in the data mining algorithm training set is no longer relevant. In the case of context awareness, concept drift refers to when context data in the training set the data mining classifier was produced from the system no longer correctly correlates to when the service should and should not be delivered. For example, say the situation the health-monitoring context-aware application classifier was trained to recognize was rowing, but the user no longer rows, and bikes instead. The “behavioural envelope” [6] that the classfier was produced to recognize has now changed, and the classifier may no longer be valid. Thus, the context-aware application may no longer perform adequately.

2.3 Requirements Engineering for Mobile Context Adaptive Systems

Software *requirements* are defined as “a condition or capability needed by a user to solve a problem or achieve an objective [29].” Users want context-aware systems to anticipate and respond to their needs as unobtrusively and correctly as possible. In order to do this, a context-aware system must be able to detect changing situations, and correctly meet requirements associated with the current situation. Unfortunately, it is impossible for designers to anticipate all the contexts of use that a context-aware system in a dynamic environment will be operating in. This is especially true of ubiquitous and mobile systems where contexts of use may be constantly changing [32, 19, 16].

As such, context classifiers for context-aware applications on mobile devices need to evolve to continue to reflect the context situations they represent so that user requirements can continue to be met [32, 19, 16, 4, 7, 9, 31, 6]. This means that data mining algorithms should be applied to context training data when concept drift occurs to the point that the context classifier is no longer able to fulfill the user requirements. This raises the question as to how much training data is required before the data mining algorithms can adequately derive context classifiers. It also raises the question as to what context impacts the performance of the resulting classifier. An empirical, time-series analysis is an appropriate approach to address these.

Chapter 3

Research Approach

This chapter describes the research approach used in this study to address the problem of how system contexts of use (*context situations*) can be better understood to support the requirements engineering process. The approach uses passively collected historical sensor data to define context situations based on a correlation to the triggering of behavioural and functional requirements. The data mining approach applied in this study is described by Kotsiantis, illustrated below in Figure 3.1.

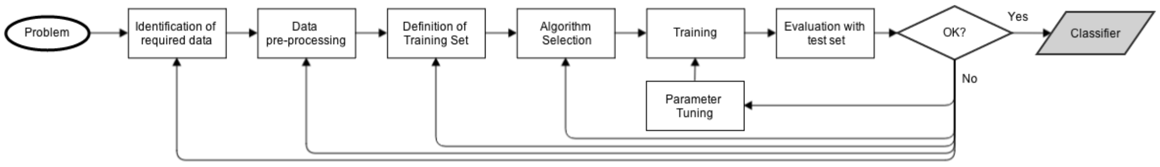

Figure 3.1: Kotsiantis' described approach to Data Mining [17].

With respect to the current work, the goal of interpretation and implementation by requirements engineers, and eventual automatic implementation on mobile systems is desired. The final outcome of this approach is a method to push defining when requirements should be triggered from design time to runtime.

3.1 Identification of Required Data

The first step in defining the triggering context for functional and behavioural requirements is to gather relevant runtime reified contextual data for analysis. This includes passively and actively collected quantitative sensor data. Passively collected data refers directly to information gathered directly from sensors, that do not require input from a user. Actively collected data refers to information gathered in conjunction with user input, such as when a rower records information in a log book. As well, any qualitative insights from the users that may serve to verify the correlation of the requirement to the context, can be useful. Passively collected reified contextual data, or actively collected reified contextual data that can be directly interpreted by the system is preferred because the eventual goal is fully automatic definition and evolution of the triggering context for requirements.

It can be difficult to identify sufficient relevant reified context for a given requirement. Depending on the requirement, reified data (sensor or otherwise) from one or more of the four context types defined by Dey, Abowd, and Salber [7] (time, location, user, behaviour) can be essential to effectively defining the triggering contexts using data mining algorithms. Additionally, rate of frequency of data readings from the data sources, and the amount and completeness of historical reified contextual data available can have huge impacts on deriving triggering contexts.

It may be possible for a human analyst to adequately identify the reified context data sources for a given requirement to apply data mining to. However, two of the goals of the approach taken in this study are towards full automation with as lightweight an implementation as possible. In order to support these goals, a brute-force method was employed, whereby all passively-collected, reified contextual sensor data available after the data pre-processing step (Section 3.2) was included in the data mining process for each of the requirements. This approach serves to minimize human involvement in identifying the triggering contexts, and to allow the data mining algorithms the opportunity to identify relevant triggering context that a human may not have considered relevant. In addition, certain attribute-selection algorithms may be employed to varying degrees, but again, in support of a lightweight implementation, this approach was not considered.

Once all available reified contextual data was identified and collected, data pre-processing took place in order to convert the data into a format that the data mining algorithms could be effectively applied to.

3.2 Data Pre-Processing

In data mining practice, data pre-processing takes approximately 80% of the total computational effort [35]. The relevant data pre-processing steps for this study are outlined below:

Data Integration

Data integration involves combining the disparate data sets into a format that data mining algorithms can be applied to. In this study, this consisted of merging the collected historical sensor data into a single matrix for each of the two case studies. This was accomplished primarily through aligning records together into a single matrix based on the Coordinated Universal Time (UTC) field available in each of the data sets.

Offset due to merging data on erroneous fields can introduce noise into the data set and lead to inaccurate results. As a refinement of the second case study, merged data sources were checked for offset by graphing and comparing other redundant attributes for correct alignment. This was particularly valuable where a power loss in one of the data collection devices introduced erroneous timestamps into the UTC field. Redundant time and latitude and longitude fields between disparate data sources allowed for minimizing record offset through visual inspection. Comparing periods of normative behaviour was also valuable for integrating biometric data from each of the users into the merged data source. This approach is discussed more fully in detail in Section 5.2.

The frequency of sensor readings was different between the disparate data sets. In the exploratory case study, the data set with more frequent readings was merged onto the UTC field of the data set (to the minute) with less frequent readings. This resulted in the data mining algorithms being applied to a smaller data set than originally available, but with few missing sensor readings. In the confirmatory case study, the data set with the most frequent readings was used to merge the sparser data sets onto. The UTC column of each of the data sets was used again as a key. The difference in the the reading rates produced missing values in the merged data set. While there were no duplicate entries in the data sets, the data was carefully inspected beforehand for duplicate entries to ensure that this method of merging would be successful.

Data Cleaning

Data cleaning attempts to remove and/or replace erroneous data from a data set for data mining in an effort to reduce noise and improve the resulting classifier. This primarily involved removing erroneous outliers, and replacing missing values with persistent ones from the merge approach used in the confirmatory case study (Section

5.2). Outliers were identified by deviations from normative sensor behaviour, and in some cases, able to be correlated to sensor technical failure.

Missing data due to changes in the rate of frequency of sensor readings was estimated by persisting current readings through to the next one (within time constraints). That is, if data was missing from a column in the merged data set due to a difference in frequency of readings in the disparate data sources, that missing entry would be filled by a previous sensor reading. This was under the condition that the time difference between the previous sensor reading and the sensor reading in question fell within a specific time limit. This was to ensure that relevant reified contextual data would not be excluded simply because of a difference in the volume of available data from different data sources.

The reasoning behind this 'persistent' approach was an effort to emulate a realistic implementation for data mining historical contextual sensor data on mobile devices by minimizing processing overhead. While missing data can be averaged between sequential sensor readings, the computational complexity of applying this approach is greater than a persistent approach. A persistent approach requires holding a sensor reading in memory and doing a simple compare for staleness until a new reading is obtained. Conversely, an averaged approach requires several additional computations.

Normative behaviour and deviations from normative behaviour captured by sensor readings was particularly valuable. Not only did it help to remove any offset in merging the disparate data sources, but it also allowed additional noise to be removed from the data set. In the exploratory case study (Section 5.1), normative user behaviour for the system was indicated by the consistency of sensor readings. This helped improve data mining classification results by removing some of the outlying circumstances that the users did not use the system in. In the confirmatory case study (Section 5.2), normative behaviour was used to help identify times when the users were operating the system under standard conditions. Once this was identified, deviations in normative behaviour could be inferred in the sensor data through visual inspection. System requirements could then be correlated to these deviations in normative behaviour, and in some cases, contexts for new requirements could be suggested to the users.

Data Transformation

In the exploratory case study (Section 5.1), normalization was applied to all sensor data in the merged matrix. The normalization applied to our data sets scaled the ranges of values in each column to between 0 and 1. Normalization in these cases also applies to nominal values (such as words) where they are assigned a numeric value in between 0 and 1 instead. Normalization is useful in some data mining algorithms to improve algorithm performance and classifier results. This is done by comparing the euclidian distance between values instead of simply the values themselves so that the resulting classifier is not disproportionately skewed to specific attributes (thus reducing noise).

Data Reduction

Data reduction involves reducing the data set to exclude irrelevant data and data that introduces noise into the system. It is a way to improve algorithm performance and the accuracy of the resulting classifier.

The merged data set from each of the exploratory and confirmatory case studies was reduced by removing columns full of sensor data that was empty, or identical within the same data source (e.g. certain time attributes). In the exploratory case study, additional columns of data considered not relevant to the context of interest were dropped. This included redundant attributes such as latitude and longitude data from sensors with less frequent sample rates than the ones left in the data set, and certain on-board instrumentation status.

In the confirmatory case study, however, only the empty and identical columns were removed. This was because a 'brute-force' method was used in our approach, and all available reified contextual sensor data was included in the data sets the data mining algorithms were applied to. Again, this 'brute-force' method was used in order to emulate lightweight application implementation for mobile systems with as little human involvement as possible in selecting the reified context of use data relevant to specific requirements.

Even given the 'brute force' method, several more columns of data were excluded from the historical sensor data from the confirmatory case study the mining algorithms were applied to. This was because those columns excluded were comprised primarily of unique values. Several columns of primarily unique values were uncovered in the exploratory case study. The classifiers produced were found to have to be over-fitted to these unique values because of the noise introduced into the analysis by the uniqueness of the location values. For example, certain time attributes (month, day) were fairly unique in both our case studies with very little repetition occurring through the duration of the case study. Additionally, the location values (latitude and longitude) are relatively unique because the users did not tend to revisit their exact locations for the durations of the studies. Conversely, if the users revisited the same location(s) several times in the study, the value of the location data may have been more useful (similar to the time and location values of the trips the BC Ferries make every day at regularly scheduled times).

3.3 Training and Test Data Sets

Data Characteristics

As described in the above Pre-Processing sections, many aspects of the data set used for training can have an impact on the accuracy of the classifier produced. These include a number of factors related to consistency of data collection, and removing as many errors in the data as possible. In addition, the amount of data available can impact the quality of the classifier. If an algorithm is not applied to a 'critical mass' of data, the accuracy of the resulting classifier produced may be too low to be useful. This 'critical mass' of data can vary from set to set, and it may be difficult to define exactly how much sensor data is needed for the training set and the test set.

Instead, it is better to consider that the training data set must be representative of the test set. For example, if user buying patterns over a year are of interest, and the data mining algorithm is applied to only a training set of data from January, February, and March, then the resulting classifier may not be accurate on test data from December. Additionally, as buying patterns change over time, it may be more useful to consider data from the last three years rather than including those up to five years ago. Similarly, it may take more than one instance of a requirement being active for the algorithm to produce a satisfactory classifier, and since our assumption is that system context changes over time, older data may need to be dropped from the training set so that the classifier produced reflects only the current context instead of being muddied by that which is no longer relevant.

The training set should include representative examples of sensor data from the context situation(s) that the requirement should be active in, and also the situations that the requirement is not active in so that the algorithm can effectively differentiate between the two. Therefore, a binary classification is assumed for the requirement. If the separation between these situations is clearly and consistently defined, less sensor data may be required to produce a classifier of adequate *accuracy*, *precision*, and *recall*. Accuracy refers to the total number of rows the classifier correctly classifies in a data set divided by the total number of rows in the data set. Precision for the active or inactive state is characterized by the number of instances correctly identified for the state divided by the sum of the correctly and incorrectly classified instances for that state. Recall refers to the number of rows a classifier correctly identifies as being in the active or inactive state, divided by the number of rows that actually are in that state. If the separation between these situations are not clearly defined and there is a lot of contradictory overlap in the training set between the context situations where the requirement should be active, and when it should not, then the accuracy, precision, and recall of the resulting classifier may not be adequate for automation.

Target Attribute

Defining what separates when the requirement is active/triggered and inactive/not triggered is the key to linking user requirements with reified context, illustrated in Figure 3.2.

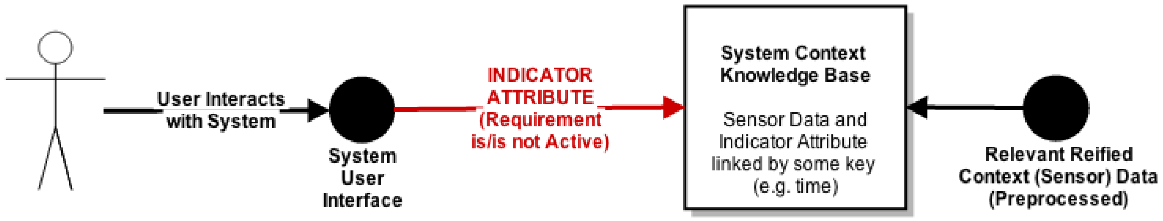
User Interacts with System
System User Interface
INDICATOR ATTRIBUTE (Requirement is/is not Active)
System Context Knowledge Base
Sensor Data and Indicator Attribute linked by some key (e.g. time)
Relevant Reified Context (Sensor) Data (Preprocessed)

Figure 3.2: Flow of information linking user behaviour to sensor data for data mining classifier training through the target attribute.

In the approach taken in this study, this mapping between user requirements and reified context has been implemented for data mining through the *target attribute*. Each requirement being automated has a target attribute associated with it. This target attribute is represented for data mining as an additional column with a binary label for each row in the table of sensor data indicating whether or not the requirement it represents was active/triggered (1 or ‘y’) or inactive/not triggered (0 or ‘n’) for that row.

It is assumed that the ability of a system to continually capture new user feedback about the target attribute for each requirement at runtime is necessary in order to ensure that it continues to meet performance standards. This user feedback is what allows the system to determine whether or not the context situation it had defined for a requirement is still valid or if it needs to evolve. While it may be effective to implement an interface where users can actively indicate when specific requirements should be triggered (for example, simple ‘on/off’ switches for when a requirement should be active/triggered or should be inactive/not triggered), this takes a lot of cognitive overhead, and may not be considered worth the effort of automation by users. Ideally, these switches between active and inactive requirement states would be automatically captured by the system passively in some way, for example, when changes occur in the system settings, or a combination of low-overhead user input and passive collection would occur. However, this passive or active collection by the system is not always possible, and the target attribute must be inferred in some other way. In this study, the target attribute was inferred in one of the following three ways for each requirement and validated with the users:

1. Through mathematical derivation based on some numerical threshold or function,
2. Through visual inspection of the sensor data in graphical form and correlating log data with anomalous patterns in the sensor readings, and
3. Through a combination of (1) and (2).

It should be reiterated at this point that the eventual goal of the approach taken is for the system to automatically predict when the requirement should be active through context awareness, so the target attribute on its own is not sufficient, as it is merely a record of when the requirement state (i.e., when the requirement is active/triggered or inactive/not triggered) in relation to the corresponding sensor data.

Similarly, relying on specific sensors alone for context awareness is not sufficient because possible sensor failure means that the requirement may go unfulfilled as long as that sensor is offline. The ultimate goal of the system is being able to adapt to changing contexts, including sensor configuration changes. Given this, it is important for the system to be able to cope with sensor loss and continue to be context-aware, so being able to draw context (when necessary) from a number of different sensors instead of just one or two is preferred.

3.4 Algorithm Selection

Algorithm selection for data mining is influenced by a number of factors. These include whether the data the algorithm is being applied to are discrete, continuous, or binary (or a combination of these), and the quality of the data itself, such as whether there are missing and/or erroneous entries in the data. Depending on the computing system resources available, other factors such as algorithm complexity, the accuracy of the resulting classifier, and the speed of producing the classifier may be important. Additionally, depending on how and for what purpose the classifier is going to be used, the transparency of the resulting classifier (i.e., how easily understood the logic behind it is), and the speed of classification on incoming sensor data may also be vital.

Algorithms for Requirements Engineering

Human comprehensibility of the classifiers produced was considered to be important to algorithm selection for three reasons. The first was that transparency of the logic behind the classifiers produced should be obvious so that the context situations relevant to specific requirements could be easily understood and verified with the users by requirements engineers. This included being able to easily discern the context attributes most prevalent to each context situation for each classifier.This was important to the early stages of the approach while it was being explored for feasibility.

The second reason for using classifiers with relatively high transparency was considered later in the study during our exploration of the automatic implementation of data mining algorithms for system context evolution. This concern focused on the transparency of identified context presented through an interface to users for validation and verification purposes. It was assumed that classifiers that were already relatively comprehensible would be easier for end-users to interpret and provide necessary feedback to ensure the classifiers continued to perform within acceptable performance levels.

The third concern, influenced directly by these first two, was that algorithms that required manual parameter tuning in order to produce more accurate results were considered to be more cognitively intensive for requirements engineers and end users to use. As a result, whether or not a classifier required parameter tuning was also taken into consideration.

Algorithms for Mobile

While cloud computing has allowed for data from mobile devices to be transmitted and centralized externally for data mining analysis, the resource allocation and processing capabilities of mobile devices are a concern when data mining is implemented directly on them. Given these considerations, lightweight data mining algorithms with low complexity were explored before those with high complexity. Additionally, data mining algorithms that could not cope with missing or erroneous data were excluded given that incoming sensor data from mobile devices can have both. Because incoming data from sensors can come from a variety of disparate sources, the form of that sensor data can also be quite disparate. Therefore, algorithms that could handle a variety of different data types including nominal, continuous, and binary were preferred over those that were negatively impacted by a particular data type. Transparency was considered important to mobile users for the same reasons given above.

Speed of classification was also very important because the system needs to be context aware at runtime. If the system takes to long to discern the requirement state it should be in based on the time it takes to apply the classifier to the incoming sensor data, the context may have already changed, and the requirement state determined by the classifier may no longer be relevant to the current context situation. This lag may be unacceptable to the user. Therefore, the quicker the classifier can determine whether or not the current context situation applies to a particular requirement state, the better.

Data Profile

The actual 'operational profile' between different data sets may be very different, so while one algorithm may perform very well on a data set, another may not, and vice versa [17]. In the exploratory case study, several types of data mining algorithms (chosen based on the considerations above) were applied to the data set with rule-learning and decision-tree algorithms providing the most accurate results, and the logistic regression and support-vector machine algorithms providing the least accurate results[1]. This finding was consistent with Kotsiantis' observation that the rule-learning and decision tree algorithms shared a similar operational profile [17], and as such, the rule-learning JRip algorithm, and the decision tree J48 algorithm were chosen for

[1]These details are explored in section 5.1.6.

extensive evaluation in the confirmatory case study.

3.4.1 JRip (RIPPER)

The JRip (RIPPER) rule-learner algorithm was chosen in this study for a number of reasons. When considering comprehensibility to requirements engineers and end users, the rules-based algorithms were considered to be among the more easily understood data mining algorithms [17]. Additionally, it has the ability to handle missing data and a variety of data types including discrete, binary, and continuous (to a certain degree). It only produces rules for the target class (i.e., the active/triggered requirement state in the cases in this study), making it a more lightweight algorithm than others. It is also very quick to classify, which is important for context awareness on mobile systems as described above. Aside from its relatively fast classification speed and the high transparency of its resulting classifiers, it is described as being a relatively average algorithm in most other respects including tolerance to irrelevant data, accuracy in general, and speed of learning [17]. This was considered a benefit to generalizability with the reasoning being that if good results could be obtained with a relatively average data mining algorithm, then more advanced algorithms may produce even better results.

The JRip algorithm[2] is the Weka implementation of the well-known *repeated incremental pruning to produce error reduction* (RIPPER) algorithm [5].

Algorithm Output

Output for this algorithm takes the form of a list of classification rules that cover the rows in the training set for each of the two classes. In this study, these classes take on the value ‘1’ for the active/triggered requirement state, and ‘0’ for the inactive/not triggered state. The rules can be interpreted as a series of *if...then* statements to predict whether or not the current context situation (represented by incoming realtime sensor data) requires the requirement to be active/triggered, or inactive/not triggered.

To do this, incoming sensor data is compared against the classifier rules, starting from the first rule and working sequentially through to the bottom in order to find a match in conditions. For example, consider Table 3.1, which examines the JRip context classifier rules produced in the Weka data mining application for requirement R2 at point 18 from the time-series runtime analysis described in Section 5.3.

[2]http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/JRip.html

Table 3.1: Example of JRip Context Classifier Produced in this Study

JRip context classifier rules produced in the Weka data mining application for requirement R2 at point 18 from the time-series runtime analysis described in Section 5.3.
(Rower2SleepWake <= 0) and (Rower4SleepWake <= 0) =>classifier?=1 (6262.0/0.0)
(Rower1SleepWake <= 0) and (Rower3SleepWake <= 0) =>classifier?=1 (5651.0/0.0)
(Rower2SleepWake <= 0) and (Rower3SleepWake <= 0) =>classifier?=1 (2172.0/0.0)
(Rower4SleepWake <= 0) and (Rower1SleepWake <= 0) =>classifier?=1 (1777.0/0.0)
(Rower4InBed >= 1) and (Rower3SleepWake <= 0) and (Rower4SleepWake <= 0) =>classifier?=1 (415.0/0.0)
(Rower1SleepWake <= 0) and (Rower2SleepWake <= 0) =>classifier?=1 (202.0/0.0)
=>classifier?=0 (54524.0/0.0)

The first rule in Table 3.1 can be interpreted as '***if*** *Rower 2 is sleeping* ***and*** *Rower 4 is sleeping,* ***then*** *requirement R2 should be active/triggered*'. So, when the context aware system receives realtime sensor data from the *SleepWake* sensor for Rower 2 and the *SleepWake* sensor for Rower 4, and both indicate that those rowers are sleeping at the same time, then requirement R2 should be active/triggered. If no match can be found in any of the rules for the active/triggered state (i.e., the first six rules in Table 3.1), then the requirement is inactive/not triggered. The seventh rule in Table 3.1 indicates this and can be interpreted as '***else*** *requirement R2 should be inactive/not triggered*'.

The first number in brackets after each rule indicate the *coverage* for that rule (i.e., the number of rows of data from the training set that the rule applies to). The second number indicates how many rows in the training set were misclassified using that rule. For example, the first rule in Table 3.1 covers 6262 rows of the training set, and the last rule covers 54524 rows of data. Neither of the rules misclassified any rows of data.

3.4.2 J48 (C4.5)

The J48 algorithm is a Weka decision tree algorithm based on the well-known C4.5 algorithm [26]. Decision tree classifiers are considered among the more comprehensible of the data mining classifiers along with rules-based classifiers. According to Kotsiantis [17], where they lack comprehensibility seems to lay in the fact that they model all classes, not just the target class. The fact that it covers all classes, not just the target class, also increases complexity. Additionally, decision tree algorithms also have the ability to handle missing data and a variety of data types including discrete, binary, and continuous, and they tend to perform better than rule-classifiers with these data characteristics [17]. Along with high speed of classification, these characteristics are desirable for mobile implementation.

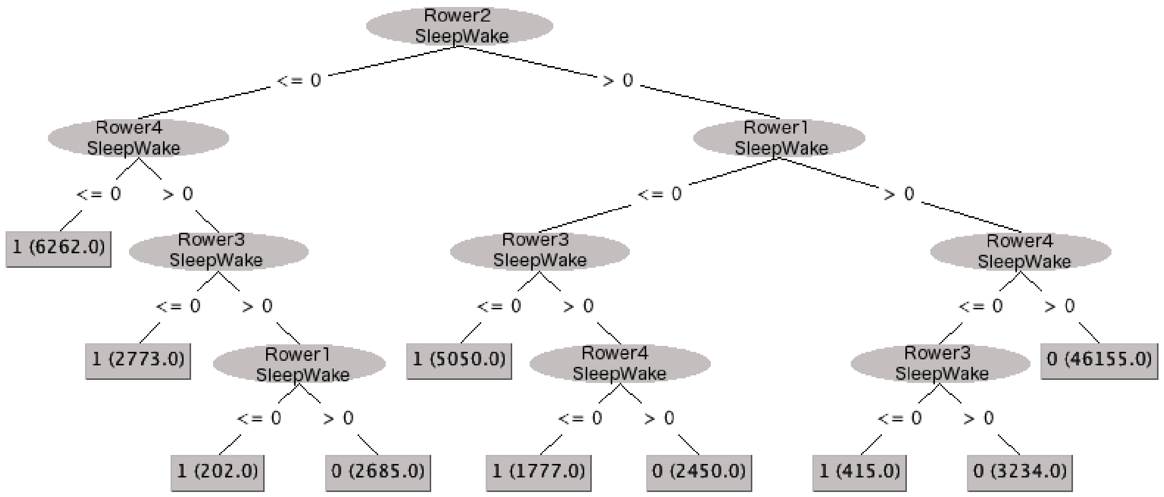

Figure 3.3: Visualization of J48 decision tree context classifier produced for R2 point 18 from the time-series runtime analysis from Section 5.3.

Algorithm Output

Output for this algorithm is conceptually similar to the JRip algorithm in that it covers the rows in the training set for each of the two classes; however, the J48 classifier takes the form of a decision tree instead of a list of rules. The J48 decision

Table 3.2: Example of J48 Context Classifier Produced in this Study

J48 context classifier rules produced in the Weka data mining application for requirement R2 at point 18 from the time-series runtime analysis described in Section 5.3.

```
Rower2SleepWake <= 0
| Rower4SleepWake <= 0: 1 (6262.0)
| Rower4SleepWake >0
| | Rower3SleepWake <= 0: 1 (2773.0)
| | Rower3SleepWake >0
| | | Rower1SleepWake <= 0: 1 (202.0)
| | | Rower1SleepWake >0: 0 (2685.0)
Rower2SleepWake >0
| Rower1SleepWake <= 0
| | Rower3SleepWake <= 0: 1 (5050.0)
| | Rower3SleepWake >0
| | | Rower4SleepWake<= 0: 1 (1777.0)
| | | Rower4SleepWake >0: 0 (2450.0)
| Rower1SleepWake >0
| | Rower4SleepWake <= 0
| | | Rower3SleepWake <= 0: 1 (415.0)
| | | Rower3SleepWake >0: 0 (3234.0)
| | Rower4SleepWake >0: 0 (46155.0)
```

tree classifier can, however, also be interpreted as a series of rules in the same way that the JRip algorithm does. For example, the J48 decision tree has the same rule as the one described in the JRip algorithm above with the same coverage. That is, looking at either the decision tree diagram in Figure 3.3 (the top and leftmost nodes), or the Weka output of the same decision tree in Table 3.2 (the first two lines), shows the rule '***if*** *Rower 2 is sleeping* ***and*** *Rower 4 is sleeping,* ***then*** *requirement R2 should be triggered/active*' with the same coverage and error as the corresponding JRip rule.

Unlike the JRip algorithm that efficiently focuses on producing context classification rules for the smaller (active/triggered) state, the J48 decision tree algorithm makes explicit all the cases where the requirement would be inactive/not triggered as well. This increases computational overhead, making this algorithm less efficient than the JRip algorithm. It does, however, show gains in accuracy, precision, and recall over the JRip algorithm[3], so the tradeoff may be worth it.

[3]These results are shown in Figures 5.11 to 5.13 from the time-series analysis in Section 5.3.

3.5 Training and Parameter Tuning

Training the classifiers for each requirement was accomplished using Weka 3.6.9, a data mining application [12]. Weka was created by the University of Waikato in New Zealand, and contains a number of machine learning algorithms that can be applied to data mining tasks. Because the focus of this study was full automation of integrated data mining, there was no parameter tuning of individual algorithms.

Training for each classifier for each requirement occurred over the entire set of available data in each of the exploratory case study and the confirmatory case study. That is, the classifiers produced to define the context for the requirement from the exploratory case study were obtained by applying data mining algorithms to the entire set of data from that study. Similarly, the classifiers produced for the eight requirements from the confirmatory case study were obtained by applying data mining algorithms to the data set from that study alone.

3.6 Evaluation with Test Set

Ten-Fold Cross Validation

Every time a classifier was produced in this study, it was evaluated using ten-fold cross validation [33]. This evaluation technique is used to determine the general performance of the classifier by breaking the training set into ten parts (folds), training on the union of nine of those folds using the desired data mining algorithm, and then testing on the one remaining fold. This process is repeated, one for each fold, and the performance results of all ten folds are averaged to obtain a reduced-variance estimate of performance rates on the training set [17]. The performance metrics produced in Weka that are averaged using ten cross-fold validation for each classifier include accuracy, precision, recall, as well as a number of others not discussed in this thesis.

These performance rates were used to determine the feasibility of the approach in the exploratory case study, and were also used to determine which data mining algorithms should be extensively evaluated in the confirmatory case study. However, the performance results in the confirmatory case study were consistently high using the ten-fold cross validation technique when evaluating each of the chosen data mining algorithms. Because of this, further evaluation was conducted through the time-series analysis detailed below in Section 3.8.

3.7 Classifiers Linking Context to Requirements

The classifier(s) produced for each requirement to sufficient accuracy, precision, and recall defined the context situations that the corresponding requirement is active/triggered in and inactive/not triggered in. The classifiers generated can be interpreted by requirements engineers to better understand the context that each requirement is active/triggered in without observing that context directly. At this point, the research goal of how unobservable system contexts can be better understood to support the requirements engineering process through data mining has been achieved.

3.8 Time-Series Analysis of Algorithm Performance on Runtime Data

While post-runtime, human-in-the-loop analysis and system evolution is valuable, the ultimate purpose of this study is to work towards fully automating this context awareness and system evolution process at runtime. In order to further explore the feasibility of this, a time-series analysis of each algorithm's performance in producing context classifiers for each requirement at runtime was undertaken on the JRip and J48 classification algorithms on the confirmatory case study data. This analysis determined the runtime performance of the JRip and J48 algorithms at successive points in time for the entire confirmatory case study data set for eight contextual requirements[4]. Through this analysis, insight was gained into when evolution of the system's definitions of the context situations for each of the requirement states might occur.

In order to complete the time-series analysis, several steps had to be taken. The graphs for the time-series analysis[5] were produced according to the following algorithm applied to the entire set of runtime data from the Dakar to Miami voyage for all requirements for the JRip and J48 algorithms:

```
for each requirement {
    define desired analysis data points
    determine data intervals in between desired analysis data points
```

[4]See Figures 5.10 to 5.13.

[5]These details are illustrated in Figures 5.11 to 5.13

```
    test set is entire runtime sensor data set
    training set is empty
    for each analysis data point {
        append interval of sensor data from the test set to the training set
        remove interval from the test set
        save training set and test set for that data point
    }
}
for each data mining algorithm under investigation {
    for each requirement {
        for each data point {
            apply data mining algorithm to training set
            (validate with 10-fold cross validation, if desired)
            apply context classifier produced to test set
            record resulting performance metrics of context classifier
        }
        graph desired performance metrics of context classifiers for requirement
    }
}
```

Define Analysis Points

First, which points in the time series should be analyzed needed to be determined. For comparison between the graphs of each of the eight requirements, and to take into account the impact of sensor configuration changes, analysis was completed in increments of approximately every three days in the data (i.e. every 4320 rows), with three additional analysis points added when sensor configuration changes occurred[6]. Running the data mining algorithms on the cumulative data gathered in increments of approximately three days gave an indication of how the accuracy of the resulting classifier changed depending on how much contextual historical sensor data was available.

[6]See Section 5.3.

Divide the Data into Training and Test Sets for each Requirement

Once each point for analysis in the time series was defined, it was divided into a training set of all sensor data available before that point, and a test set of all sensor data available after that point in terms of each of the eight requirements. Care was taken to accurately reflect sensor configuration changes at the points where they occurred at runtime by removing or adding the data from those sensors in the training set for those points accordingly. Additionally, due to a limitation in Weka, the same sensor data was removed or added from the test set.

Incrementally Create Classifiers on Training Sets and Evaluate on Test Sets

For each requirement, and for each of the data mining algorithms under investigation, the chosen data mining algorithm was applied to the training set at each point. The resulting classifiers produced by the algorithm for each point was evaluated on the test set for that point, and the desired performance metrics were graphed. Ten-fold cross validation was used for each classifier generated at each point for comparison; however, evaluation on the test set for each point may have been sufficient.

3.8.1 Performance of Algorithms over time

The changes in runtime performance of the algorithms on the data from the confirmatory case study could be seen over time. This allowed observations to be made about how long it took for each of the data mining algorithms to predict the relevant context for each requirement to acceptable levels for initiating system context awareness for that requirement. It also allowed observations to be made about how some sensor configuration changes affected the data mining algorithms' ability to accurately define relevant context situations for each of the requirement states, and how those changes would impact the system's ability to maintain context awareness for the requirements that rely heavily on particular sensors to do so.

As predicted, the approach of automatically generating a new classifier at roughly every three days of runtime sensor data was ultimately inefficient because the amount of sensor data collected automatically every three days in itself did not prove to have the most impact on the quality of the classifier produced at each point. Instead of automatically applying the data mining algorithm to historical runtime sensor data at a set interval, analysis should be performed after new sensor data pertaining to the active/triggered state of a requirement comes in (after a system ‘settling time’) until the classifier performs within acceptable bounds. For example, each time the active/triggered state for a requirement changes to inactive/not triggered until the classifier is performing with acceptable accuracy, precision, and recall (or whichever performance metrics are preferred). These times of change in system context of use are candidates for when system context evolution should occur and are discussed further in Section 5.3.4.

Chapter 4

Evaluation Methodology

This chapter details the steps taken (shown in Fig. 4.1) to evaluate the research approach described in Chapter 3.

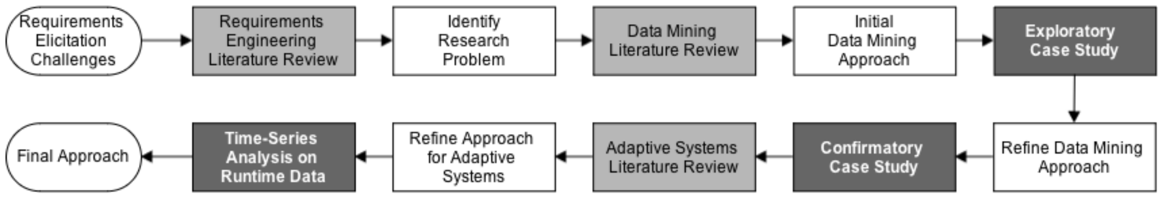
Requirements Elicitation Challenges
Requirements Engineering Literature Review
Identify Research Problem
Data Mining Literature Review
Initial Data Mining Approach
Exploratory Case Study
Refine Data Mining Approach
Confirmatory Case Study
Adaptive Systems Literature Review
Refine Approach for Adaptive Systems
Time-Series Analysis on Runtime Data
Final Approach

Figure 4.1: Evaluation Methodology.

The evaluation of the approach began in September 2012 when the design and implementation of a mobile application under development uncovered **challenges in the requirements elicitation process**. Specifically, the challenges of eliciting a complete set of system requirements, and the contexts that these requirements are active in, for users operating in environments that are unobservable by system developers.

A **requirements engineering literature review** was undertaken and aimed at uncovering effective ways of discerning and documenting requirements in two situations:

1. where designers did not have direct access to the operational environment that the users would be interacting with the system in, and

2. where a full set of system requirements was not able to be captured at design time.

The literature review revealed that complete sets of user requirements are extremely difficult to elicit, due to factors such as the users' inability to fully express (or even be aware) of them. Because it is so difficult to elicit a complete set of requirements at design time, even in observable contexts of use, systems must be evolved iteratively not only to incorporate new system requirements, but also to adapt existing requirements to changing contexts of use. Unfortunately, this evolution process is often extremely costly to undertake manually, therefore automation of this process is desirable.

A context-aware mobile application for passively collected context of use sensor data was designed for the OAR Northwest organization for context aware service delivery during system runtime as they crossed the Atlantic Ocean. The team also actively wrote daily log entries during their voyage, which was also at system runtime. Because it was not possible for developers to observe the context of use directly, the **research problem** of how to leverage passively collected sensor data and actively collected log data for the requirements elicitation process was investigated.

A literature review focused on data mining explored the possibilities of applying data mining algorithms to the sensor data collected at runtime to automatically identify the context relevant to specific requirements. Following this, the initial data mining approach to automatically define the context relevant to specific requirements was developed and an exploratory case study was undertaken to investigate the feasibility of the approach.

A pragmatic approach to Kotsiantis' process of supervised machine learning (as described in Chapter 3), explored through statistical analysis of data mining algorithms applied to sensor data was the primary means of investigation in an **exploratory case study** and a **confirmatory case study**. This approach was chosen because the ultimate goal of the investigation was to determine a way to derive context for requirements automatically; as such system adaptation decisions based on data collected at runtime alone were critical.

The approach was evaluated by comparing the context situations identified by the data mining algorithms through the context classifiers against the actual context situations in which each of the requirements under investigation should be active/triggered and inactive/not triggered. The results of these statistical analysis were further supported by data from exploratory interviews with the users pre-runtime, field logs recorded by the users at runtime, and confirmatory interviews with the users post-runtime.

Chapter 5

OAR Northwest Case Studies Results

To fulfil the research objectives, two case studies were undertaken based on historical contextual data gathered by OAR Northwest over two separate rowing voyages. The approach was applied to a single requirement in the first case study in order to explore whether or not the approach warranted further investigation. Once the approach was determined to be viable, it was refined and applied to a confirmatory case study in order to evaluate the approach on five requirements with a total of eight sets of results. The feasibility of the approach for runtime requirements evolution was further investigated with a time-series analysis on the historical contextual data gathered at runtime for each of the five requirements from the confirmatory case study. The results of all three of these analyses are presented in this chapter.

5.1 Exploratory Case Study: Circumnavigation of Vancouver Island

In this case study, a user requirement for the ToTEM mobile application was identified and the context situations for the active/triggered and inactive/not triggered states of the requirement were derived using a variety of data mining algorithms. The accuracy levels of the resulting classifiers produced were compared, and the data profile of the historical sensor data set was identified. The resulting classifiers were compared to the actual context situations in which the requirement was active/triggered and inactive/not triggered. The data mining approach was also refined.

The requirement focused on for the exploratory case study was one influenced by extreme conditions that prevented the rowers from continuing to row. In cases such as these, the rowers cannot continue with normal activities requiring the context adaptive system. Therefore, they wish for the system to disable alerts until they can begin rowing again. This state where the rowers are unable to continue rowing is described as being *On Sea Anchor*, and because of the extreme nature of the conditions that usually surround this state, they wished for the deactivation/silencing to be automated so that they were not distracted by the system during these times.

Table 5.1: Requirement Investigated in Exploratory Case Study

Requirement	Target State	Context Aware Action
R1	*On Sea Anchor*	*automatically disable system alerts*

The data set examined was collected by onboard sensors that recorded environmental and biometric data from a row OAR Northwest completed in 2012 over a period of 22 days around Vancouver Island. This voyage was undertaken to test the functionality of the rowboat and sensors, and included disembarkments at several stops. Because the data collection and goals of the rowers were flexible and inconsistent, the sensor data collected likewise had many inconsistencies.

5.1.1 Identification of Required Data

The contextual data for the exploratory case study was collected by OAR Northwest during a row around Vancouver Island, BC from April 11 to May 8, 2012. The data from the Vancouver Island row was recorded using a high-sensitivity (-160dBm) GPS telemetric tracking system, and an athlete biometric data management software system with sensors attached directly to one of the rowers. The accuracy of the ReadiBand biometric sensor is purported to have an accuracy agreement rate with polysomnography of 93% on the manufacturer's website [27], but the sensitivity and accuracy of the GPS sensor is unknown.

The telemetric data (regularly recorded roughly every 15 minutes) consisted of 15 attributes including a reading ID, coordinated universal time (UTC) and local time, position based on various instruments, direction based on various instruments, some conditions local to the boat, and technical status of the sensor and readings themselves. The biometric data (regularly recorded once every minute) consisted of time data, empty positional data, vigorousness of movement data, sleep information, and mental fatigue information.

5.1.2 Data Pre-Processing

Significant preprocessing was required to prepare the sensor data from the Vancouver Island voyage for analytics. This preprocessing was done iteratively, as shown to be sometimes necessary in Kotsiantis' approach (Figure 3.1).

Data Integration

Coordinated universal time (UTC) was used as a key to integrate data sets. However, inconsistent formatting in the UTC of the biometric data made this integration difficult. Data was examined for formatting inconsistencies and corrected. Rounding of UTC time was done to the nearest minute as the seconds between the individual sensors was not coordinated and prevented a simple merge. The resulting data set consisted of 1845 rows of data (tuples).

Data Cleaning

Rows with data that fell outside of reasonable sensor ranges were removed, as were those with all-zero sensor entries and other sensor measurement errors. The result of this cleaning process reduced the total number of tuples from 1845 down to 1592 (i.e., 13.7% of "noisy" rows were eliminated).

Data Transformation

The sensor data in each column was normalized so that it fell within a range between 0 and 1. As previously mentioned, this transformation helps improve the performance of some data mining algorithms that rely on euclidian distance to generate classifiers. Additionally, the GPS Pacific Standard Time column was converted from a column of unique values to three normalized columns of days, hours, and minutes.

Data Reduction

Exploring whether or not relevant context for specific requirements could be derived from the available sensor data was the goal at this point in the study. As such, all data from sensors that were not considered relevant to the context of requirement (R1) were removed. These included columns consisting entirely of zeros, and those pertaining to the hardware functionality of the GPS sensor, the reading ID, and UTC. Columns with redundant attributes were also eliminated, with those demonstrating the highest precision being retained in order to maximize algorithm performance [17]. The full data set also contained oceanographic context data (such as water salinity), which was excluded (validated with rowers' knowledge) based on their lack of relevance to the context surrounding when the rowers were on sea anchor. The full list of sensor data used for the exploratory case study is as follows:

- Day (from 1 to 22),
- Local Time Hour,
- Local Time Minute,
- Latitude,
- Longitude,
- Speed over Ground,
- Course Over Ground (compass direction),
- Altitude,
- CEP (unknown sensor attribute),
- Environmental Temperature,
- Actigraphy (vigorousness of movement),
- In Bed or Not,
- Asleep or Awake, and
- Mental Fatigue.

The inconsistency of sensor readings was used to infer abnormal rower behaviour during the row. That is, inconsistent sensor reading frequency was used to determine when the rowers were not actively simulating true trans-Atlantic rowing conditions. The readings were expected to be similar every day as the rowers were simulating a rigid and regulated environment. A "normal" measure of the number of rows of data per day with consistent data collection was determined to be 91 +/- 4 readings for that day. Given that little information for the Vancouver Island row outside of the passively collected sensor data was available (i.e., no actively collected data such as daily logs), an evaluation was run on the days that fell outside of this normal number of readings per day range and determined that the rowers were, in fact, on land those days. One of the rowers from the Vancouver Island trip later confirmed this fact. The rows of data from days that did not have 91 +/-4 sensor readings for that day were removed for a better classification of the *On Sea Anchor* state. This again reduced the data set from 1592 down to 1378 (i.e., an additional reduction of 13.4%).

5.1.3 Training and Test Data Sets

Data Characteristics

As indicated above, this voyage included disembarkments at several stops and goals that were not firmly consistent with the goals of the voyage in the confirmatory case study. The sensor data collected likewise had many inconsistencies as far as detecting the normal behaviour of the rowers went. Nonetheless, the purpose of the exploratory case study was to determine whether or not the context for a requirement could be derived from passively collected sensor data using data mining algorithms. It was reasoned that if such context could be derived from “noisy” data such as that from the exploratory case study, then it would be possible to derive results from the confirmatory case study. This was towards the first goal from Section 1.3.2.

Target Attribute

An analysis of when the rowers were *On Sea Anchor* from the exploratory case study data revealed that the speed of the boat naturally converged to zero. Given this, a correlation was assumed, and the *Speed Over Ground* sensor data was used to derive the target attribute for R1.

This correlation is not, of course, entirely adequate because there may have been conditions when the rowers were stopped that the system was not *On Sea Anchor* and the system alerts needed to continue to be enabled. An example of a situation like this is when the rowers need to scrape mussels from the hull of the row boat for maintenance: the boat needs to be stopped in order for the rowers to complete this task, however they may still want system alerts enabled.

Therefore, while *Speed Over Ground* was not considered a perfect target attribute for R1 in that it was not able to define when the rowers were *On Sea Anchor* on its own, it was considered an adequate starting point for system context awareness with automated context evolution being the ultimate goal for the system. That is, while using *Speed Over Ground* as a target attribute for R1 is a good starting point for context awareness, the exploratory case study was conducted with eventual context evolution in mind in order to better separate when the rowers were stopped because they were *On Sea Anchor*, and when they were stopped for other reasons.

After identifying *Speed Over Ground* as a good indicator of when the rowers were *On Sea Anchor*, an appropriate threshold needed to be established to determine when the rowers were stopped. As is consistent with Kotsiantis' approach (Figure 3.1), several iterations of refinement were used to narrow the target attribute threshold for when the rowers were *On Sea Anchor* from all rows of data where *Speed Over Ground* was below 0.05 down to all rows of data where *Speed Over Ground* was below 0.01. The frequency distribution of the sensor readings for *Speed Over Ground* can be seen in Figure 5.1. *Speed Over Ground* provides clear separation of when the rowers are stopped and when they are actively rowing between 0.01 and 0.03. The classification results with threshold 0.01 produced the rules with the highest accuracy rate across all the data mining algorithms.

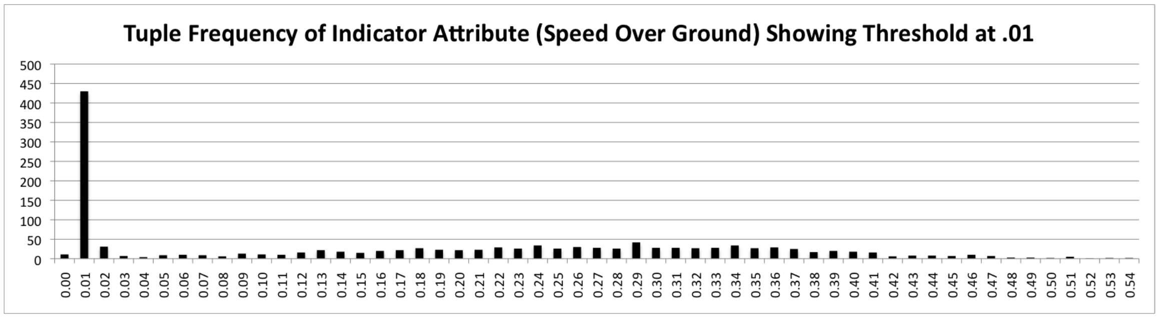

Figure 5.1: Threshold value separation at 0.01 for *On Sea Anchor* context using target attribute *Speed Over Ground* for Exploratory Case Study requirement R1.

5.1.4 Algorithm Selection

Because the data set had an unknown profile at this point (see Section 3.4), a variety of data mining algorithms considered potentially suitable for interpretation by requirements engineers and implementation on mobile devices were applied to the data set in order to determine which algorithms produced the best classification results. These included the JRip [5], J48 [26], Random Forest [34], Logistic Regression [17], and Support Vector Machine (SVM) [17] algorithms.

5.1.5 Training, Parameter Tuning, and Evaluation with Test Set

Applying the data mining algorithms to the exploratory case study data in order to train and produce classifiers to recognize the context for the *On Sea Anchor* state for requirement R1 took place in the Weka application as described in Section 3.5. Algorithms were applied first to the set of data consisting of 1592 rows and then to the set of 1378 rows with outlier days removed (as described in Section 5.1.2). As discussed in Section 3.5, parameter tuning on the various algorithms did not occur in an effort to adhere as closely as possible to lightweight, automated conditions that would be present in full implementation on a mobile device. Additionally, in order to test the algorithms for robustness, the column of *Speed Over Ground* sensor data was not included in the sensor data that the algorithms were applied to. Ten-fold cross validation (Section 3.6) was applied to all classifiers produced by all algorithms, and the performance metrics from these evaluations were used to determine which of the different algorithms applied were suitable for further analysis in the confirmatory case study.

5.1.6 Summary of Results

Results for the best performing data mining algorithms applied to the historical contextual sensor data from the exploratory case study are shown in Figure 5.2. In an effort to increase generalizability of the context classifiers produced for R1, the data mining algorithms were repeatedly applied after the data set was incrementally reduced by columns of sensor data with a significant number of unique values. For example, one-time only values such as combinations of latitude and longitude. Moreover, the entire non-error-prone data set was compared with and without outlier days. JRip, J48, Logistic Regression, and SVM were applied to the data set that included outlying days (1592 tuples). JRip, J48, Random Forest, and Logistic Regression were applied to the data set that had outlying days removed (1378 tuples).

Logistic Regression and Support Vector Machine algorithms are not included in the results table because the tree and rule algorithms are correctly classified at much better rates. This is consistent with Kotsiantis' assertion that decision tree and rules algorithms share a similar data profile [17]. However, for comparison, the Logistic Regression algorithm produced a rate of 70.9% correctly classified instanced with a false positive rate of 7.9% on the attribute set that didn't include *Latitude*, *Longitude*,

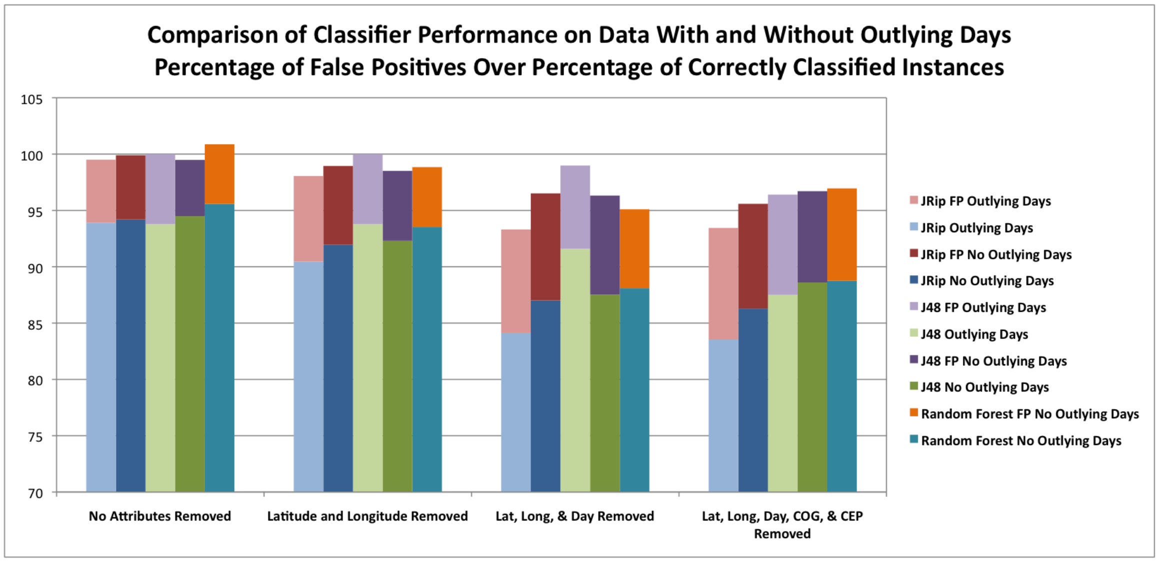

Figure 5.2: Best performing data mining algorithms on Exploratory Case Study data showing False Positive rate over Accuracy for each of the different combinations of sensor data analyzed. Performance for the JRip and J48 algorithms are shown for the data set that includes outlier days (1592 rows), and performance for the JRip, J48, and Random Forest algorithms are shown for the data set that does not include outlier days (1378 rows).

Day, *Course Over Ground*, or the unknown sensor *CEP* when applied to the data set that didn't include outlier days.

As can be seen in Figure 5.2, the tree and rule algorithms correctly classified whether or not the rowers were stopped (within the threshold of .01) 95.6% to 83.5% of the time with a standard deviation equal or less than 2% of each other depending on the attribute grouping. Similarly, the false positive rates for the same algorithms ranged from 5% to 9.9% with a standard deviation of 1%, across both data sets. The Random Forest algorithm performed the best, on average, as far as correctly classified instances goes, and the best false positive rate was divided between J48 and Random Forest for the data that didn't include the outlier days. It is unknown at this point if Random Forest would perform equally well on the data that included the outlier days.

5.1.7 Insights into Data Mining Approach from the Exploratory Case Study

As discussed in Section 1.3.2, there were three objectives in the empirical analysis undertaken in this exploratory case study. The first of these was to investigate whether or not a requirement could be linked to the context situation it is active/triggered in by applying data mining algorithms to passively collected historical sensor data. Indeed, after applying Kotsiantis' approach, it was found that the context situations for requirement active/triggered and inactive/not triggered states could be derived using the JRip, J48, Random Forest, and to a less accurate extent, other data mining algorithms as well. Second and third, the JRip, J48, and Random Forest algorithms all performed the best with accuracy of greater than 80 percent.

Following the results obtained from studying the exploratory case study data set, an extensive evaluation of the JRip algorithm and the J48 algorithm was undertaken in the confirmatory case study. The Random Forest algorithm was not used for extensive evaluation because it was considered to be less comprehensible than the JRip and J48 algorithms and also higher complexity, thus not as well suited to interpretation by requirements engineers and implementation on mobile devices.

Certain considerations into data mining for requirements were brought to the forefront as the exploratory case study was completed. These consisted of insights into the Data Reduction stage (Section 3.2). The Data Reduction stage was particularly influenced by identifying when the rowers were adhering to their scheduled rowing and sleeping regiment and consistently collecting sensor data and when they were not. That is, identifying when the rowers were actively engaged in what was to be considered normal behaviour during the confirmatory case study, and when they were taking part in other activities that would not be undertaken during the confirmatory case study. Removing sensor data from time periods of the exploratory case study that would have no bearing on the confirmatory case study (e.g., times spanning disembarkments) from the data set the classifier was trained on was an attempt to strengthen the resulting classifiers by making them more representative of the data in the confirmatory case study (see Section 3.3).

This attempt to generalize the classifiers was also reflected in the removal of columns of context sensor data that was relatively unique to the exploratory case study. For example, by removing the *Latitude* and *Longitude* columns, the context sensor data that was applicable to the OAR Northwest rows in general, rather than those specific to Vancouver Island could be scrutinized more carefully.

Additionally, removing columns of relatively unique values, such as *Day*, strengthened the classifiers in that the actual patterns in the data that defined the *On Sea Anchor* state could be more fully discerned rather than the algorithm simply identifying the most convenient sensor data that covered the *On Sea Anchor* state. For example, if the *On Sea Anchor* state only occurred on day 13, then the system may identify Day as the most important sensor to monitor for the *On Sea Anchor* state rather than the true environmental factors that really impact the rowers into entering the *On Sea Anchor* state.

5.2 Confirmatory Case Study: Transatlantic Voyage from Dakar to Miami

In the confirmatory case study, five user requirements for the ToTEM mobile application were identified and the context situations for the active/triggered and inactive/not triggered states of the requirements were derived using the JRip (RIPPER) and J48 (C4.5) data mining algorithms. The accuracy, precision, and recall levels of the resulting classifiers produced were compared. The resulting classifiers were also compared to the actual context situations in which the requirement was active/triggered and inactive/not triggered. The data mining approach was once again refined.

There were five requirements focused on for the confirmatory case study (shown in Table 5.2). The context situations for requirement R1 from the exploratory case study (*On Sea Anchor*) was further refined into *On Sea Anchor Resting* and *On Sea Anchor Active*. The context situations for requirement R1 became CR1 (*On Sea Anchor Resting*) for the confirmatory case study, and a new requirement, CR4 was associated with the *On Sea Anchor Active* context situations. The reasoning for this being that while the rowers still could not continue with their regular rowing activities when *On Sea Anchor*, they preferred to rest at night (thus the system should still *automatically disable system alerts* when *On Sea Anchor Resting*), but they would still like to be active during the day with other, non-rowing activities scheduled (thus the system should *automatically assign non-rowing activities* when *On Sea Anchor Active*).

CR2 and CR5 were a direct result of the rowers' need for cognitive offloading when

Table 5.2: Requirements Investigated in Confirmatory Case Study

Req't	Target State	Context Aware Action
CR1	*On Sea Anchor Resting*	*automatically disable system alerts*
CR2	*Two Rowers are Sleeping*	*automatically turn on wake-up alarms only*
CR3	*Mentally Fatigued but Still Rowing*	*automatically assign less cognitively challenging activities*
CR4	*On Sea Anchor Active*	*automatically assign non-rowing activities*
CR5	*One Rower is Sleeping*	*automatically set alerts to visible only (no audio alerts)*

they are mentally and physically fatigued. Because of the physically demanding nature of the rowers' activities and the dangerous environment when on the open ocean, sleep and rest times are vital for their health and safety. When the rowers are resting, they do not want their rest to be disturbed by unecessary alarms. Therefore, CR2 was defined as when *Two Rowers are Sleeping* the system should *automatically turn on wake-up alarms only*. Similarly, for CR5, when *One Rower is Sleeping*, the system should *automatically set alerts to visible only (no audio alerts)* so as to not disturb the sleeping rower while the rower that is awake is still able to receive alerts.

Requirement CR3 was also a means to support the rowers when they are extremely physically and mentally fatigued but the group is still rowing (i.e., not *On Sea Anchor*). During these times of extreme fatigue, it is difficult for the rowers to complete cognitively challenging tasks (e.g., some research tasks); however, they can still complete less cognitively challenging regular activities. Therefore, when an individual rower is *Mentally Fatigued but Still Rowing*, then the system should *automatically assign non-rowing activities*.

The goal of implementing context adaptation for all of these requirements into the scheduling system was justified by the need for the system to support the rowers with cognitive offloading during times of extreme fatigue and physical danger. All of the requirements investigated in the confirmatory case study were validated by the rowers as desirable for context awareness in the system.

The confirmatory case study data set covers a period of 64 days from a row that OAR Northwest undertook in early 2013. Like the exploratory case study, the data from this voyage was also collected by onboard sensors that recorded biometric and environmental data. The rowers had a firm user goal to row the entire voyage in under

100 days without any non-maintenance stops, and their rowing and sleeping schedule was regimented and adhered to as much as safely possible. Sensor data collection was as consistent as safety and equipment operation allowed, although there were some sensor loss and gain due to technical difficulties. Compared to the exploratory case study data set, the data collection and the goals of the OAR Northwest rowers were observably more consistent in the confirmatory case study.

5.2.1 Identification of Required Data

The contextual data for the confirmatory case study was collected by OAR Northwest during a 73-day trans-Atlantic row attempt from Dakar, Senegal to Miami, Florida, USA. The sensor data used in this study is the available data from 64 days of the row from January 22 to March 26, 2013. As stated in Section 5.1.1 the data for the exploratory case study included telemetric readings from a GPS sensor and the biometric sensor readings from one rower. The confirmatory case study included passively collected sensor data from from the same GPS sensor, the same biometric readings from all four rowers on board the boat, as well as environmental readings and telemetric readings from an Airmar PB200 WeatherStation. Additionally, daily log data from the voyage and interviews with the rowers were used to evaluate and validate the results of the confirmatory case study.

An effort was made to focus on the passively (and consistently) collected sensor data for the analysis. This was keeping in mind the goal of future full automation of context awareness and context evolution with as little actively collected data used as possible except for validation. The performance results from the classifiers produced on this passively collected data were of such a high accuracy that it was decided to focus on the runtime analysis rather than possible latent factors (e.g. from the less consistent, actively collected salinity readings).

5.2.2 Data Pre-Processing

Data Integration

Data integration was difficult for a number of reasons. In particular, a technical failure with the Airmar PB200 WeatherStation during the row made the date/time information a challenge to correct. However, disparate data sets were once again merged according to UTC time and checked to make sure offset was minimized through comparing time series graphs of redundant attributes and verification with daily logs.

Once again, the different sensors had different reading rates. The biometric sensors from each of the four rowers had a reading rate of one reading per minute. The Airmar sensor had a reading rate of once every fifteen minutes; however, there were no readings from the Airmar for a period of approximately two weeks in the middle of the voyage. Finally, the GPS sensor had a reading rate that ranged from once every fifteen minutes to once every three hours.

The question of how to integrate the data in light of these inconsistencies was solved by merging the data to the minute on the biometric sensors, and filling in the missing data from the other sensors with persistent readings up to one hour (a heuristic value) after the reading was taken. That is, for the fifteen minute gaps in between the Airmar sensor readings, the last reading was carried through the missing values until the new reading was taken. For the three hour gaps in between the GPS sensor readings, data from the last reading was only carried through for the first hour after a reading was taken. The logic behind the one hour value was that the environment (e.g. weather or wave hight) could change dramatically after that time. The reason that the readings were persisted instead of averaged, for example, was to keep the process as lightweight as possible in order to mimic conditions on a mobile device.

Data Cleaning

In the exploratory case study, rows with data that fell outside of reasonable sensor ranges were removed, as were those with all-zero sensor entries and other sensor measurement errors. This was not the case in the confirmatory case study. Instead, the erroneous values were identified as ‘missing’ to the data mining algorithms so that the rest of the relevant contextual data from each row could be taken into consideration instead of discarding it entirely.

Data Transformation

Once again, the sensor data in each column was normalized so that it fell within a range between 0 and 1 in order to improve algorithm performance. This also served to identify erroneous outliers in sensor data and made it easier to graph sensor data against each other for comparison and analysis. Additionally, the UTC column was also converted from a column of unique values to three normalized columns of days, hours, and minutes.

Data Reduction

Data was not reduced to the same degree in the confirmatory case study as in the exploratory case study. Because full automation of this process was the goal, only columns that contained no values, and columns that contained redundant date/time information were initially removed with even instrument status (e.g. battery) being left in. However, given the insights from the exploratory case study, columns with primarily unique values and those without any bearing on generalization (month, day, latitude and longitude) were also removed. The full list of sensor data from the six different sensors used in the confirmatory case study is as follows:

- UTC Hour,
- UTC Minute,
- Rower 1 Actigraphy (vigorousness of movement),
- Rower 1 In Bed or Not,
- Rower 1 Asleep or Awake,
- Rower 1 Mental Fatigue,
- Rower 2 Actigraphy,
- Rower 2 In Bed or Not,
- Rower 2 Asleep or Awake,
- Rower 2 Mental Fatigue,
- Rower 3 Actigraphy,
- Rower 3 In Bed or Not,
- Rower 3 Asleep or Awake,
- Rower 3 Mental Fatigue,
- Rower 4 Actigraphy,
- Rower 4 In Bed or Not,
- Rower 4 Asleep or Awake,
- Rower 4 Mental Fatigue

- Airmar True Wind Speed,
- Airmar Relative Wind Speed,
- Airmar Bow Wind Speed,
- Airmar True Wind Direction,
- Airmar Relative Wind Direction,
- Airmar Bow Wind Direction
- Airmar Ship Roll,
- Airmar Ship Pitch,
- Airmar GPS Speed Over Ground,
- Airmar GPS Course Over Ground,
- Airmar Atmospheric Temperature,
- Airmar Atmospheric Pressure,
- Airmar Ship Heading,
- Yellowbrick Speed over Ground,
- Yellowbrick GPS Speed over Ground,
- Yellowbrick Course over Ground,
- Yellowbrick GPS Course over Ground,
- Yellowbrick Altitude,
- Yellowbrick Battery Status,
- Yellowbrick CEP (unknown sensor attribute),
- Yellowbrick Environmental Temperature,
- Yellowbrick Distance Over Ground in Kilometers.

Unlike in the exploratory case study, durations of inconsistent behaviour were not removed from the data set because the rowers were consistently working to achieve their goal of crossing the Atlantic Ocean in under 100 days. Therefore, there were no disembarkments that needed to be accounted for.

5.2.3 Training and Test Data Sets

Data Characteristics

The goals and behaviour of the rowers was relatively consistent throughout the duration of the voyage. This allowed for a number of representative examples of both the context states that the requirements should be active in, and the context states when they should not. The rowers stated in an interview that there was an initial adjustment period to being on the boat of approximately three weeks at the beginning of the row, however the data after this initial adjustment period is considered to be representative of standard behavior. There was sensor loss and gain occurred during the voyage, resulting in sensor configuration changes and missing data. There were also rowing shift and partner changes that altered the patterns of normal behaviour for the rowers in the data.

Target Attribute

As described in Section 3.3, the target attribute for each of the contextual requirements was derived for data mining in one of three ways. These are detailed in Table 5.3 for each of the five requirements from this case study. The target attribute for each requirements was validated by rowers.

Table 5.3: Target Attribute Definition for Requirements from Confirmatory Case Study

Req't	Target State	Target Attribute Definition
CR1	*On Sea Anchor Resting*	**visual inspection** of sensor data and verification with log data
CR2	*Two Rowers are Sleeping*	**function** based on *Asleep or Awake* sensor data for all four rowers
CR3	*Mentally Fatigued but Still Rowing*	**visual inspection** of sensor data and verification with log data combined with *Mental Fatigue* sensor data **threshold** of 0.2 or less for each rower
CR4	*On Sea Anchor Active*	**visual inspection** of sensor data and verification with log data
CR5	*One Rower is Sleeping*	**function** based on *Asleep or Awake* sensor data for all four rowers

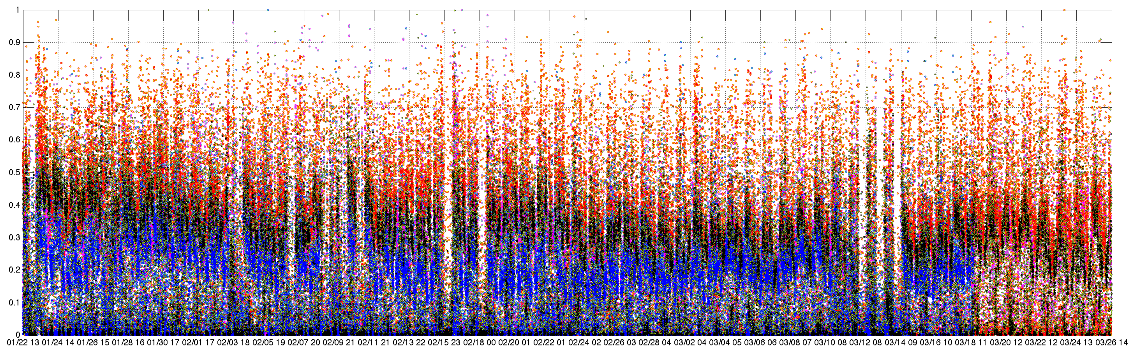

Figure 5.3: Time-series graph of *Actigraphy* sensor readings for all four rowers for all 64 days of sensor data in the confirmatory case study data set.

The target attributes for CR1 and CR4 were discerned by examining the graphed sensor data. The starting point for both of these was the *Speed Over Ground* sensor data because the target attribute for R1 was defined by a threshold from that sensor in the exploratory case study. Unfortunately, no clear threshold between stopped and moving could be discerned. *Actigraphy* for the rowers was examined next, but again, no clear threshold between stopped and moving could be discerned in this sensor data. Finally, the combined *Actigraphy* for all four rowers was graphed and periods of low actigraphy were verified with the log data (see Figures 5.3 and 5.4, and Table 5.3) for when the rowers were *On Sea Anchor Resting* during the row. This same method was used to discern the *On Sea Anchor Active* target attribute. The log data verifying that the times shown in Figure 5.4 were correctly identified as being *On Sea Anchor* is from the March 14 entry: *"We just spent 3 nights on sea anchor and as of this morning we are on the move at 1.5 to 2.0 kts."*

It is generally unclear from the graph and log data the exact minute that the rowers go on and off *Sea Anchor*, therefore minutes were rounded to the nearest quarter hour. There is also some ambiguity between when the rowers are resting, and when they are active while *On Sea Anchor*. This resulted in some overlap in the target attributes for CR1 and CR4 (e.g., Feb 9), and some gaps in coverage (e.g., Mar 12). Given this, there may be some noise in the target attributes for CR1 and CR4 surrounding these boundaries, and conflicts like this in a context aware system implementing these requirements automatically would need to be resolved. It is possible that overlap/gap situations like this would occur between other independently derived target attributes as well.

The target attributes for CR2 and CR5 were both based on functions derived

Table 5.4: Time Spans derived through visual inspection for *On Sea Anchor Resting* (CR1), and *On Sea Anchor Active* (CR4) target attributes.

On Sea Anchor Resting (CR1)	*On Sea Anchor Active* (CR4)
Jan 22 (22:00) - Jan 23 (11:00)	
	Jan 27 (17:00 - 22:00)
	Jan 31 (12:00 - 14:00)
Feb 3 (12:00) - Feb 4 (07:00)	
	Feb 6 (19:00 - 22:00)
Feb 6 (20:00) - Feb 7 (08:00)	
Feb 8 (21:00) - Feb 9 (08:00)	
	Feb 9 (08:00 - 21:45)
Feb 9 (21:00) - Feb 10 (08:00)	
Feb 10 (21:00) - Feb 11 (08:00)	
	Feb 15 (18:00 - 21:00)
Feb 15 (21:00) - Feb 16 (11:00)	
	Feb 16 (10:00 - 17:00)
Feb 17 (21:00) - Feb 18 (09:00)	
	Feb 18 (09:00 - 12:00)
	Mar 11(19:00 - 22:30)
Mar 11 (22:00) - Mar 12 (09:00)	
	Mar 12 (10:45 - 23:00)
Mar 12 (23:00) - Mar 13 (09:00)	
	Mar 13 (10:45 - 23:00)
Mar 13 (22:00) - Mar 14 (09:00)	
	Mar 14 (10:00 - 12:00)

from the *Asleep or Awake* sensor data for all four rowers. The *Asleep or Awake* sensor reading in the data is represented as a binary value of either '0' indicating the rower is asleep or '1' indicating the rower is awake.

The function to define the target attribute for each each row of data for CR2 added the readings for all four of the rowers' *Asleep or Awake* sensor data with the sum of all four variables less than or equal to 2 for all rows of data up and including row 79228 (see Table 5.5 for equations). After row 79228, the function needed to be re-defined to take into account the loss of the sensor from *Rower 3* had on the logic of the equation. Because the function could no longer determine when exactly two or three rowers were awake from the remaining three biometric sensors from *Rower 1*, *Rower 2*, and *Rower 4* alone, the function had to be redefined from row 79229 on to take this into account (see Table 5.6). A total of 3008 rows of data out of 90748 in

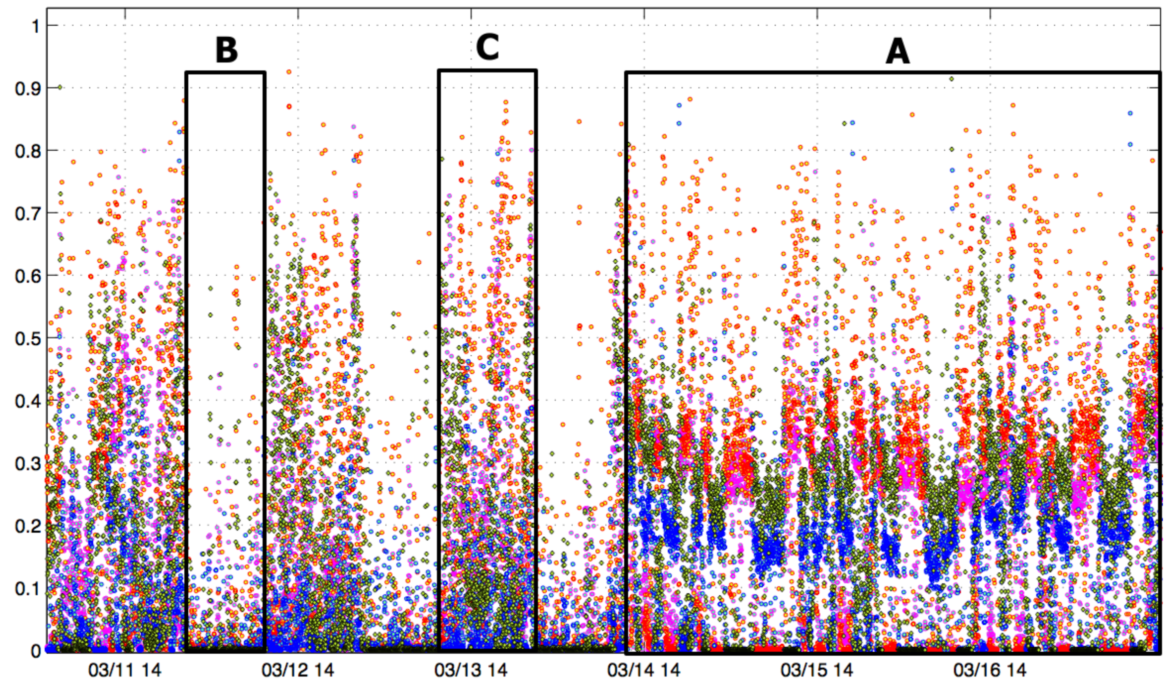

Figure 5.4: Graph of combined actigraphy for all four rowers demonstrating examples of (A) normal rowing behaviour in shifts by teams of two rowers at a time, (B) *On Sea Anchor Resting* context, and (C) *On Sea Anchor Active* context.

the data set could not be assigned active/triggered or inactive/not triggered classes for requirement CR2 by defining the target (indicator) attribute in this way because of the sensor loss from *Rower 3*.

The function used to define the target attribute for CR5 added all the entries of all four of the rowers' *Asleep or Awake* sensor data with the sum of all four variables equal to 3 up to row 79228 (see Table 5.5 for equations). Again, the sensor loss from *Rower 3* at row 79229 in the data table, had an impact on the logic of the function, and the active/triggered situation for CR5 could no longer be determined after that point. This was because it could not be determined when exactly three rowers were awake from the remaining three biometric sensors from *Rower 1*, *Rower 2*, and *Rower 4* alone. The target attribute for from row 79229 to row 90748 had to be recalculated to reflect the uncertainty that the missing data introduced. This can be seen in Table 5.6. A total of 9854 rows of data out of 90748 in the data set could not be assigned active/triggered or inactive/not triggered classes for requirement CR5 by defining the target (indicator) attribute in this way. In addition, no active/triggered situations could be identified for CR5 after the sensor loss using the function alone to define the class for each row in the target (indicator) attribute. The Active Context Precision and Recall graphs for CR5 in Figures 5.10 and 5.12 reflect this as they cut off at the point of biometric sensor loss from *Rower 3*.

The rows that had an unknown target (indicator) attribute class (instead of an active/triggered or inactive/not triggered class) assigned to them were ignored by the data mining algorithms when training the classifiers for CR2 and CR5, and were ignored from the performance results when evaluating the classifiers for CR2 and CR5 on the test sets.

In addition to the unknown class for some of the rows, there was a second reason that the functions used to define the target attribute classes for CR2 and CR5 were not entirely sufficient. This was for a similar reason that the target attribute for R1 in the exploratory case study was not entirely adequate. The granularity of the target attributes was not quite fine enough. That is, even though we can judge when the rowers are asleep or awake, we cannot indicate when the mobile device with our system on it is inside or outside the cabin where the rowers are sleeping. Therefore, the target attribute would need to be trained for instances when 2 rowers may be *In Bed*, but the device may be outside of the cabin. (Just like in the exploratory case study when the rowers may be stopped, but not *On Sea Anchor*.) This refinement may depend on specific rower combinations and times. For example, *Rower 1* and *Rower 2* may prefer to have the device inside of the cabin during the night sleep time, but during the day sleep time for *Rower 1* and *Rower 2*, *Rower 3* and *Rower 4* may prefer to keep it outside with them so that they can be alerted as to when they need to complete research activities. This kind of situation is another motivation for implementing system context evolution to support context awareness.

The target attributes for CR3 for each of the four rowers were individually derived from a combination of the states that were **not** *On Sea Anchor*, and when each of the rowers' *Mental Fatigue* sensor readings were equal to or below 0.2. This was because it was observed in the graphed data that *Mental Fatigue* levels improved when the rowers were in either of the *On Sea Anchor Resting* or *On Sea Anchor Active* situations, and the requirement was to assign less cognitively intensive tasks to the rowers when they were fatigued, but still attempting to continue to row. Because the rowers expressed a desire to continue rowing as regularly as possible, we chose a heuristic threshold of less than 0.2 (20%) for *Mental Fatigue* levels. According to the manufacturers of the biometric sensors, any *Mental Fatigue* readings below 60% have a blood alcohol equivalent of greater than 0.11%, and risk of accidents or serious errors are very high

[24]. Mental fatigue levels for each of the rowers in the confirmatory case study data set can be seen in Figure 5.5.

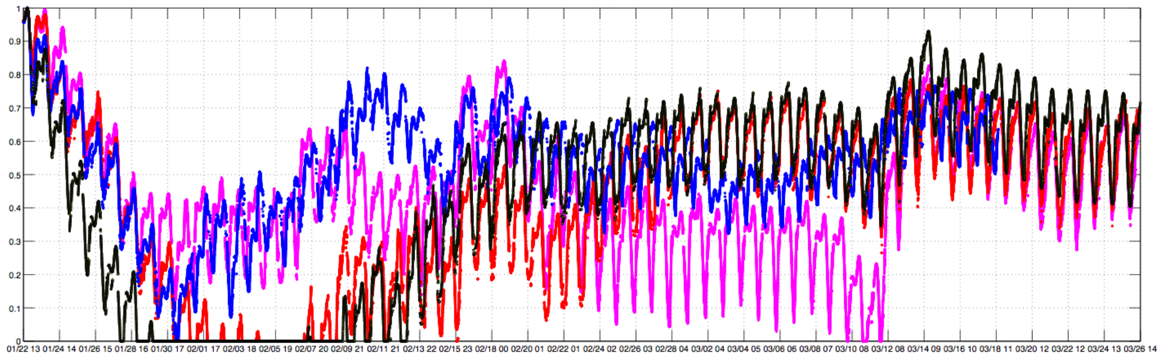

Figure 5.5: Time-series graph of *Mental Fatigue* sensor readings for all four rowers for all 64 days of sensor data in the confirmatory case study data set.

5.2.4 Algorithm Selection, Training, Parameter Tuning, and Evaluation

Given that this was a confirmatory case study based on the results of the exploratory case study, only the JRip and J48 algorithms were selected. Training occurred on the entire confirmatory case study data set, and stratified ten-fold cross validation was used to compare performance.

5.2.5 Summary of Results

Performance results using ten-fold cross validation were high, as can be seen in Tables 5.7 and 5.8. These results are from point twenty-two of the Time-Series Analysis on Runtime Data from Section 5.3 and are representative of results from the entire data set.

When the confirmatory case study data was examined under the same process used for the exploratory case study data, it was confirmed that context for the eight requirements could be derived using data mining algorithms over the entire 64 days of data. It was also confirmed that both the JRip and J48 algorithms could define context for the given requirements to an accuracy of between 95% to greater than 99% when evaluated with the ten-fold cross validation technique.

5.2.6 Insights into Data Mining Approach from the Confirmatory Case Study

How to effectively correlate requirement active/triggered and inactive/not triggered states with context situations through the target attribute was one of the major concerns of this study. A target attribute that is truly reflective of the times when each requirement is active/inactive is critical to the data mining algorithms' ability to define context from sensor data for those requirements. Intuitively, it is also vital for ensuring that the context derived by the data mining algorithm is truly representative of the context that the requirement is active/inactive in so that system context awareness is properly implemented.

In order to ensure that the data mining algorithms would produce accurate and relevant context from the contextual sensor data provided and the derived target attributes, the target attributes were verified and validated through multiple means, when possible. These included looking for patterns in the sensor data that indicated relevant context for a requirement (see Figure 5.4), and then verifying that those patterns were representative of the context for the requirement under investigation (verifying and validating with log data and interviews with the users).

Ideally, the target attribute would be passively captured by the system when the requirement is active/triggered, and inactive/not triggered. Input may also be captured from the users as to when the requirement should be active or inactive. However, neither method on its own may be entirely precise, (thus, introducing noise into the target attribute and producing less accurate results with the data mining algorithms). Therefore, target attributes for requirements for context awareness should be correlated to passively collected sensor data and active user input for verification and validation when possible.

Because the performance results from the analysis of the JRip and J48 algorithms confirmatory case study had such promising implications for context awareness, further analysis was undertaken to investigate the potential for using data mining algorithms for context evolution for each of the requirements. As outlined in Section 1.3.4, this analysis focused on when over the 64 day period the context situations derived for each requirement using the data mining algorithms reached acceptable accuracy, precision, and recall levels. Additionally, the robustness of the JRip and J48 algorithms' ability to continue to derive relevant context situations for each of the requirements given significant sensor configuration changes was investigated.

5.3 Time-Series Analysis on Runtime Data

As discussed in Sections 1.3.4 and 5.2.5, the purpose of the time-series analysis on the runtime data from the OAR Northwest Dakar to Miami row was to investigate the applicability of the JRip and J48 data mining algorithms to runtime context evolution. Through this time-series analysis on the JRip and J48 algorithms, the following were observed:

1. the runtime performance of the JRip and J48 algorithms in their ability to predict the relevant context situations for each contextual requirement from the Dakar to Miami voyage if the requirements had been implemented for context awareness and context evolution at runtime, and
2. how sensor configuration changes would have affected the data mining algorithms' ability to accurately define relevant context situations for each of the requirements at runtime, thus affecting the ability of the system to provide context awareness for that requirement.

5.3.1 Application of Approach

Define Analysis Points

Analysis points for the time-series analysis were set for intervals of approximately three days of runtime sensor data (4320 rows). This was a heuristic interval that attempted to separate significant events in the contextual sensor data to a granularity such that their impact on the performance of the context classifiers could be observed in the resulting graphs. This was considered to be particularly important during the early points on the graph where observation of the 'critical mass' point of the data needed to produce acceptable context classifiers for context awareness for each requirement was desired. The times of the analysis points were the same for both the JRip and J48 algorithms over all requirements in order to more easily compare final results.

Three additional analysis points were added to the graph when sensor configuration changes occurred in order to observe what impact these changes would have on context awareness for the remainder of the data set. These points are indicated by the filled black circles in Figures 5.6 to 5.13, and the interval of contextual sensor data immediately preceding these points are less than three days. This resulted in a combined total of twenty-two points for the time-series analysis for each graph.

Divide the Data into Training and Test Sets for each Requirement

Training and test sets for each requirement were produced using Matlab scripts. This resulted in a total of twenty-two training and test sets for each of the eight requirements under investigation. Due to a limitation in Weka, sensor loss and gain was accounted for by removing those columns from the resulting test and training sets accordingly (also done in Matlab).

Incrementally Create Classifiers on Training Sets and Evaluate on Test Sets

Context classifiers using each of the JRip and J48 algorithms were systematically produced on each of the twenty-two training sets (one for each point) for each requirement. The accuracy, precision, and recall for each of the context classifiers produced at each point using stratified 10-fold cross validation was very high, as can be seen in figures 5.6 to 5.9. Because the results were so high, the context classifiers were further evaluated on the corresponding test sets for each point for each requirement. The performance results from these evaluations were recorded and graphed as shown in Section 5.3.2

5.3.2 JRip and J48 Algorithm Performance Over Time

The matricies of graphs shown in Figures 5.10 to 5.13 show the time-series analysis performance results for the JRip and J48 algorithms on the eight requirements from the confirmatory case study. Columns represent the results for each requirement investigated, while the rows indicate the performance metric graphed (i.e., overall accuracy of the context classifier, precision for the active/triggered context for the requirement, recall for the active/triggered context for the requirement, precision for the inactive/not triggered context for the requirement, and recall for the inactive/not triggered context for the requirement). The x-axis represents the date the test set for each point includes contextual sensor data up to. The grey diamonds' positions on the y-axis represent how the classifier produced at that point on the training set performed on the test set (i.e., all remaining data after that point). This value of

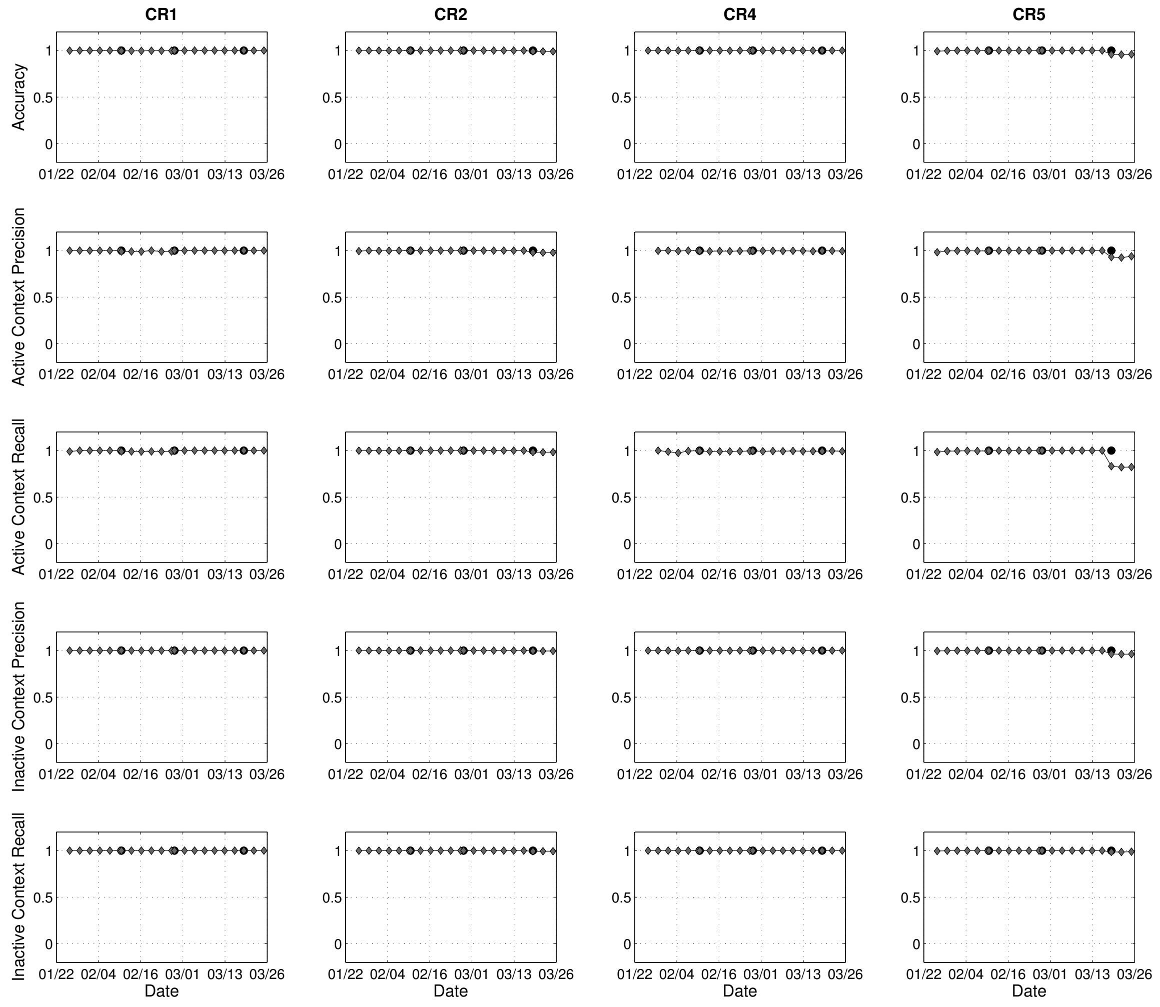

Figure 5.6: JRip algorithm stratified 10-fold cross validation results for CR1, CR2, CR4, CR5.

the y-axis in each graph is shown as a normalized percentage for ease of comparison across graphs.

The filled black circles at the top of the graphs indicate a time when sensor configuration changes occur. The interval between the first and second circles indicates when the Airmar PB200 WeatherStation did not provide sensor data. The third circle indicates when biometric data from *Rower 3* was lost (up to the end of the voyage). It should also be noted that the active/triggered state is not generally well-distributed throughout the data, except possibly in the cases of CR2 and CR5. For the rest of the requirements, data on the active/triggered state tends to be clustered together in one or more groups throughout the data.

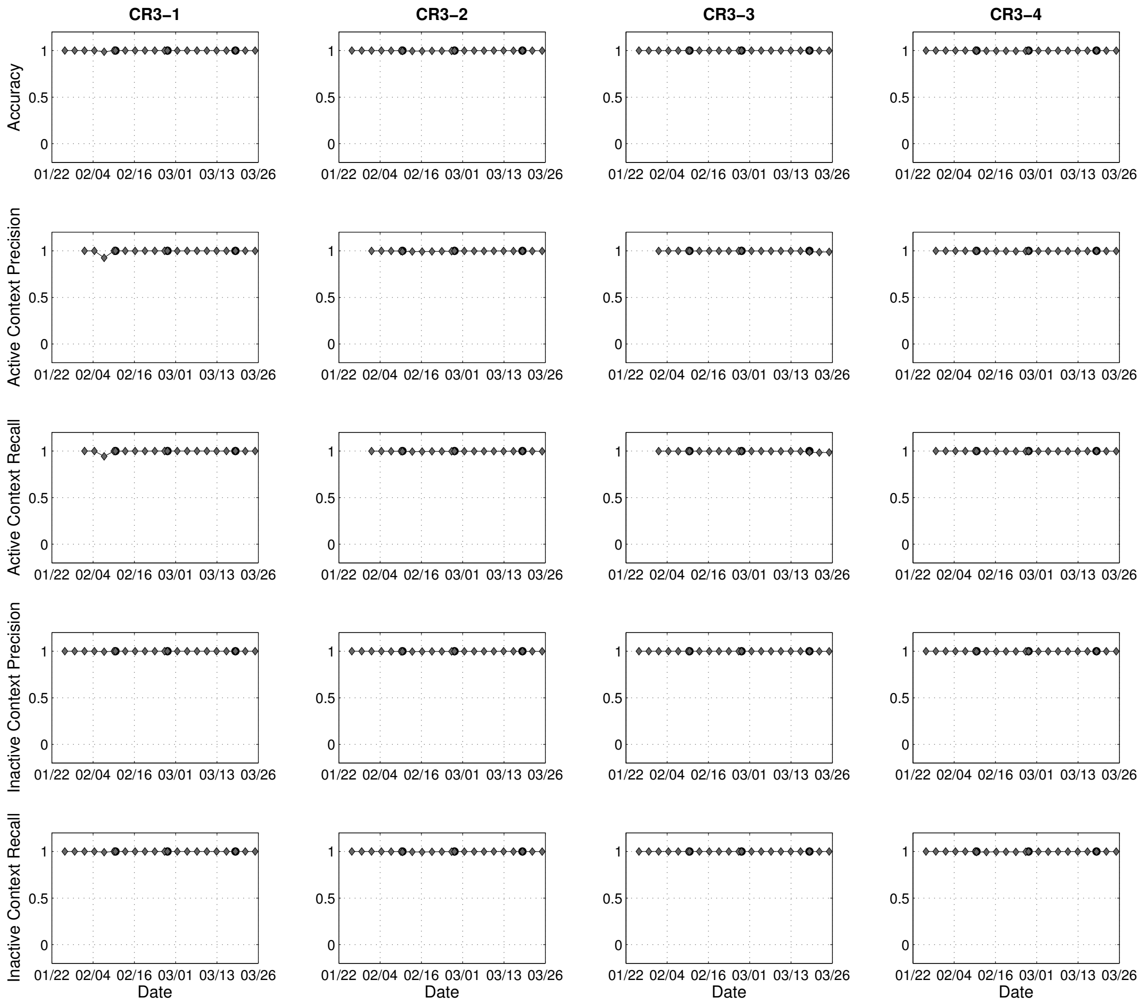

Figure 5.7: JRip algorithm stratified 10-fold cross validation results for CR3-1, CR3-2, CR3-3, CR3-4.

Accuracy

The accuracy of the context classifier in the top row is a calculation of the number of rows of data from the test set that the classifier correctly identified as being active/triggered and inactive/not triggered divided by the total number of rows classified in the test set (including the ones that were not classified correctly). While this is a good indicator of an algorithm's general performance, it is not entirely sufficient for our purposes because the majority of the rows of data in the data set are not in the active/triggered state for the requirement (see Table 5.9 for details). Therefore, the individual performance of each of the active/triggered and inactive/not triggered states needs to be examined as well.

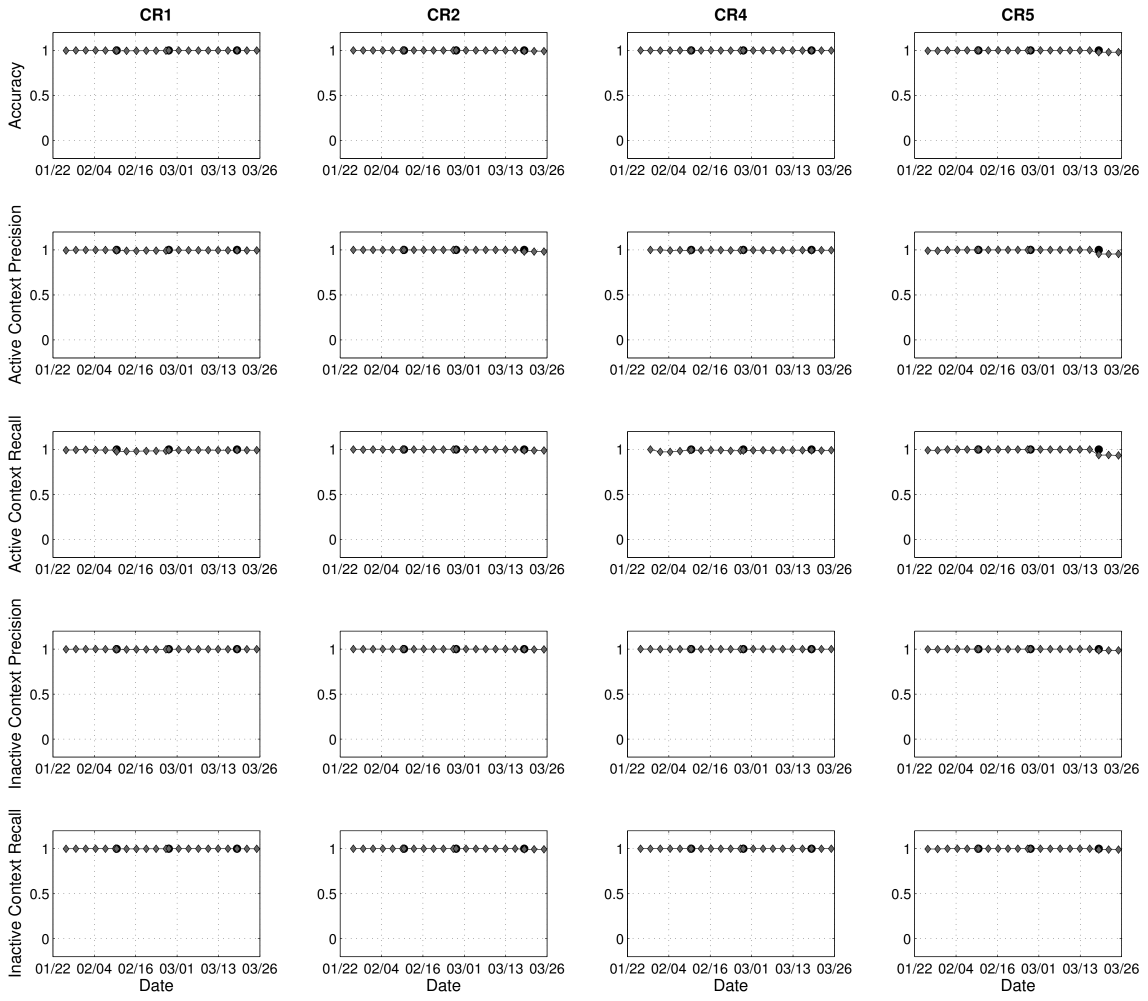

Figure 5.8: J48 algorithm stratified 10-fold cross validation results for CR1, CR2, CR4, CR5.

Precision

Precision is an indicator of how well the context classifier can discriminate between the active/triggered and inactive/not triggered states, and is a measure of the number of false positives (FP) that the classifier generates in a test set. It is also calculated at each point in the graphs for how the context classifier produced at that point performs on all the data in the test set for that point (i.e., all remaining sensor data after that point). The more times the context classifier at a point indicates that the wrong context is the right context in the corresponding test set, the lower the precision will be. Conversely, the less the context classifier incorrectly identifies the wrong context as being the right context in the corresponding test set, the higher the precision will

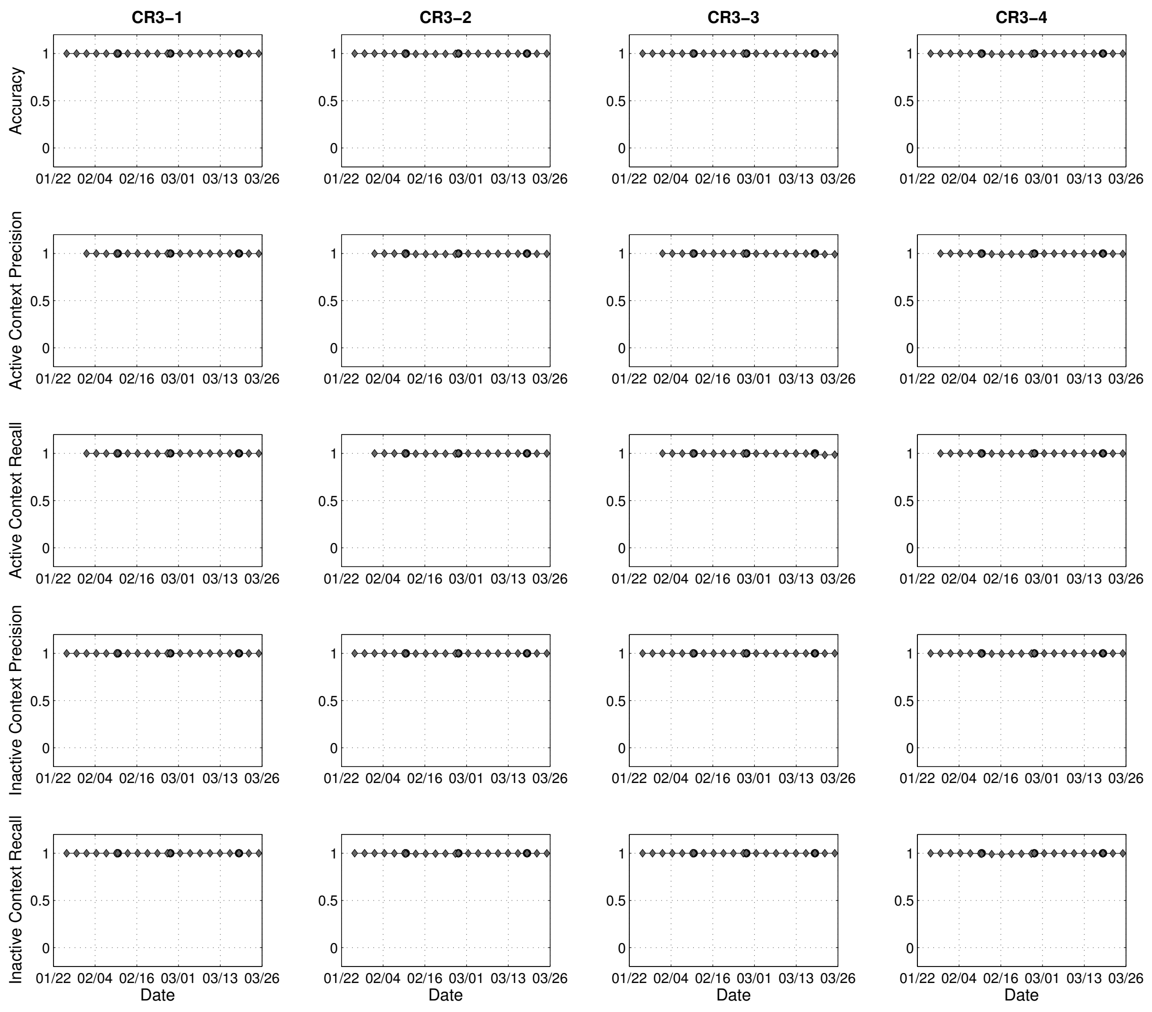

Figure 5.9: J48 algorithm stratified 10-fold cross validation results for CR3-1, CR3-2, CR3-3, CR3-4.

be.

Precision for the active/triggered state represents the number of rows of data in the test set that the context classifier correctly identifies as being in the active/triggered state divided by the number of rows of data from the test set that the classifier correctly and incorrectly identifies as being in the active/triggered state. Precision for the inactive/not triggered state is also calculated by the number of rows of data in the test set that the classifier correctly identifies as inactive/not triggered divided by the number of rows of data from the test set that the classifier correctly and incorrectly identifies as being in the inactive/not triggered state. Precision for the active/triggered state for the requirement in each column is shown in the second row

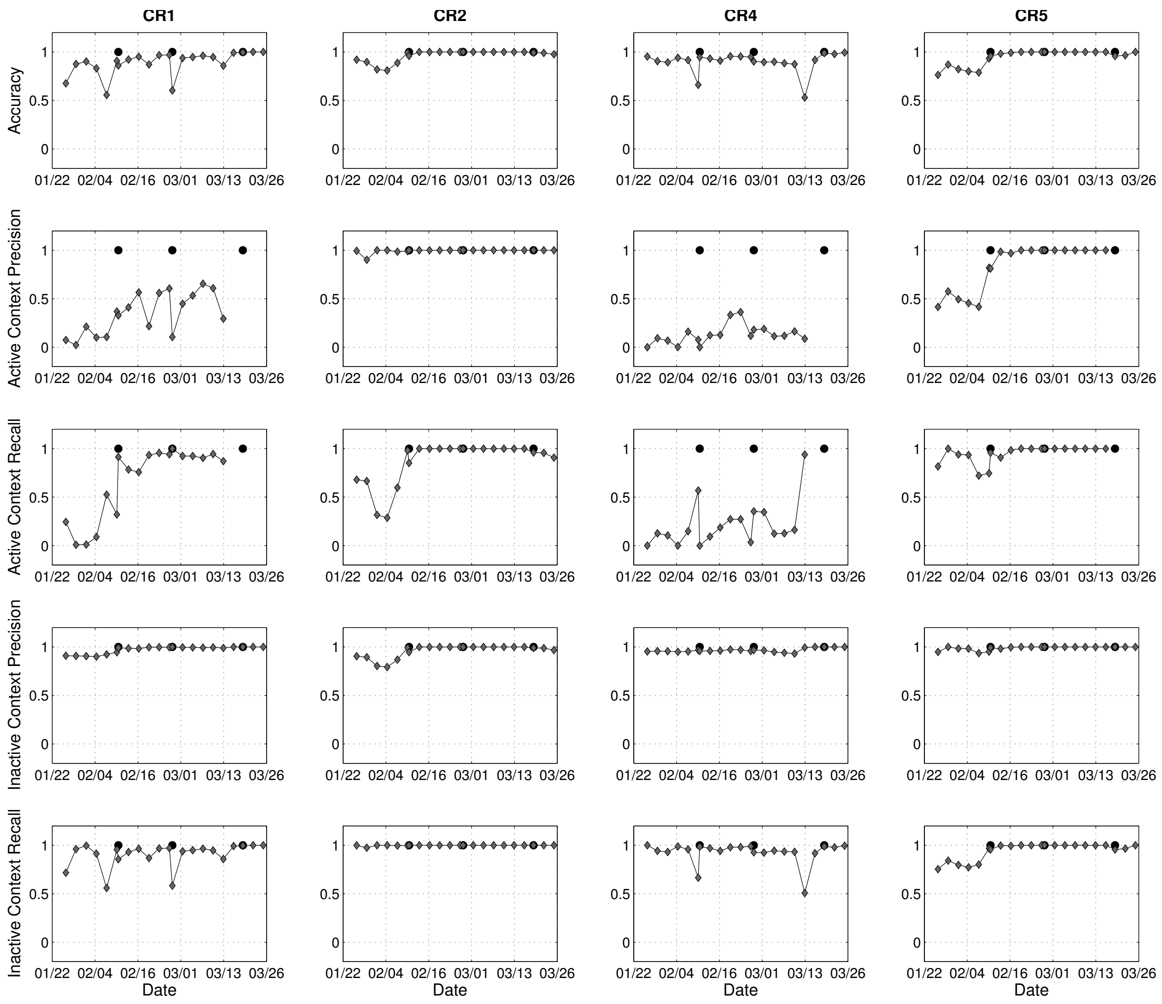

Figure 5.10: JRip algorithm performance analysis on runtime data over time for CR1, CR2, CR4, CR5.

of Figures 5.10 to 5.13, and in the fourth row for the inactive/not triggered state.

Precision graphs in the second and fourth rows terminate when the target state for the requirement (i.e., active/triggered for the third row or inactive/not triggered for the fifth row) no longer occur in the test set. This is because the precision calculation in these cases produces a 0/n result.

Recall

Recall is an indicator of how well the context classifier can recognize the context state it's looking for, and is a measure of the number of false negatives (FN) that the classifier generates in a test set. It is also calculated at each point in the graphs

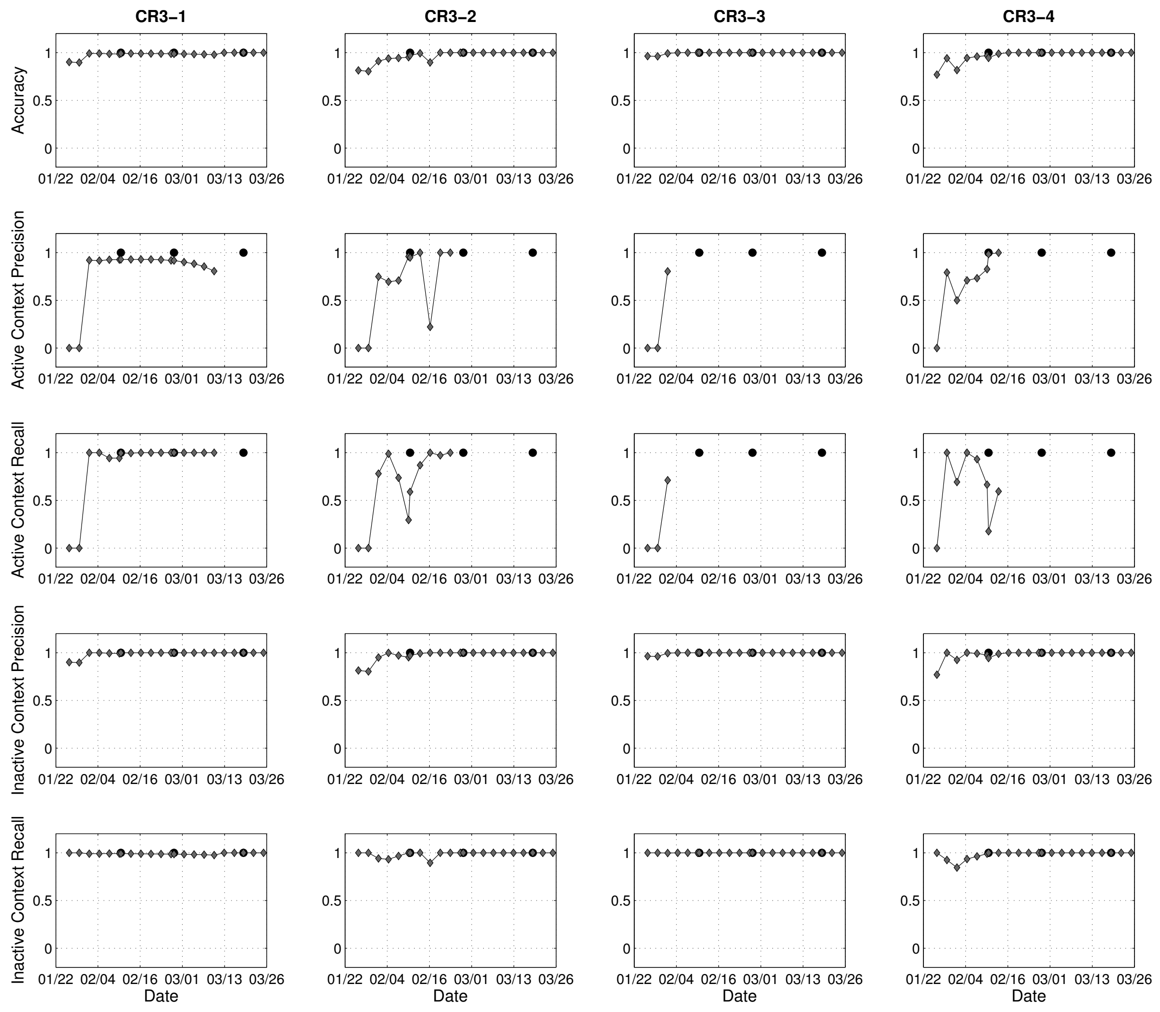

Figure 5.11: JRip algorithm performance analysis on runtime data over time for CR3-1, CR3-2, CR3-3, CR3-4.

for how the context classifier produced at that point performs on all the data in the test set for that point (i.e., all remaining sensor data after that point). The more times the context classifier misses identifying relevant context as being such, the lower the recall will be. Conversely, the more rows in the test set that the corresponding context classifier correctly identifies as being relevant, the higher the recall at that point will be.

Recall for the active/triggered state calculated at each point in the time-series runtime analysis represents the number of rows of data in the test set that the classifier correctly identifies as active/triggered divided by the number of rows of data in the test set that actually are in the active/triggered state in the test set. Recall for

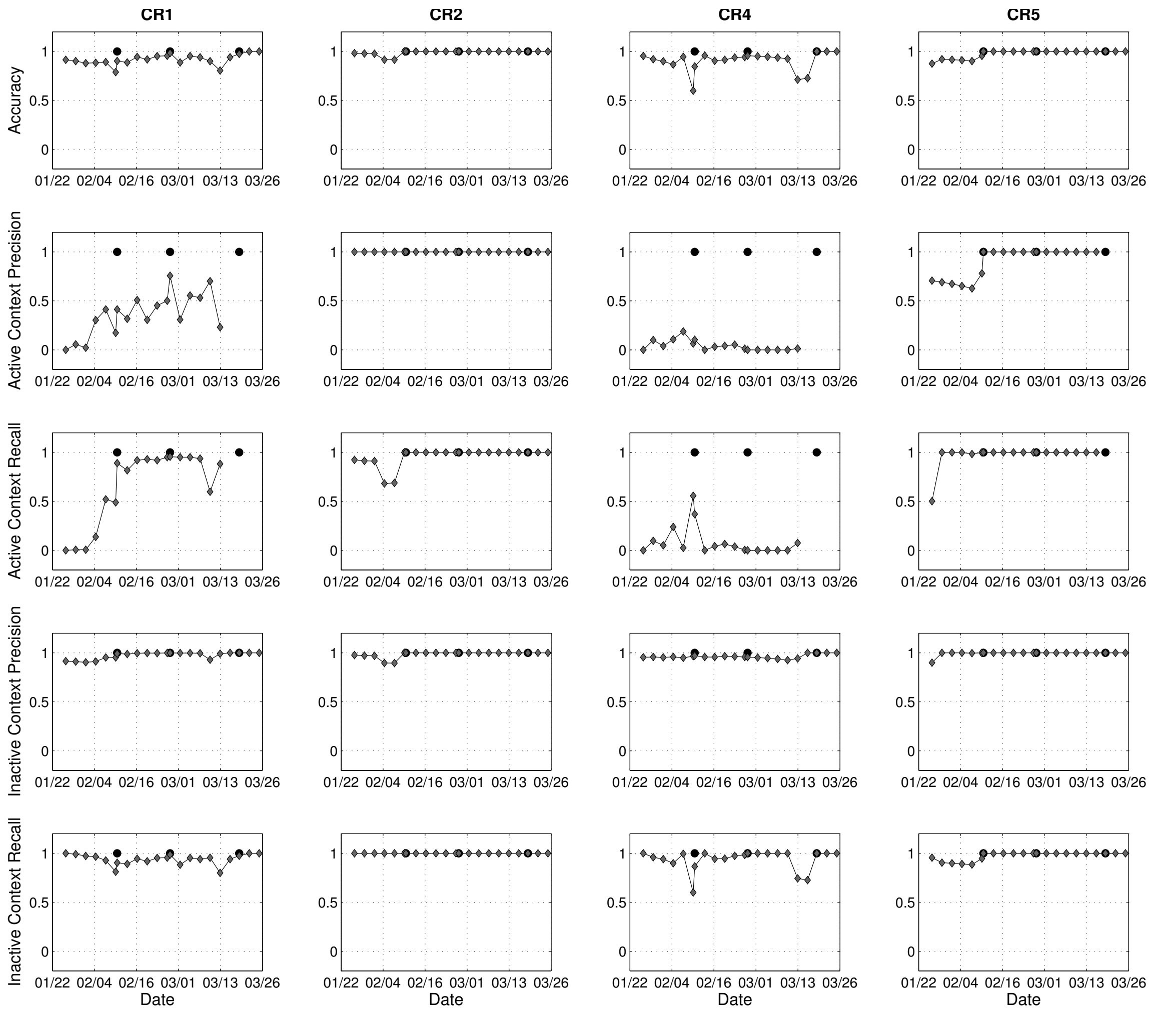

Figure 5.12: J48 algorithm performance analysis on runtime data over time for CR1, CR2, CR4, CR5.

the inactive/not triggered state is also calculated in the same way with the number of rows of data in the test set that the classifier correctly identifies as inactive/not triggered are divided by the number of actual rows of data in the test set that are indicated as being in the inactive/not triggered state. Recall for the active/triggered state for each requirement is shown in the third row of Figures 5.10 to 5.13, and in the fifth row for the inactive/not triggered state.

Recall graphs in the third and fifth rows terminate when the target state for the requirement (i.e., active/triggered for the third row or inactive/not triggered for the fifth row) no longer occur in the test set. This is because the recall calculation in these cases produces an 0/0 result.

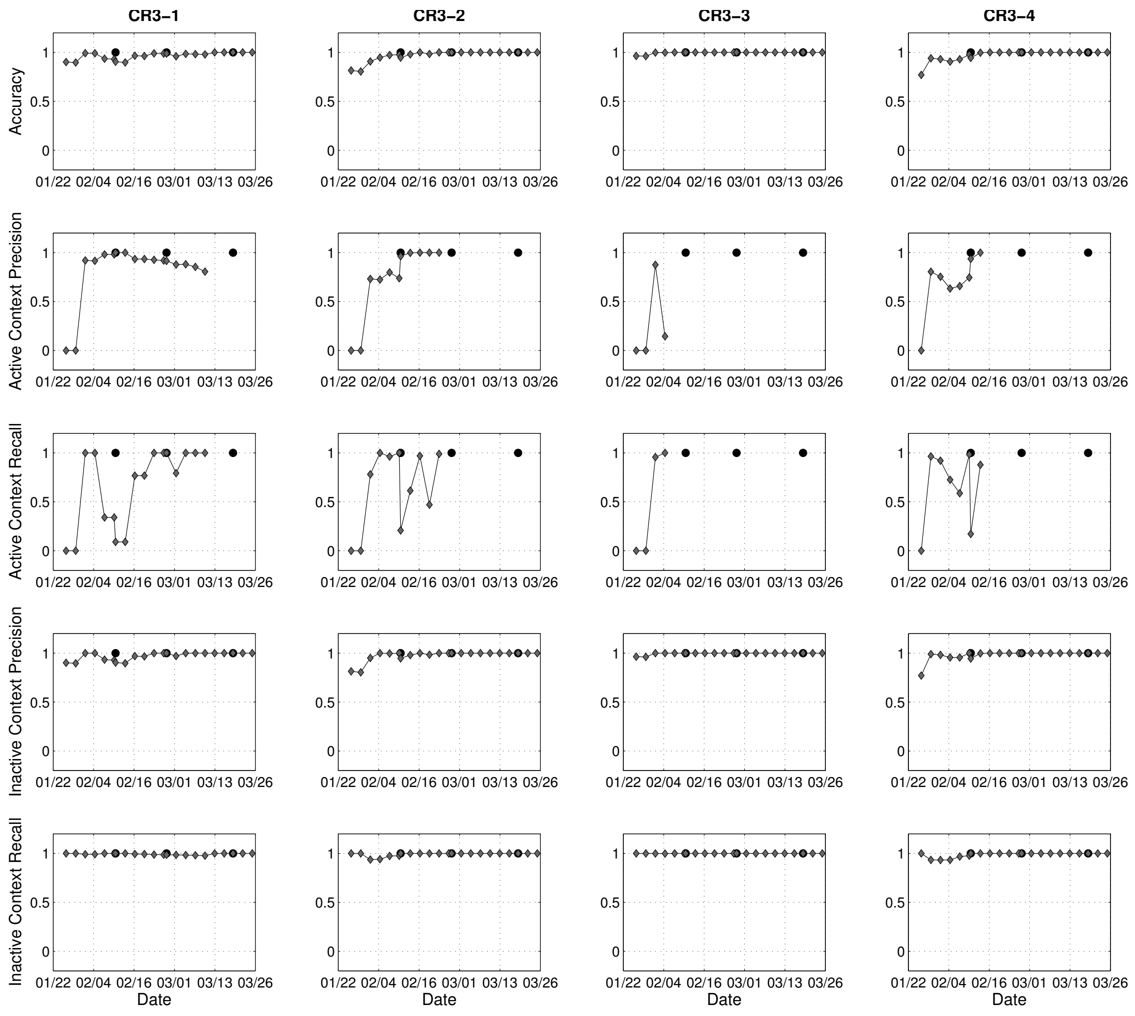

Figure 5.13: J48 algorithm performance analysis on runtime data over time for CR3-1, CR3-2, CR3-3, CR3-4.

Results for Requirement CR1

As discussed in Section 5.2.3, the target attribute for CR1 was visually defined and verified with daily log data gathered during the row. The active/triggered state occurred in clusters as shown in Table 5.4, and more instances of the *On Sea Anchor Resting* state were added to the training set as the time-series analysis progressed through the runtime data set. There were no unknown classes in the training and test data sets for this requirement.

As can be seen in Figures 5.10 and 5.12, accuracy for both the JRip and J48 algorithms on CR1 was generally high with the J48 algorithm performing better overall. A significant dip in accuracy occurred at point 5 followed by a sharp rise at point 6 occurred in the data for the JRip algorithm that did not occur in the J48 algorithm. This dip/rise occurs in the midst of a cluster of *On Sea Anchor Resting* data being added to the training sets (refer to Table 5.4). While the active/triggered precision and recall around these points generally show improvement, the recall for the inactive/not triggered state dips significantly. This would seem to indicate that inactive/not triggered state data between points 4 and 5 (added to the training set for point 5) for CR1 holds very little value to classifying context in the test sets after that point. Indeed, while the JRip algorithm suffers this dip, J48 algorithm seems to account for this lack of relevance, and no dip in accuracy is noted. However, both algorithms also experienced a slight dip in accuracy at point 18, which is in the midst of another cluster of new *On Sea Anchor Resting* state data. A dip in precision at point 10 for the active/triggered state coincides with the addition of more *On Sea Anchor Resting* active/triggered state data. As expected, significant new data for the *On Sea Anchor Resting* state has an impact on the context classifiers derived for CR1.

Another, similar dip in accuracy, precision for the active/triggered state, and recall for the inactive/not triggered state occurs in the JRip graph at point 13, at the same time as a sensor configuration change (see Table 5.10). A similar, though generally less pronounced dip at the same point can be seen in the J48 graph. Interestingly, another slight dip occurs previously at point 7 for both algorithms where the Airmar sensor data is lost, however, it is small in comparison. Surprisingly, the dip at point 13 does not coincide with a *loss* in the Airmar sensor data, but as soon as the Airmar data is *regained*. This would seem to indicate that when the Airmar data from the beginning of the data set up to point 7 is re-included in the training set, it introduces noise into the training set (remember that the Airmar sensor data was not available between points 7 and 13). This is, however, quickly compensated for, as an improvement in accuracy can be seen in points 14 for the JRip algorithm, and point 15 for the J48. The loss in the sensor data for Rower 3 at point 20 does not seem to have any impact on the accuracy of CR1.

As expected, sensor changes may have an observable impact on the performance of the context classifiers derived for CR1, depending on the relevance of the particular context the sensors represent for the requirement. Surprisingly, the individual biometric sensor loss of *Rower 3* did not have a significant impact on the performance of the algorithms, even though that same data was crucial to defining the target attribute for CR1. In addition, the re-introduction of the older (greater than approximately two weeks) Airmar sensor data to the training set introduced a significant decline in accuracy, precision, and recall performance, however this performance was nearly regained after a few more days of Airmar data was added to the training set.

Observable rowing shift/partner changes in the data (see Table 5.10) did not appear to directly impact the algorithm performance. This seems intuitive because the data from the four rowers was considered as a group when the target attribute was defined for CR1. Nonetheless, because these partner changes both occur immediately after the introduction of significant contextual data for the *On Sea Anchor Resting* state, their impact may be obscured by other factors.

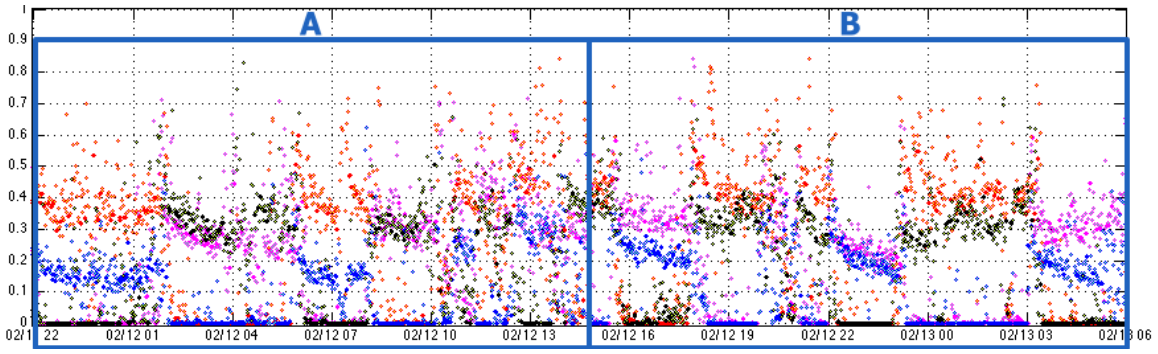

Figure 5.14: Time-series graph of combined *Actigraphy* sensor readings showing one of the two observable rower partner changes (see Table 5.10). In (A), the red and blue rowers are rowing partners, and the magenta and black rowers are partners. In (B), the rowers have switched to red and black as partners, and magenta and blue as partners. This rowing shift change occurred on February 12.

Results for Requirement CR2

As discussed in Section 5.2.3, the target attribute for CR2 was defined by functions based on the *Asleep or Awake* sensors from all 4 rowers up to row . The active/inactive state occurred fairly regularly throughout the entire set of data. There were 3008 rows of data where the active/triggered and inactive/not triggered class could not be defined for the target attribute by the functions (refer to Section 5.2.3 for details).

As can be seen in Figures 5.10 and 5.12, overall accuracy peaked at point 6 for both the JRip and J48 algorithms. This time period coincided approximately with the end of the three-week time period the rowers took to get used to being on ocean and rowing in a rigid schedule of alternating shifts. Because the patterns in group sleeping behaviour were inconsistent during this time of adjustment, the target attribute for this time was also subject to noise. Additionally, not all cases of when two rowers were sleeping were available in the data set until point 6, as indicated in the dip in accuracy before that point. The dips in recall before point 6 in both the JRip and J48 algorithms may also be subject to this noise due to inconsistent behaviour.

The sudden performance gain from point 4 to 6 coincides with a period of *On Sea Anchor* context situations where it is likely that the rowers would have been sleeping in irregular shifts. This would make it easier for the JRip and J48 algorithms to capture all the possible combinations of the *Two Rowers are Sleeping* context situation for CR2 before the rowing partner changes (see Table 5.10) made the irregular combinations normal behaviour.

The recall at point 7 for the active/triggered state for the JRip algorithm was negatively impacted by the Airmar sensor loss at that point, but recovered performance by point 8. This would again indicate that a 'settling time' should occur before re-classification after the loss in sensor data. The J48 algorithm did not suffer the same drop in performance at point 7. The reintroduction of the Airmar sensor at point 13 did not have any impact on the performance of the context classifiers produced by either algorithm at that point. The loss of the biometric sensor data from *Rower 3* did, however, have a negative impact on the performance of the classifier produced by the JRip algorithm at point 20. This negative performance continued and did not improve over time. This would indicate that the context classifier for CR2 should not be re-created by the JRip algorithm after point 19. The classifiers produced by the J48 algorithm did not suffer the same drop in performance after the biometric data from *Rower 3* was lost.

Overall performance between the JRip and J48 algorithms were in favour with the J48 algorithm in every case for CR2.

Results for Requirement CR3 (1-4)

The target attributes for CR3 for all four rowers was defined through a combination of visual inspection (the complement of the information from CR1 and CR4), and a threshold value based on the *Mental Fatigue* sensor from each rower. The active/inactive state occurred in tightly grouped clusters at the beginning of the data set for CR3-1 to CR3-4, and also later in the voyage for CR3-1 and CR3-2. Although the biometric sensor for one of the rowers failed near the end of the row, it was known from interviewing the rowers that the *Mental Fatigue* for that rower never crossed the threshold again, so there were no rows of data where the state was unknown for this requirement (unlike CR2 and CR5).

Performance in accuracy and the precision and recall for the inactive/not triggered state is consistently high for CR3-1 to R3-4 for both the J48 and JRip algorithms. The precision and recall for the active/triggered state, however, varies quite drastically between the two algorithms on the four requirements. While the precision using JRip is generally lower than the J48 algorithm, particularly for R3-2, the recall is generally higher, particularly in the case of R3-1. This indicates that while the JRip algorithm generally produces a greater number of false positives than the J48 algorithm over the four requirements, it produces a lower number of false negatives.

Significant dips in recall performance for R3-2 and R3-4 occur when the rowers cross the fatigue threshold and are in what is considered to be one of the two *On Sea Anchor* states. This is similar to the confusion the algorithms had in the case of CR4 in discerning the differences between the active/triggered and inactive/not triggered states. The addition of new active/triggered state data coincides with drops in the recall performance in several cases, but like CR1 and CR4, this is generally fairly quickly corrected by the algorithms in the following training sets.

Interestingly, the sensor changes did not seem to have an observable impact on the performance of the JRip and J48 algorithms on R3-1 to R3-4. It is unknown what impact rowing shift changes had on the performance for the same reasons as CR1.

Results for Requirement CR4

The target attribute for CR4 was visually defined in a similar manner to CR1. The active/triggered state occurred in clustered groupings shown in Table 5.3, and more instances of the *On Sea Anchor Active* state were added to the training set as the time-series analysis progressed through the runtime data set. There were no unknown classes in the training and test data sets for this requirement.

Accuracy of the JRip and J48 algorithms was fairly similar, however, the precision and recall of the active/triggered state for the JRip algorithm performed slightly better than the J48. The precision and recall of the inactive/not triggered state was also fairly similar.

Results were similar to CR1 in that new *On Sea Anchor Active* active/triggered state data added to the training set had an immediate impact on performance. In CR4, this was shown as a dip in performance at point 6 and point 18 for the inactive/not triggered state, but an overall improvement in the active/triggered state. While these points indicate an improvement in recall for the active/triggered state, there is a decline in recall for the inactive/not triggered state. This indicates while more of the active/triggered state context is being correctly identified (fewer false negatives), less of the inactive/not triggered state is (more false negatives). Therefore, there is some overlap in the relevant context of the active/triggered and inactive/not triggered states such that the algorithms have a difficult time distinguishing between the two. This was also shown in the relatively high precision for the inactive/not triggered state, and the low precision for the active/triggered state. The algorithms simply had a difficult time distinguishing the active/triggered state from the inactive/not triggered state.

Interestingly, the sensor changes did not seem to have an observable impact on the performance of the JRip and J48 algorithms on CR4. It is unknown what impact rowing shift changes had on the performance for the same reasons as CR1.

Results for Requirement CR5

The target attribute for CR5 was also defined by a function based on the *Asleep or Awake* sensors from all 4 rowers. The active/inactive state also occurred fairly regularly throughout the entire set of data. There were 9854 rows of data where the active/triggered and inactive/not triggered class could not be defined for the target attribute by the functions (refer to Section 5.2.3 for details).

The results for CR5 were subject to a similar adjustment period as CR2 before the rowers' behaviour became more consistent. Likewise, the performance of accuracy, precision, and recall for CR5 was subject to similar impacts because the target attribute for CR5 was based on the same sensors as CR2. The noise introduced into the system by inconsistent rower behaviour is particularly pronounced in the recall graph for the active/triggered state for the JRip algorithm: the recall reaches 1 by the second point, but it quickly falls away and does not fully recover until point 10. The corresponding precision of the active/triggered context situation for the JRip algorithm, also does not reach similarly high levels until point 10. The J48 algorithm reaches consistently high levels of accuracy, precision, and recall much sooner at point 7.

The Airmar sensor loss or gain does not appear to have an impact on the performance of CR5 on the context classifiers produced by either the JRip or the J48 algorithms. The loss of the biometric data from *Rower 3*, however, definitely has an impact on the recall of the inactive/not triggered situation for CR5 for the classifiers produced by the JRip algorithm at that point (point 20). However, the performance drop has been recovered by point 22. It is unknown what impact the loss of biometric sensors from *Rower 3* would have on the active/triggered state because, as discussed in Section 5.2.3, the instances of the active/triggered context situation for CR5 could not be derived by the function after that point.

The context classifiers produced by the J48 algorithm consistently outperform the JRip algorithms for CR5.

5.3.3 Summary of Empirical Insights

We can see from Figures 5.10 to 5.13 how the JRip and J48 algorithms would have predicted the active/triggered and inactive/not triggered context states for each requirement during the OAR Northwest Dakar to Miami voyage had the algorithms been applied to produce new context classifiers every three days plus times of significant sensor configuration changes. The performance of the JRip and J48 algorithms was fairly similar, with slight improvements or losses depending on the individual requirement. The J48 algorithm did, however, perform better overall.

The results for both the JRip and J48 algorithms show that the overall accuracy for all five requirements was consistently high with the precision and recall for the inactive/not triggered state also high. The results for the active/triggered state varied greatly. Generally low precision for CR1 and CR4 (visually defined target attributes) could be observed with steadily improving recall for CR1, but generally low recall for CR4. The precision for CR2 (target attribute defined by function) was almost immediately high, with the recall taking more time to settle at high rates. The precision and recall for CR5 (target attribute also defined by function) generally showed a steady improvement with high performance being achieved for both. CR3-1 to CR3-4 (visually and threshold-defined target attribute) all generally showed quick rise time to high performance for both precision and recall with the exception of the precision for CR3-3.

Sensor configuration changes had an impact on the performance of CR1, CR2, and CR5, however, they did not appear to have as much impact as expected on performance. The re-introduction of old sensor data at point 13 for CR1 in the JRip results from Figure 5.10 had a negative impact, as did the loss of biometric sensor data in the case of CR2 and CR5. However, the classifiers produced immediately prior to these losses appeared adequate for future instances. This may indicate that the context classifiers produced by the J48 and JRip algorithms before a sensor loss are robust enough in our case to cope with significant sensor configuration changes after a sensor loss. The same appeared to be true for rowing partner changes, although this may be misleading because of how closely the rowing partner changes coincided with other significant context impacts to classifier performance. However, it should be noted that the lack of observable impacts by both changes in sensor configuration and rowing partner changes may simply be due to the fact that there are not enough data points in the graphs, and finer granularity is required to observe them.

While sensor configuration changes had less of a negative impact than expected, the addition of new active/triggered context state data for several requirements had more. Several significant drops in performance coincided with the addition of new active/triggered context situation data for several requirements (CR1, CR3-2, CR3-4, CR4), although performance was generally regained after the next context classifier was generated. Additionally, several gains occurred immediately after new active/triggered context situation data occurred (CR2, CR6). This would seem to indicate that new classifiers should be generated after the active/triggered state occurs, however, a settling time before doing so may be appropriate in order to account for noisy data.

5.3.4 Insights into Data Mining Approach from the Time-Series Analysis on Runtime Data

Declines in the performance of the algorithms shown in Figures 5.10 to 5.13 indicate significant changes in the context that defines the active/triggered and/or inactive/not triggered states for a given requirement. The descending slope before the local minimums indicates that the previous context data for the requirement is becoming less relevant to the future context. The slope of the ascent after a minimum indicates how quickly the algorithm is able to define new, relevant context for the future data. The greater the slope, the more quickly the algorithm can define new, relevant context rules for the requirement, and thus provide context awareness for that requirement more effectively.

Surprisingly, sensor configuration changes did not have as much impact on the performance of the context classifiers as expected. However, the introduction of new active/triggered state data for a given requirement had a much greater impact than anticipated. While it was expected that new data such as this would cause an immediate increase in performance, often the opposite often happened. However, these dips in performance were often immediately followed by a sharp rise in performance again. This might indicate that a 'settling time' after receiving new active/triggered state data may be desirable before generating a new context classifier. Additionally, a sliding window for the reintroduction of old sensor data after a sensor comes back online may be desirable so as to reduce noise, as in the case of the reintroduction of Airmar data for CR1.

Although the target attributes for CR1 and CR4 were visually defined in very similar ways, the performance of the precision and recall was significantly lower for CR4. Although this could be due, at least in part, to the fact that CR1 had approximately twice the active/triggered state data that CR4 did (see Table 5.9), it is more likely due to the fact that the algorithms simply could not adequately discern differences between the active/triggered and the inactive/not triggered states, as was described in the previous section.

5.4 Limitations

This thesis is an empirical investigation into integrating data mining algorithms into self-adaptive systems for context awareness and context evolution. As such, the research approach as covered in Chapter 3 was arrived at iteratively through the evaluation methodology described in Chapter 4. As insights were gained through the evaluation process, the approach was refined.

Internal Validity

Effort was taken to maximize the internal validity of each of the case studies and the time-series analysis from Chapter 5. This was completed in the exploratory case study by pre-processing the data, and then systematically applying the data mining algorithms to two subsets of that data while systematically reducing the number of columns of data in those subsets. Internal validity was further improved in the confirmatory case study by again pre-processing the data, and then reducing the independent variable to the indicator attributes for each requirement. It should be noted that CR2 and CR5 had slightly smaller data sets due to the number of rows of data where it was unknown whether they were in the active/triggered state or inactive/not triggered state (see Table 5.9). The target attributes were again examined in the time-series analysis where the number of rows of data in the training and test sets were incrementally changed.

In an effort to produce more representative results, the target attributes for the confirmatory case study and the time-series analysis were derived in three different ways, with at least two target attributes resulting from each of the three ways (see Section 5.2.3).

Significant effort was made to validate and verify the target attributes for each of the requirements (described in Sections 5.1.3, and 5.2.3) through examining the sensor data, examining the daily logs, and in interviews with the rowers. However, it was not possible to determine the target attributes' actual correlation to the context each of the requirements was active/triggered and inactive/not triggered in. Regardless, the ultimate goal of this thesis is the investigation of how context-aware systems can adapt to changes in when requirements are typically active/triggered and inactive/not triggered. That is, one of the assumptions is that the target attribute for a given requirement will be subject to change over time, regardless of how it was initially defined. Even so, an attempt was made to minimize the impact of erroneous target attributes on the analyses by completing them on multiple requirements in the exploratory case study in Section 5.2 and in the time-series analysis in Section 5.3.

External Validity

The settings of our case studies were almost experimental as the rowers operated under normal conditions in very isolated and physically extreme conditions with a small number of external impacts. Given this, there may be some question as to how the results may change in a more subtle environment with less clear differentiation between active/triggered and inactive/not triggered states for each requirement (see *Results for Requirement CR4* in Section 5.3.2). Nonetheless, the runtime data used in the studies in this thesis are from real situations that took place in different locations and times, with different users, and different levels of adherence to user goals.

Additionally, while the users were operating in a restricted environment with similar activities being completed consistently over time, the results may also be applicable to other contexts with consistent patterns of activity. These might include other contexts that rely on shift work (such as hospital wards, or factories), teams of athletes, or even military personnel. Alternatively, given the results in the exploratory case study in Section 5.1, the results may also be applicable to individuals acting with consistent patterns of activity.

Ecological Validity

The investigation and analysis described in this thesis was was carried out on two laptops using Weka 3.6.9, as implementation of the requirements from this thesis for context-awareness on a mobile device was out of the scope of this thesis. This means that scalability issues with memory and performance may come into play when performing the analysis from this thesis on a mobile device instead of a laptop. However, as described in Section 3.3, an effort was made to select well-known data mining algorithms appropriate for mobile implementation, and the J48 algorithm has recently been implemented on mobile using Weka. Additionally, high performance results were obtained in several of the requirements after only a few days of sensor data in the training sets (see Figures 5.10 to 5.13). Also, the noise that sensor data older than two weeks appeared to introduce into CR1 (see Section *Results for Requirement CR1*) may indicate that a sliding window of contextual sensor data may be enough, and even more desirable in some cases rather than a larger set of data.

Additionally, it is assumed that all disparate sensor sources can be preprocessed, and integrated centrally so that data mining algorithms can be applied. This was because of the isolated and unobservable setting that occurred in the case study where wireless cloud connectivity was prohibitively costly, thus necessitating mobile implementation of the data mining component of the context-aware system. While capturing all the relevant sensor data may not be possible in all cases, many mobile devices contain an array of on-board sensors, and there are also a number of bluetooth-enabled sensors (e.g., heart rate monitors, sleep tracking, movement sensors) that can currently be coupled with mobile devices.

Table 5.5: Functions (based on actual sensor values) used to derive target attribute classes for CR2 and CR5 from row 1 to row 79228 (*before* sensor loss from *Rower 3*) in data set. Note that the value for *Rower i Asleep or Awake* is either 1 (Awake) or 0 (Asleep).

Target Attribute Class	Function Target Attribute Class Derived From
CR2 active/triggered *(i.e., class = 1)*	(*Rower 1 Asleep or Awake* Sensor Value) + (*Rower 2 Asleep or Awake* Sensor Value) + (*Rower 3 Asleep or Awake* Sensor Value) + (*Rower 4 Asleep or Awake* Sensor Value)$<=$ 2
CR2 inactive/not triggered *(i.e., class = 0)*	(*Rower 1 Asleep or Awake* Sensor Value) + (*Rower 2 Asleep or Awake* Sensor Value) + (*Rower 3 Asleep or Awake* Sensor Value) + (*Rower 4 Asleep or Awake* Sensor Value)$>$ 2
CR5 active/triggered *(i.e., class = 1)*	(*Rower 1 Asleep or Awake* Sensor Value) + (*Rower 2 Asleep or Awake* Sensor Value) + (*Rower 3 Asleep or Awake* Sensor Value) + (*Rower 4 Asleep or Awake* Sensor Value)$=$ 3
CR5 inactive/not triggered *(i.e., class = 0)*	(*Rower 1 Asleep or Awake* Sensor Value) + (*Rower 2 Asleep or Awake* Sensor Value) + (*Rower 3 Asleep or Awake* Sensor Value) + (*Rower 4 Asleep or Awake* Sensor Value)$\neq$ 3

Table 5.6: Functions (based on actual sensor values) used to derive target attribute classes for CR2 and CR5 from row 79229 to row 90748 (*after* sensor loss from *Rower 3*) in data set. Note that the value for *Rower i Asleep or Awake* is either 1 (Awake) or 0 (Asleep).

Target Attribute Class	Function Target Attribute Class Derived From
CR2 active/triggered *(i.e., class = 1)*	(*Rower 1 Asleep or Awake* Sensor Value) + (*Rower 2 Asleep or Awake* Sensor Value) + (*Rower 4 Asleep or Awake* Sensor Value)<= 1
CR2 inactive/not triggered *(i.e., class = 0)*	(*Rower 1 Asleep or Awake* Sensor Value) + (*Rower 2 Asleep or Awake* Sensor Value) + (*Rower 4 Asleep or Awake* Sensor Value) = 3
CR2 unknown *(i.e., class = ?)*	(*Rower 1 Asleep or Awake* Sensor Value) + (*Rower 2 Asleep or Awake* Sensor Value) + (*Rower 4 Asleep or Awake* Sensor Value) = 2
CR5 inactive/not triggered *(i.e., class = 0)*	(*Rower 1 Asleep or Awake* Sensor Value) + (*Rower 2 Asleep or Awake* Sensor Value) + (*Rower 4 Asleep or Awake* Sensor Value)<= 1
CR5 unknown *(i.e., class = ?)*	(*Rower 1 Asleep or Awake* Sensor Value) + (*Rower 2 Asleep or Awake* Sensor Value) + (*Rower 4 Asleep or Awake* Sensor Value)>= 2

Table 5.7: JRip Algorithm Stratified 10-Fold Cross Validation Results for all Requirements in Confirmatory Case Study

Req't	Accuracy %	FP Rate	Precision	Recall
CR1	99.91	0	0.995	0.995
CR2	99.05	0.007	0.978	0.983
CR3-1	99.99	0	0.999	1
CR3-2	99.89	0.001	0.998	0.996
CR3-3	99.91	0	0.989	0.985
CR3-4	99.87	0.001	0.997	0.997
CR4	99.94	0	0.994	0.992
CR5	95.8	0.012	0.938	0.823

Table 5.8: J48 Algorithm Stratified 10-Fold Cross Validation Results for all Requirements in Confirmatory Case Study

Req't	Accuracy %	FP Rate	Precision	Recall
CR1	99.87	0.001	0.995	0.992
CR2	99.29	0.006	0.981	0.989
CR3-1	99.98	0	0.999	1
CR3-2	99.91	0.001	0.996	0.999
CR3-3	99.93	0	0.992	0.987
CR3-4	99.89	0.001	0.996	0.999
CR4	99.95	0	0.997	0.992
CR5	98.01	.01	0.955	0.934

Table 5.9: Number of active/triggered state, inactive/not triggered state, and unknown state rows in each of the requirements examined for a total of 90748* rows in the time-series analysis (and confirmatory case study) data set.

Req't	Active/Triggered State	Inactive/Not Triggered State	Unknown State
CR1	8190	82559	0
CR2	21270	66471	3008
CR3-1	8531	82218	0
CR3-2	16066	74683	0
CR3-3	3190	87559	0
CR3-4	19908	70841	0
CR4	4005	86744	0
CR5	14561	66334	9854

**note that a small boundary error in the script used to divide the test and training sets resulted in an overlap between the sets of one row*

Table 5.10: Context Changes Visually Observable in Confirmatory Case Study Runtime Data

Rowing Partner Changes	Sensor Configuration Changes
Feb 12	Feb 11
Mar 14	Feb 26
	Mar 19

Chapter 6

Conclusion

This section discusses how the evaluation methodology described in Chapter 4, carried out as described in Chapter 5 fulfilled the purpose of investigating how passively collected sensor data, and actively collected log data could be leveraged for the requirements elicitation process. Also discussed are some of the ways that this study fits into current work in context awareness, context evolution, and mobile systems.

6.1 Addressing Research Objectives

As presented in the results of the exploratory (Section 5.1) and confirmatory (Section 5.2) case studies, the data mining approach presented in Chapter 3 was shown to be useful for better understanding unobservable system operating environments to support the requirements elicitation process.

Context for the active/triggered and inactive/not triggered states for one requirement was defined using data mining algorithms applied to the contextual sensor data sets in the exploratory case study, and five requirements in the confirmatory case study (requirement CR3 had four results produced, one for each of the users, for a total of eight results). Various algorithms appropriate to requirements engineering and mobile device implementation were applied in the exploratory case study (see Section 3.4 and Section 5.1.4), with the JRip and J48 algorithms being applied for extensive evaluation in the confirmatory case study (see Section 5.2.4).

In conclusion, there were several implications for using data mining for Requirements Engineering, for Context-Awareness for Mobile Applications, and for Group-Context-Aware Mobile Applications. These are described in the following sections.

6.2 Data Mining and Requirements Engineering

There were several influences on the results. Data pre-processing proved to have a significant impact, particularly data reduction (see Section 3.2 and 5.1.7). Additionally, the characteristics of the data set including how consistent the data collection was, how much data was available, how representative the training sets were of the test sets, and how consistent the user goals were over the period the sensor data was collected all impacted the results (Sections 3.3, 5.1.3, 5.2.3), with more generally producing more accurate results.

The separation between the active/triggered state and the inactive/not triggered state for each requirement, not only in the target attribute, but also in the contextual sensor data appears to have an impact on the results (see *Results for Requirement CR1* in Section 5.3.2). Additionally, the target attribute was not captured by the system at runtime and was, instead defined, verified, and validated in one of three different ways, depending on the requirement (see Sections 3.3, 5.1.3, and 5.2.3). The possible discrepancy between the target attributes derived in this study and target attributes that would have been captured at runtime is unknown. Therefore, the impact that this discrepancy would have on the results is also unknown.

The results of the time-series runtime analysis in Section 5.3 were used to address how much contextual sensor data needed to be collected in order to define the active/triggered and inactive/not triggered situations for each of the requirements. The time-series runtime analysis also addressed how context sensor configuration changes and other significant contextual changes impacted the performance of the context classifiers.

The reduction of physical contextual sensor data from the training sets (i.e., the Airmar sensor environmental data and the biometric data from *Rower 3*) didn't affect the results of the classifiers as much as expected. However, the addition of active/triggered state data was shown to have a loss of performance impact on the classifiers in several cases, which was unexpected. Concept drift may have been notably present in one case (CR1) upon the reintroduction of the Airmar sensor data into the training set after a period of approximately two weeks.

Distribution of the active/triggered state throughout the data, as well as how the target attribute was defined may have an impact on the performance of the classifiers. However, because the sample size of the analysis is limited to within eight results (five requirements, one with four sub-results), it cannot be said for certain.

6.3 Data Mining and Context-Awareness for Mobile Applications

As discussed in Section 2.2, data mining has shown to be useful for identifying situations for service delivery in context-aware applications. In this study, data mining was used on historic sensor data passively collected at runtime in order to define the active/triggered and inactive/not triggered situations for several requirements in a context-aware application. This was done to support the requirements elicitation process.

The (isolated) and dangerous nature of the system context of use made it impossible to perform recommended requirements elicitation techniques (e.g., [13]) in order to determine the contexts that the user groups might be situated in. Instead, passively collected runtime sensor data, daily log activities, and interviews with the rowers were relied upon for requirements elicitation. Data mining proved to be a useful means of identifying subtle context of use situations for context-aware service delivery for several requirements.

In addition, location was not considered as relevant to our context-aware application for the requirements we investigated because the locations were unique throughout the duration of the application's use in the contexts investigated (long distance, open-water rows). Nor does study take into account contextual information provided through wireless means (ubiquity) because the users were in an extremely isolated environment with wireless connectivity being prohibitively expensive (i.e., the Atlantic Ocean). As discussed in Section 2.2, there are many existing studies and context-aware applications centred on leveraging location context for mobile ubiquitous systems. This study gives an example of a context-aware application that is not only primarily dependent non-location sensor data alone, but also gives a real example of an application where the context-aware mobile system cannot leverage wireless connections.

As discussed in Section 2.2, several studies feature actigraphy (motion sensors) as a useful physical and user context for context-aware applications. This study centred around biometric sensor data (physical and user context) from each of the four rowers in the confirmatory case study, as well as physical contextual data from two environmental sensors. While the environmental sensor data did appear in many of the classifiers produced, far more important in the ranking was the sensor data obtained directly from the users. This is interesting because although the weather may be one of the causes of the context-aware service being delivered, the physical effect it had on the rowers appeared to be far more relevant to the algorithm's definition of the classifier. This was demonstrated in the case of CR1 where high wind and waves influenced when the rowers were *On Sea Anchor*, but the most important rule in one of the the classifiers (i.e., those with the most coverage), had to do with when three of the rowers were described as being *In Bed* by their biometric sensors. This confirms the importance of biometric sensors for individual users, and may be an indicator that they are more important than environmental sensors to deriving situations for context-aware applications, even when the environment is the cause of the current context situation. It is known that processing data on resource-constrained devices such as mobile phones is challenging [11]. By prioritizing the sensors that are monitored (e.g., choosing to monitor the biometric sensors and potentially dropping the environmental ones entirely), resources on mobile devices can be better optimized.

6.4 Group-Context-Aware Mobile Applications

In addition, the context situations investigated were complex and unique, often depending on the data from more than one user at a time (see Section 5.2). Many previous studies rely on defining context situations for application service delivery based on movement activities from a single user, rather than a group of users. While this study does, indeed take into account biometric information from a single user in several cases (i.e., the *On Sea Anchor* requirement from Section 5.1, and requirements CR3-1 to CR3-4 from Section 5.2), there are also four requirements for which situations for the entire group is considered (i.e., CR1, CR2, CR4, and CR5 in Section 5.2).

Group context-aware mobile applications are an emerging area of interest with implications for emergency first responders, teams, and military field personnel in order to help them complete missions [21, 22]. Two of the requirements in this study were directly based on hostile environmental conditions (CR1 and CR5), and one was based on the users' physical and mental condition in that hostile environmental condition (CR3-1 to CR3-4). The study presented in this thesis could have implications for this domain, particularly since the context of use in this study is on the "tactical edge" (in a resource-limited and hostile environment), as are many of the applications in this emerging field.

Bibliography

[1] Leon Barnard, Ji Soo Yi, Julie A Jacko, and Andrew Sears. An empirical comparison of use-in-motion evaluation scenarios for mobile computing devices. *International Journal of Human-Computer Studies*, 62(4):487–520, 2005.

[2] Jenna Burrell and Geri K Gay. E-graffiti: evaluating real-world use of a context-aware system. *Interacting with Computers*, 14(4):301–312, 2002.

[3] Galle Calvary, Jolle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt. A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3):289–308, 2003.

[4] Betty H.C. Cheng, Rogério Lemos, Holger Giese, Paola Inverardi, Jeff Magee, Jesper Andersson, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, Giovanna Marzo Serugendo, Schahram Dustdar, Anthony Finkelstein, Cristina Gacek, Kurt Geihs, Vincenzo Grassi, Gabor Karsai, HolgerM. Kienle, Jeff Kramer, Marin Litoiu, Sam Malek, Raffaela Mirandola, Hausi A. Müller, Sooyong Park, Mary Shaw, Matthias Tichy, Massimo Tivoli, Danny Weyns, and Jon Whittle. Software engineering for self-adaptive systems: A research roadmap. In BettyH.C. Cheng, Rogrio Lemos, Holger Giese, Paola Inverardi, and Jeff Magee, editors, *Software Engineering for Self-Adaptive Systems*, volume 5525 of *Lecture Notes in Computer Science*, pages 1–26. Springer Berlin Heidelberg, 2009.

[5] William W. Cohen. Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[6] Joëlle Coutaz, James L. Crowley, Simon Dobson, and David Garlan. Context is key. *Commun. ACM*, 48(3):49–53, March 2005.

[7] Anind K. Dey, Gregory D. Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Hum.-Comput. Interact.*, 16(2):97–166, December 2001.

[8] Nathan Eagle and Alex (Sandy) Pentland. Reality mining: Sensing complex social systems. *Personal Ubiquitous Comput.*, 10(4):255–268, March 2006.

[9] Anthony Finkelstein and Andrea Savigni. A framework for requirements engineering for context-aware services. In *First International Workshop From Software Requirements to Architectures (STRAW 01) 23rd International Conference on Software Engineering*. IEEE Computer Society Press, 2001.

[10] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: A review. *SIGMOD Rec.*, 34(2):18–26, June 2005.

[11] Pari Delir Haghighi, Shonali Krishnaswamy, Arkady Zaslavsky, Mohamed Medhat Gaber, Abhijat Sinha, and Brett Gillick. Open mobile miner: A toolkit for building situation-aware data mining applications. *Journal of Organizational Computing and Electronic Commerce*, 23(3):224–248, 2013.

[12] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[13] Dan Hong, Dickson K. W. Chiu, and Vincent Y. Shen. Requirements elicitation for the design of context-aware applications in a ubiquitous environment. In *Proceedings of the 7th International Conference on Electronic Commerce*, ICEC '05, pages 590–596, New York, NY, USA, 2005. ACM.

[14] Dan Hong, DicksonK.W. Chiu, VincentY. Shen, S.C. Cheung, and Eleanna Kafeza. Ubiquitous enterprise service adaptations based on contextual user behavior. *Information Systems Frontiers*, 9(4):343–358, 2007.

[15] Jason I. Hong and James A. Landay. An architecture for privacy-sensitive ubiquitous computing. In *Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services*, MobiSys '04, pages 177–189, New York, NY, USA, 2004. ACM.

[16] Eija Kaasinen. User needs for location-aware mobile services. *Personal Ubiquitous Comput.*, 7(1):70–79, May 2003.

[17] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.

[18] John Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.

[19] N.D. Lane, E. Miluzzo, Hong Lu, D. Peebles, T. Choudhury, and A.T. Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, Sept 2010.

[20] Sunyoung Lee and Kun Chang Lee. Context-prediction performance by a dynamic bayesian network: Emphasis on location prediction in ubiquitous decision support environment. *Expert Systems with Applications*, 39(5):4908 – 4914, 2012.

[21] Grace Lewis, Marc Novakouski, and Enrique Snchez. A reference architecture for group-context-aware mobile applications. In David Uhler, Khanjan Mehta, and JenniferL. Wong, editors, *Mobile Computing, Applications, and Services*, volume 110 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 44–63. Springer Berlin Heidelberg, 2013.

[22] Grace A Lewis, Soumya Simanta, Marc Novakouski, Gene Cahill, Jeff Boleng, Edwin Morris, and James Root. Architecture patterns for mobile systems in resource-constrained environments. In *Military Communications Conference, MILCOM 2013-2013 IEEE*, pages 680–685. IEEE, 2013.

[23] Chang Liu, Qing Zhu, Kenneth A. Holroyd, and Elizabeth K. Seng. Status and trends of mobile-health applications for ios devices: A developer's perspective. *Journal of Systems and Software*, 84(11):2022 – 2033, 2011. Mobile Applications: Status and Trends.

[24] Raymond W. Matthews, Sally A. Ferguson, Xuan Zhou, Anastasi Kosmadopoulos, David J. Kennaway, and Gregory D. Roach. Simulated driving under the influence of extended wake, time of day and sleep restriction. *Accident Analysis*

and Prevention, 45, Supplement(0):55 – 61, 2012. Managing Fatigue in Transportation, Resources and Health8th International ConferenceFremantle, Western Australia21-24 March 2011.

[25] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen. Activity classification using realistic data from wearable sensors. *Information Technology in Biomedicine, IEEE Transactions on*, 10(1):119–128, Jan 2006.

[26] John Ross Quinlan. *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann, 1993.

[27] C A Russell, J A Caldwell, D Arand, L J Myers, P Wubbels, and H Downs. Validation of the fatigue science readiband. `http://fatiguescience.com/wp-content/uploads/2014/02/Readiband-Validation.pdf`, March 2014.

[28] Charles Samuels. Sleep, Recovery, and Performance: The New Frontier in High-Performance Athletics. *Physical medicine and rehabilitation clinics of North America*, 20(1):149–159, February 2009.

[29] Approved September. Ieee standard glossary of software engineering terminology. *Office*, 121990(1):1, 1990.

[30] P.-N. Tan, V. Kumar, and M. Steinbach. *Introduction to Data Mining*. Pearson Publishing, 2005.

[31] NorhaM. Villegas, Gabriel Tamura, HausiA. Müller, Laurence Duchien, and Rubby Casallas. Dynamico: A reference model for governing control objectives and context relevance in self-adaptive software systems. In Rogério Lemos, Holger Giese, Hausi A. Müller, and Mary Shaw, editors, *Software Engineering for Self-Adaptive Systems II*, volume 7475 of *Lecture Notes in Computer Science*, pages 265–293. Springer Berlin Heidelberg, 2013.

[32] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 226–235, New York, NY, USA, 2003. ACM.

[33] S. M. Weiss and C. A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. M. Kaufmann Publishers, San Mateo, USA, 1991.

[34] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[35] Shichao Zhang, Chengqi Zhang, and Qiang Yang. Data preparation for data mining. *Applied Artificial Intelligence*, 17(5-6):375–381, 2003.