Integrating Data Mining into Feedback Loops for Predictive Context Adaptation

Angela Rook, Alessia Knauss,
Daniela Damian, Hausi A. Müller, Alex Thomo

Dept of Computer Science, University of Victoria, Canada
{arook}@uvic.ca, {alessiak, danielad, hausi, thomo}@cs.uvic.ca

Abstract

Requirements for today's systems are increasingly valid only within certain operating contexts. Requirements engineering and implementation stages of system development must carefully consider how to integrate evolving context related to specific requirements in order for the system to stay relevant and flexible. In this paper we propose to use data mining techniques for predictive context adaptation. Our approach leverages data collected from the past and decides, based on this historical data, which context conditions to monitor in order to predictively identify when a system needs to be adapted to fulfill a particular requirement. We demonstrate our approach on an adaptive mobile application to support the coordination of a team of rowers in an environment with a continually changing operational context.

1 Introduction

Software systems are highly integrated and ubiquitous in a way that goes beyond the device the software is running on. Many software systems are web-based, but developers of those systems also try to make integration with mobile devices a top priority. Such systems need to be updated frequently, and this adaptation creates a workload challenge for software engineers to adapt to the systems' rapidly changing operating context. Therefore, researchers develop techniques for self-management and self-adaptation of the software itself, so that the amount of work put in by actual humans is reduced. Such systems are then capable of automatically adapting to the current operating context.

Adaptive systems possess the ability to decide how to self-configure to best achieve their goals based on the conditions that they sense in their environment by using feedback loops [2]. If the system does not update its knowledge about the changing context required for specific system goals, those goals may not be achieved because the system is unable to recognize that the operating context has changed.

Current approaches either use a predefined static set of context conditions that the system monitors, or allow users to manually add new separate context conditions and define their conditions for adaptation, e.g., [5]. Nevertheless, to the best of our knowledge, there are no approaches proposed that by integrating context adaptivity into a system, the system itself can decide at runtime which sensors (*context attributes*) to use for context monitoring – considering internal and external context – in order to dynamically adapt to better meet system goals.

For example, for the time adjustment a mo-

bile phone app can use the GPS system for correct identification of the exact time for the location the phone is currently at. If no GPS is installed on the phone, the app would need to use other context (consisting of different context attributes, e.g., the last known location and the distance travelled assuming other installed sensors). Which context attributes are available at runtime is unknown and dynamic due to things like sensor failure or different configurations. The system decides, based on the current situation and past collected data, by using data mining to show the best possible composition of context attributes for the current state.

In this paper we propose to use *data mining* techniques in feedback loops to support predictive context adaptation. We propose an approach that leverages data collected in the past to decide what context attributes and their conditions to monitor in order to identify the state in which a particular requirement needs to be satisfied.

We evaluate our approach with a prototype, a **T**ask **o**n **T**ime **E**xecution **M**anager (ToTEM) that we develop to support the coordination of a team of four rowers of OAR Northwest during extreme rowing expeditions.[1] We select one particular requirement for adaptation and show how data mining helps in identifying relevant context-attributes and their values (conditions) that the system needs to monitor in order to fulfill this particular requirement. We validated our approach with the rowing team.

2 ToTEM Architecture

2.1 An Adaptive Scheduler

We designed ToTEM, an adaptive scheduling system, for four rowers from OAR Northwest. OAR Northwest is a non-profit organization whose members undertake long distance rowing voyages for public education and research.

ToTEM provides automated task management for the rowers for their rigid activity schedule they need to adhere to during their trips to ensure, for example, their circadian biorhythms [7]. This implies the need to minimize the impact of sudden time zone changes. ToTEM adapts by automatically updating the incrementally derived 'boat time', and to alert the rowers when to complete their tasks (e.g., rowing shift changes, sleep shifts, or when to complete research tasks).

While it is important that the system keeps them on track, it is also important that the system does not annoy them in situations where ToTEM is not needed. In an interview the rowers expressed that it is important that ToTEM stops alerting them about scheduled activities during the time that they are on sea anchor (i.e., stopped unexpectedly because conditions do not allow them to continue rowing). Therefore, the case study reported on in this paper is based on the following particular requirement for which we show its adaptation:

Switch all alerts off when on sea anchor or provide an option to switch everything off.

Our main goal is the design of runtime context adaptivity in ToTEM as shown on our example as ToTEM's functionality is important under certain conditions, but not at all important under others.

We are evaluating our data mining techniques with contextual data collected during a rowing trip around Vancouver Island, British Columbia. The data was collected from April 11 to May 8, 2012 via on-board biometric and environmental sensors. These sensors included a high-sensitivity (-160dBm) GPS tracking system and an athlete data management software system with sensors attached directly to one of the rowers. The final list of sensor data (context attributes) that were used in our data mining analysis for the selected requirement 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,

[1] http://oarnorthwest.com/

- CEP (unknown sensor attribute),
- Temperature,
- Actigraphy (vigorousness of movement),
- In Bed or Not,
- Asleep or Awake, and
- Effectiveness (relates to fatigue levels).

The data from these sensors allowed us to infer when the boat was stopped. This was the correlation we used to determine when the rowers were on sea anchor.

2.2 Recognizing Evolving Context

System requirements sometimes depend on the identification of subtly changing patterns in contextual situations from enormous data sets. For example, it would take hours, or even days for a human to analyze even the relatively small data set used in this study, and derive useful results. It is often infeasible for human analysts to interpret the vast amounts of contextual data produced by mobile and other ubiquitous systems for context-based adaptation. More efficient, automatic means of context analysis need to be implemented in order for systems to adapt to changing contexts effectively.

Data mining is useful for quickly and automatically detecting non-obvious patterns in data sets by statistically deriving classification 'rules' from them. These rules can be used to statistically determine how likely a particular outcome will be when predicting the future. We propose using rules produced automatically from contextual data sets through data mining techniques [11] to support system adaptation to changing system contexts. In that we mean that the rules produced by data mining algorithms show the important context-attributes and their conditions that describe the possible situations that the system needs to monitor for adaptation.

Villegas et al. propose to use historical information for *predictive* adaptation [12]. We propose supporting system context adaptation in a predictive manner by mining historical context data to generate classification rules to identify when a specific adaptation requirement needs to be fulfilled. In this paper, we demonstrate how data mining is used on incoming environmental and biometric sensor data to produce indicators and rules to predict when the rowers are on sea anchor in order to disable the ToTEM system alerts. These rules can then be applied to incoming sensor data to detect whether or not a particular context is relevant to a specific requirement. In the sea anchor example, the rules will recognize the sea anchor conditions to trigger the system to adapt appropriately (i.e., disable alerts).

3 Designing Feedback Loops

In our approach, we use sensor data to identify rules and algorithms that can predict when the boat stops. For validation, we use historical data and apply the algorithms to on a speed over ground boundary and measure the precision in the prediction.

An effective way of interpreting and responding to sensor data is through feedback loops. Feedback loops are used in a variety of engineering applications and in nature to monitor and regulate dynamic systems [4, 9]. Context adaptation in our system is triggered in the context monitoring feedback loop every time the rules for identifying when the boat is stopped are no longer adequate. At this point, historical context data is mined for new context rules for the requirement.

Applying Dynamico

The DYNAMICO reference model was used to engineer appropriate feedback loops for the new ToTEM *System Requirement* (i.e., *Switch all alerts off when on sea anchor or provide an option to switch everything off.*) to realize predictive adaptation. The DYNAMICO reference model is shown in Figure 1 [10, 12]. We use this model because it was developed for highly dynamic operating contexts, such as the one we developed ToTEM for. It separates feedback and adaptation concerns into three feedback loops for (1) adaptation of system requirements, (2) the dynamic behaviour of the adaptation mechanisms, and (3) the management of dynamic context.

The *Control Objectives Feedback Loop* makes sure the system is adequately meeting system

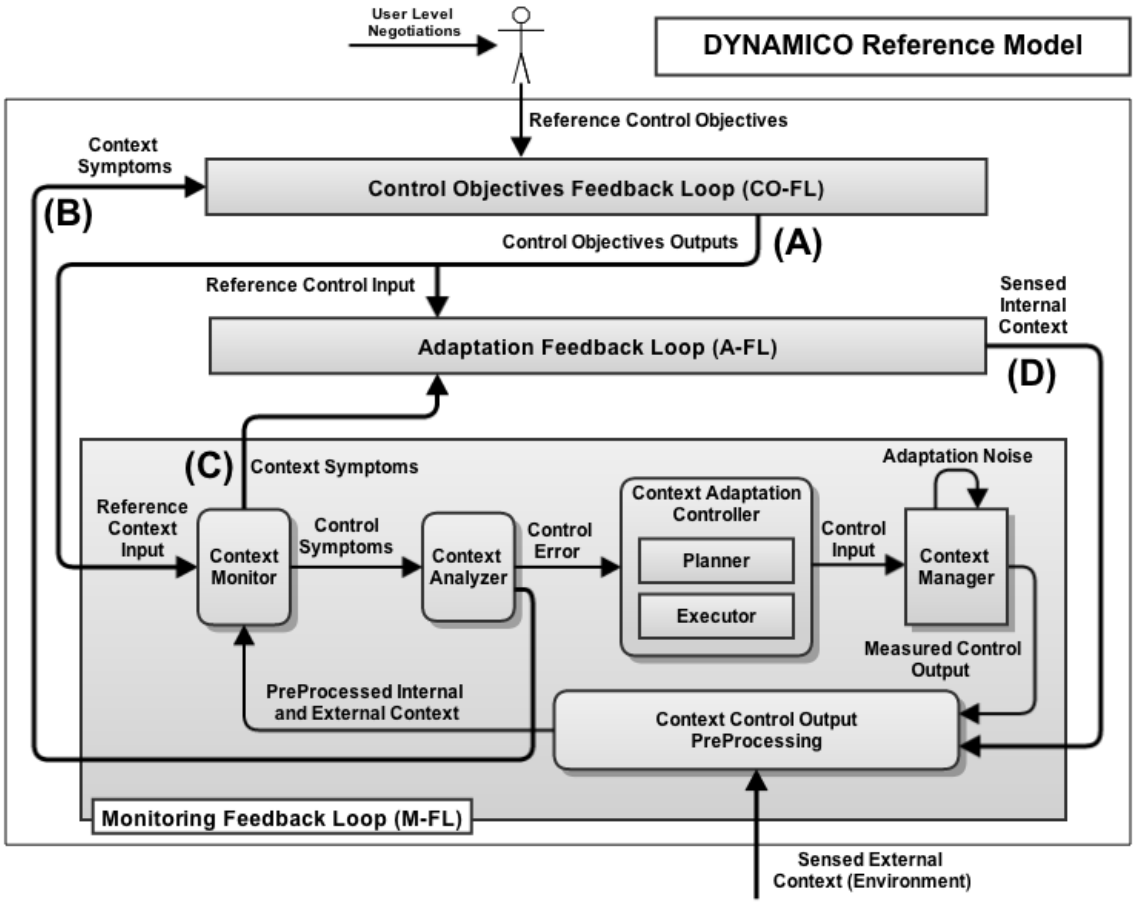

Figure 1: DYNAMICO reference model in detail [10, 12].

requirements in partnership with the *Monitoring Feedback Loop* and the *Adaptation Feedback Loop* [10, 12]. In the ToTEM system, the Control Objectives Feedback Loop is responsible for informing the rest of the system that alerts should be disabled when the rowers are on sea anchor. It then makes sure that the rest of the system fulfills the requirement adequately.

The *Monitoring Feedback Loop* is responsible for making sure that the system triggers context adaptation at the right time, and that it continues to do so over time [10, 12]. Context evolution for our system takes place in the Monitoring Feedback Loop. For example, when the system detects too many *false positives* (i.e., times when the system indicates that rowers are on sea anchor when they are not), a context evolution would take place in the Monitoring Feedback Loop.

The *Adaptation Feeback Loop* carries out actions to fulfill system requirements [10, 12]. In the ToTEM system, this means actually disabling system alerts when the system predicts that it is on sea anchor.

Moreover, we integrate a *Knowledge Base*, as a collective source of information pertaining to the adaptive system [3, 12]. It holds information such as sensor data, data mining rules derived for relevant feedback loops, data mining algorithms, and other dynamic information that the individual components of the system need to share.

All adaptations for the system, including predictive adaptations, are triggered by the *Context Monitor*. In order for the *Context Monitor* to recognize when to trigger these adaptations, it needs context sensor data. Sensor data needs to be prepared by the system

through *Context Control Output PreProcessing* (Section 3.1).

Through the *Context Monitor*, the *Context Analyzer* is provided with a set of potentially relevant context attributes to the on sea anchor context (Section 3.2.1) via *Interaction (A)*. It is also provided with acceptable performance guidelines for predictive adaptation from the *Context Monitor*. An example guideline is: *The system should correctly identify the on sea anchor state and adapt predictively to it at least 85% of the time with less than 10% false positives*. Based on the set of context attributes and the performance guidelines, the *Context Analyzer* decides which action the *Context Adaptation Controller* should take, including which data mining algorithms should be used (Section 3.2.2) to produce new rules for predictive adaptation for the on sea anchor context.

The *Context Adaptation Controller* produces the rules set from the algorithm(s) selected by the *Context Analyzer* (Sections 3.2.3, 3.2.4), and selects one based on performance and system goals (Section 3.2.5).

At this point the new adaptation rules for the requirement are saved in the *Context Manager* to be referenced by the *Context Monitor* for predictive adaptations as described above. The *Context Monitor* will also decide through the performance guidelines provided in *Interaction (A)* when context evolution needs to be performed on the context attribute rules produced for our on sea anchor requirement.

3.1 Preparing Sensor Data for Predictive Analytics

Significant preprocessing is required to prepare the sensor data from the Vancouver Island voyage for analytics:

3.1.1 Merging Context Data

UTC (Coordinated Universal Time) time 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 inconsistencies and corrected. Rounding of the 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 tuples. Tuples are groups of data taken from sensors at the same time, typically stored as rows in a table.

3.1.2 Cleaning Context Data

After merging the sensor data, all data was removed from sensors that would not contribute to the analysis of our requirement. These included columns consisting entirely of zeros, and those pertaining to the technical attributes of the GPS sensor, reading ID, and UTC. Columns with redundant attributes were also eliminated, with those demonstrating the highest precision being retained. Finally, the GPS Pacific Standard Time column was converted to normalized columns of days, hours, and minutes.

Tuples with data that fell outside of reasonable attribute ranges were removed, as were those with all-zero sensor entries and other sensor measurement errors. Moreover, some minor manual adjustments needed to be made on a few of the tuples by the rowers for the correction of some GPS values.

The result of this cleaning process reduced the total number of tuples from 1845 down to 1592 (i.e., 13.7% of "noisy" tuples were eliminated).

3.1.3 Consistency of Sensor Readings

The inconsistency of sensor readings as a potential indicator for abnormal user operating behaviour was considered. Figure 2 shows the tuple frequency of sensor readings by day. The readings were expected to be similar every day as the rowers were simulating a rigid and regulated environment. A "normal" measure of the frequency of tuples for a 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 obtained, we ran an evaluation on the days that fell outside of this normal frequency 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.

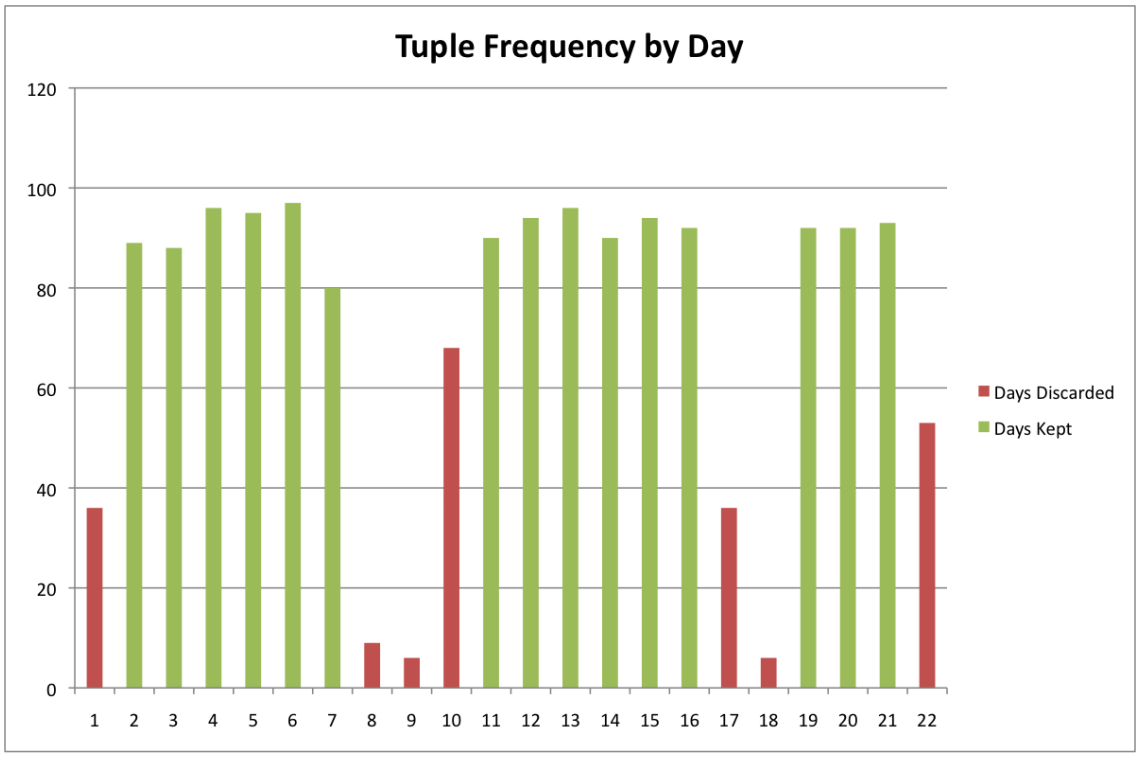

Figure 2: Tuple frequency by day - *Days Kept* indicate days that were kept for the data set Without Outlying Days, *Days Discarded* indicate outlying days that were included along with the green in the first data mining analysis, but discarded for the second

This sensor data pattern information could be integrated into the knowledge base, so that the next time a similar situation appears, the system could recognize it and adapt accordingly.

The tuples from days that did not fall within a tuple frequency range of 91 +/-4 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%).

3.1.4 Normalizing Context Data

As a final step before analytics we normalized the context data for each context attribute so that it fell within a range between 0 and 1. This transformation helps improve the performance of some data mining algorithms.

3.2 Predictive Adaptation for our Requirement

Generating new rules for predictive adaptation for the sea anchor requirement involved several sequential steps in the *Context Analyzer* and *Context Adaptation Controller*:

3.2.1 Determine Relevant Context-Attributes

Relevant sensors need to be selected to provide all the relevant information for the context-attributes available to the data mining algorithms to detect when the rowers were on sea anchor. These were all selected as potentially relevant information from context-attributes to be included in the first iteration of running the data mining algorithms on the historical context data as listed in Section 2.1. The full data set also contained oceanographic con-

text data, which was excluded (validated with rower's knowledge) based on their lack of relevance to the context surrounding when the rowers were on sea anchor.

3.2.2 Selecting Data Mining Algorithms

Based on system priorities (e.g., how much processing overhead the system can use), the *Context Analyzer* decides which data mining algorithm(s) to select as candidates for predictive adaptation. The primary goal of predictive adaptation is to produce rules (e.g., ***if*** *the temperature is below* 7 °C*,* ***then*** *the rowers are on sea anchor*) that can be used for *predictive* adaptation. Additionally, the ToTEM system is implemented on mobile smartphones, and we wanted to take into account how much of the system's processing resources (e.g., battery power) the data mining algorithm would need for predictive adaptation.

Given these considerations, a rules-based classifier, JRip [11], was selected. JRip uses relatively few system resources and produces a series of rules that can easily be converted into a series of *if...then* statements. For comparison, tree-based algorithms (J48 and Random Forest) and functional algorithms (Logistic Regression and Support Vector Machine (SVM)) were all applied to the data set to compare the results and quality of the rules generated by the candidate algorithms.

Once candidate data mining algorithms have been selected for predictive adaptation, a system change request is sent to the *Context Adaptation Controller* and the new classifier is implemented in the system.

3.2.3 Recognizing Sea Anchor Conditions

Reference Context Input provided to the *Context Monitor* contains information the system needs to recognize when the rowers are on sea anchor from historical context data. Ideally, this would be a sea anchor specific sensor. A sensor with such a direct relationship to a situation to be recognized is called a *predictor attribute*. As there is no predictor attribute for this situation, one needed to be identified as a starting point for context evolution from the historical data.

An analysis of when the rowers were on sea anchor revealed that the speed of the boat naturally converged to zero, and so we assume that there is a correlation. Therefore, the most suitable predictor attribute for when the rowers were on sea anchor was Speed Over Ground (SOG).

There was some uncertainty whether this correlation would be enough to recognize when the rowers were on sea anchor; there may be conditions when the rowers were stopped that the ToTEM system alerts needed to be enabled (i.e., the boat is stopped, but the rowers are not on sea anchor). An example of a situation like this is scraping mussels from the hull of the 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, SOG was not adequate enough on its own to predict when the rowers were on sea anchor; context evolution was required to refine the predictive adaptation rules using SOG as a predictor attribute. That is, the system needs to learn to recognize exactly when the rowers are on sea anchor (thus, turn off system alerts) and not just stopped for other reasons (may still need system alerts enabled).

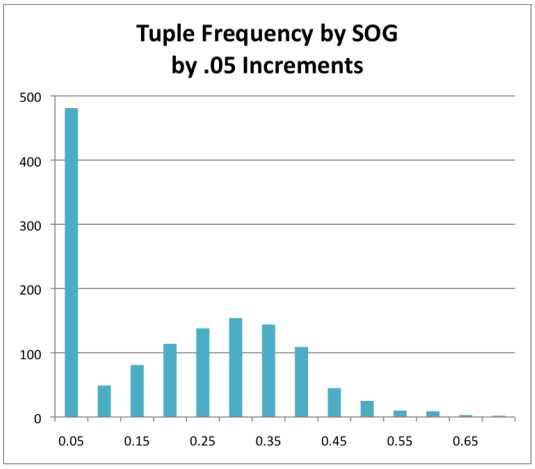

Figure 3: Tuple frequency for Speed Over Ground for the entire trip (21 days, days removed that didn't fall within 91 +/-4 readings per day).

The next step after identifying SOG as a predictor attribute was to determine the appropriate SOG sensor data ranges for determining when the rowers were stopped. The frequency distribution of the measurements for SOG can be seen in Figure 3. SOG provides clear separation between 0.01 and 0.03 of when the rowers are stopped and when they are actively rowing. Several cycles of refinement were used to narrow down the classifier threshold from 0.05 down to 0.01. The classification results with threshold 0.01 produced the rules with the highest accuracy rate across all the data mining algorithms.

Following the identification of the threshold, we could determine with a very high accuracy as can be seen in Figure 4 when exactly they stopped.

3.2.4 Deriving Predictive Adaptation Rules

An example of the predictive adaptation rules produced by the JRip algorithm on the data from all 14 sensors with outlying days removed (1378 tuples) are shown below:

- (Latitude <= 0.412296) and (Latitude >= 0.405714) and (Longitude >= 0.405633)
- (InBed >= 1) and (Effectiveness <= 0.6411) and (COG <= 0.247911)
- (Latitude >= 0.999854)
- (Longitude >= 0.934572) and (Longitude <= 0.934607)
- (Day <= 0.210526) and (COG <= 0.682451) and (Temperature <= 0.27027) and (Altitude <= 0.072993)
- (Latitude >= 0.711287) and (Latitude <= 0.711383)
- (Latitude >= 0.789422) and (Latitude <= 0.789457)
- (Latitude >= 0.927052) and (Latitude <= 0.927068)

These predictive rules can be interpreted as *if...then* statements to identify the *on sea anchor* context from sensor data. They are compared against incoming sensor data starting from the first rule and working sequentially through to the bottom in order to find a match in conditions. These rules are then used by the *Context Manager* to identify when the rowers are on sea anchor and trigger *preventive* adaptation. These are the times where the current values for the context-attributes meet one of the provided rules.

Identifying Important and Dropping Unimportant Context-Attributes
Those algorithms that produced good results were systematically applied to various combinations of context-attributes for comparison. Relevant context-attributes were identified and irrelevant ones were dropped from further predictive rules refinement iterations. In this way, the identification of context-attributes that contributed most significantly to producing the most accurate rules by order of influence was enabled and recorded in the *Knowledge Base*. This data is used for predictive adaptation cases where important context-attributes relevant to a particular requirement are no longer available to the *Context Manager* (e.g., the sensors to capture the context-attributes are broken).

In our example it was determined from the analysis that at least one attribute (CEP) was never covered by the rules produced by the algorithms and was therefore removed from rule refinement iterations and dropped from the list of relevant sensors. It did not occur in any rule sets produced by JRip. For the three attribute combinations shown in Figure 4, it only appeared in the rules that included outlying days. An explanation of Figure 4 is described in Section 3.2.5

3.2.5 Algorithm Accuracy for Predictive Adaptation

Performance results for the different classification algorithms applied to the historical context data from the Vancouver Island row are listed in Figures 4 and Figure 5, shown at the end of this paper. Best algorithm performers for percentage of tuples in the historical data set that were correctly identified as being on sea anchor, and percentage of those tuples incorrectly identified as being on sea anchor (false positives) shown in Figure 4 are bolded under

All Attributes (14)				
Outlier Days?	***Algorithm***	***%Correct***	***%FP***	***Size***
Yes	JRip	**93.9**	**5.6**	12 rules
	J48	93.8	6.2	44 leaves/87 s.o.t.
No	JRip	94.2	5.7	9 rules
	J48	94.5	**5**	36 leaves/71 s.o.t.
	Random Forest	**95.6**	5.3	10 trees, 4 r.feat's
Mean & S.D.		94.4 +/-.7	5.6 +/- .6	
All Attributes Except Latitude and Longitude (12)				
Outlier Days?	***Algorithm***	***%Correct***	***%FP***	***Size***
Yes	JRip	90.5	7.6	12 rules
	J48	**91.6**	**7.4**	61 leaves/121 s.o.t
No	JRip	91.9	7	12 rules
	J48	92.3	6.2	33 leaves/65 s.o.t.
	Random Forest	**93.5**	**5.3**	10 trees, 4 r.feat's
Mean & S.D.		92 +/- 1.1	6.7 +/-.9	
All Attributes Except Latitude, Longitude, and Days (11)				
Outlier Days?	***Algorithm***	***%Correct***	***%FP***	***Size***
Yes	JRip	84.1	**9.2**	17 rules
	J48	**86.1**	9.3	82 leaves/163 s.o.t.
No	JRip	87.0	9.5	18 rules
	J48	87.5	8.8	83 leaves/ 165 s.o.t
	Random Forest	**88.1**	**7**	10 trees, 4 r.feat's
Mean & S.D.		86.6+/- 1.6	8.76 +/-1.0	
All Attributes Except Latitude, Longitude, Days, COG, CEP (9)				
Outlier Days?	***Algorithm***	***%Correct***	***%FP***	***Size***
Yes	JRip	83.5	9.9	25 rules
	J48	**87.5**	**8.9**	91 leaves/181 s.o.t.
No	JRip	86.3	9.3	16 rules
	J48	**88.6**	**8.1**	68 leaves/135 s.o.t.
	Random Forest	88.1	8.2	10 trees, 4 r.feat's
Mean & S.D.		86.8 +/- 2.0	8.9 +/-.8	

Figure 4: Performance of algorithms

each attribute grouping for both sets of data. Mean and standard deviation of all algorithms applied to a given set of sensors over both sets of data for both percentage correctly classified and percentage of false positives are shown at the bottom of each attribute set.

Moreover, we compared the entire non-error-prone data set 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). We used a standard 10-fold cross validation technique [13] in which the dataset on context information collected during the Vancouver Island row was divided into 10 equal sets. Nine sets were then used to train the algorithm and which was then used to classify the results of the tenth set. The training was repeated ten times using each possible combination of the ten sets. Standard metrics are then computed and average accuracy of the algorithm derived.

All algorithms were processed using 10-fold cross-validation from the same data set to ensure accurate results.

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. However, for comparison, when applied to the data set that didn't include outlier days, 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, Days, COG, or CEP.

As can be seen from the data, 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.

4 Discussion

We have shown how important context-attributes and their conditions – in our case, specific value ranges in environmental and biometric data – can be identified through data mining on historical context data from sensors to trigger predictive systems adaptation. We have also shown how context evolution can be implemented through data mining when there is a lack of directly correlated context attributes for specific requirements (i.e., no 'on sea anchor' sensor for ToTEM).

Predictive Context Adaptation

We showed in our evaluation that the rules produced by the JRip algorithm on 1845 tuples of historical data could predict when the rowers were stopped in 93.9 % of the cases. Additionally, we were able to identify which

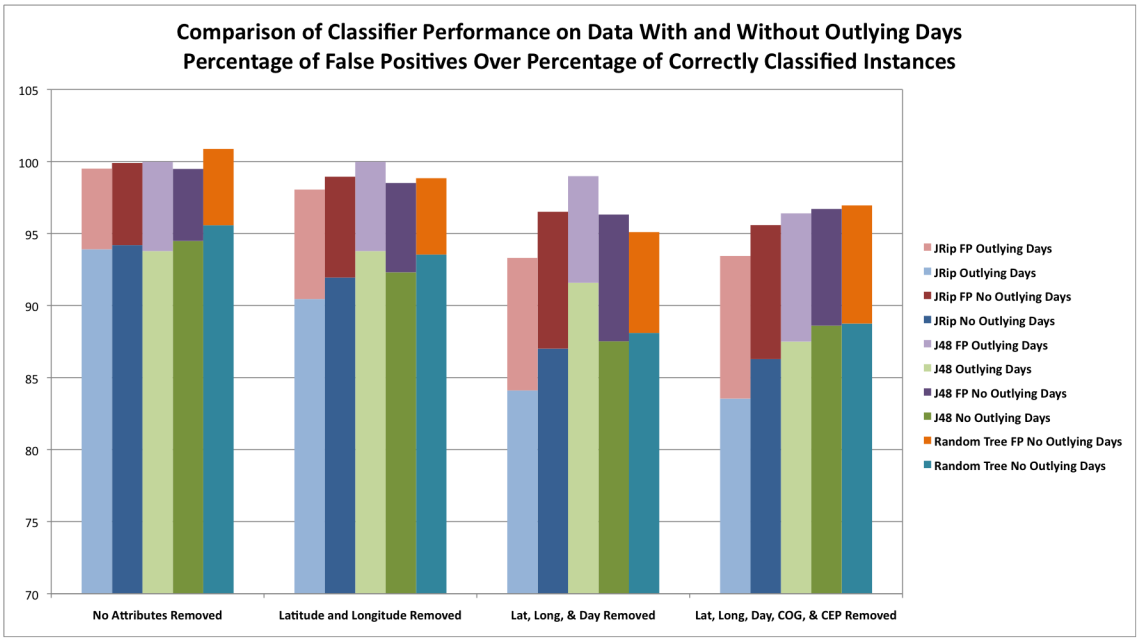

Figure 5: Comparison of algorithms with and without outlier days on different contextual attribute sets.

context-attributes were more important than others within these context conditions. Ranking context attributes in this way was found to be useful for context evolution (e.g., in cases where an important context attribute sensor is no longer available to the system for monitoring).

Data and Adaptive System used for Evaluation

The data used in this study was collected during a rowing test run around Vancouver Island. The purpose of this test voyage was to ensure that all sensors and equipment were functioning properly. This preparatory row was more casual than a true row in both the quality of data collection and in rowing consistency. There were multiple trips ashore, and sensors were tested. This resulted in noisy data that needed to be cleaned from the data set (preprocessed) before it could be used for data mining. Despite periods of inconsistent data collection, and the data set being reduced from 1862 to 1592 rows of data (to 85.5% after initial data preprocessing) and then to 1378 rows of data (to 74% after removing outlier days), we were able to derive multiple rules for monitoring these conditions for predictive systems adaptation. Our results consisted of more than 90% correctly classified instances with some of the algorithms when applied to the generated rules to the Vancouver Island data set. We expect the data quality to be even better on true rows (i.e., not test or preparatory rows) using the same equipment.

ToTEM is implemented to operate in a very specific *contextual environment*. In fact, we were able to base our system in a setting that was almost experimental as the rowers operated in very isolated, physically extreme conditions with a very small number of outside influences. The ranges of each environmental variable are relatively broad and easy to discern. While this made it relatively easy for us to identify and eliminate noisy data, more complicated contexts or more subtle conditions may give less accurate or less clear results. Nevertheless, as we used more than 10 context-attributes from a starting data set of 1862 rows of data, and had a classification rate of more than 90% of correct classified instances, we are confident that by using even larger data sets, the results should be still in a range that is useful for the system, if not even better than the results we achieved.

System Adaptivity in Unobservable Environments

The results given in this paper are based on an adaptive mobile system. Though the operational setting and direct users of our system are relatively unique (i.e. four elite athletes on an open-ocean rowing voyage), the overall challenge of developing a system for an unobservable operational context is not. Thus, our results are potentially applicable to other mobile or ubiquitous systems.

During the design of ToTEM, we could not observe the operational environment of the system. We were only able to use passively collected sensor data to understand the system's operational context. The analysis of unobservable operational environments for context becomes increasingly significant as mobile and cloud system developers create products that are used by extremely broad audiences in unexpected settings. Developers in these cases cannot anticipate the full scope of settings and users, and thus, cannot possibly anticipate all the context differentiations involved in fulfilling user requirements. Our approach can be helpful to developers designing and implementing systems for such unobservable environments to support adaptive systems in changing, unknown conditions.

5 Future Work

In the future, we plan to apply our approach to a new data set from OAR Northwest's next rowing trip (i.e., Dakar to Miami). Again, this analysis will be based on contextual data gathered from environmental and biometric sensors. Moreover, as a next step we plan to investigate requirements evolution and the identification of new requirements at runtime and mapping them to context attributes so that we can expand the number of requirements and further evaluate our approach.

Villegas et al. proposed three kinds of system adaptations: *predictive, preventive* and *corrective* [12]. While we applied data mining to *predictive* adaptations in this paper, we plan to further investigate the integration of data mining into adaptive systems for *preventive* and *corrective* adaptation. In *preventive* adaptations, the system detects *upcoming* events and starts adapting to them ahead of time. *Corrective* adaptations occur when the system detects that user requirements are no longer being met.

6 Related Work

Cayci et al. propose analyzing the execution behaviour of data mining algorithms and extracting a behaviour model for adapting data mining algorithms for different contexts [1]. They propose adapting the algorithms based on historical data about executions as data mining requires significant computing resources. Krishnaswamy et al. provide state-of-the-art mobile stream mining in which they present an overview of possible adaptation strategies, so that crashes on a phone can be avoided [8]. This could be useful for our approach. Nevertheless, they do not use data mining for context evolution (e.g., do not identify context-attributes to be considered for adaptation of a particular requirement). Haghighi presents a general approach for data stream mining based on context [6]. They aim at dynamically and autonomously adjusting data stream mining parameters according to the current internal context of the device (e.g., memory availability, battery charge or CPU utilization).

The presented work on data stream mining could be valuable for choosing the optimal algorithm for our approach. As the smart phone that is used at a rowing trip can be expected to have limited access to power and processing capabilities, it is necessary that the algorithm takes only a minimal amount of power.

Acknowledgements

We would like to give special thanks to the OAR Northwest members Adam Kreek, Jordan Hanssen, Markus Pukonen, Patrick Fleming, Greg Spooner, and Richard Tarbill for providing the data set from their rows and supporting us with their insights during the evaluation of the presented results. Without them, our evaluation would not have been possible. We would also like to particularly thank Jason Cummer.

The ToTEM project would not have been possible without his implementation of the longitudinal time adaptation functionality. We would like to thank Eric Knauss, Eirini Kalliamvakou, and Russ Rook for discussions that led to this paper.

About the Author

Angela Rook is a Master Student and Research Assistant in the Department of Computer Science, University of Victoria, Canada. Her research interests focus on feedback analysis from social media for requirements engineering and the use of data mining for the identification of credible and relevant stakeholders. She received a BA in Art History and a minor in Computer Science from the University of Victoria.

Alessia Knauss is a PhD candidate in the Department of Computer Science, University of Victoria, Canada. Her dissertation focuses on requirements elicitation for adaptive systems. The consideration of context is a major part of it. She received a Diploma Degree in Mathematics and Computer Science from Leibniz Universität Hannover, Germany.

Daniela Damian is an Associate Professor in the Department of Computer Science at the University of Victoria and the leader of the Software Engineering Global interAction Laboratory (SEGAL). Her research interests include collaborative software engineering with a special interest in global software development. She received his PhD in Computer Science from the University of Calgary, Alberta, Canada in 2001.

Hausi A. Müller is a Professor, Department of Computer Science and Associate Dean of Research, Faculty of Engineering at University of Victoria, Canada. Dr. Müllers research interests include software engineering, self-adaptive and self-managing systems, context-aware systems, and service-oriented systems. He received a Diploma Degree in Electrical Engineering in 1979 from the Swiss Federal Institute of Technology (ETH), Zürich, Switzerland and MSc and PhD degrees in Computer Science in 1984 and 1986 from Rice University in Houston, Texas, USA.

Alex Thomo is an Associate Professor and graduate advisor in the Department of Computer Science, University of Victoria, Canada. His research interests include theoretical and practical aspects of semi-structured and graph databases, social and biological networks, recommender systems, automata-based techniques, and index structures for textual data. Dr. Thomo received his PhD in Computer Science from Concordia University, Montreal, Canada in 2003.

References

[1] A. Caycia, E. Menasalvasb, Y. Saygina, and S. Eibeb. Self-Configuring Data Mining for Ubiquitous Computing. *Information Sciences*, 2013.

[2] B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. Marzo Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle. Software engineering for self-adaptive systems. chapter Software Engineering for Self-Adaptive Systems: A Research Roadmap, pages 1–26. Springer-Verlag, Berlin, Heidelberg, 2009.

[3] IBM Corporation. An Architectural Blueprint for Autonomic Computing. June 2006.

[4] R. C. Dorf and R. H. Bishop. *Modern Control Systems, 12th Ed.* Prentice Hall Publishing, 2011.

[5] S. Ebrahimi, N. M. Villegas, H. A. Müller, and A. Thomo. SmarterDeals: A Context-aware Deal Recommendation System based on the SmarterContext Engine. In *Proc. of the 2012 Conference*

of the Center for Advanced Studies on Collaborative Research (CASCON), pages 116–130, Riverton, NJ, USA, 2012. IBM Corp.

[6] P. D. Haghighi, A. Zaslavsky, S. Krishnaswamy, M. M. Gaber, , and S. Loke. Context-Aware Adaptive Data Stream Mining. *Intelligent Data Analysis*, 13(3):423–434, August 2009.

[7] K. E. Klein and H. M. Wegmann. Significance of Circadian Rhythms in Aerospace Operations. *IAGARDograph No.247*.

[8] S. Krishnaswamy, J. Gama, and M. M. Gaber. Mobile Data Stream Mining: From Algorithms to Applications. In *Proc. of the 13th Int. Conference on Mobile Data Management (MDM)*, pages 360–363, Washington, DC, USA, 2012. IEEE Computer Society.

[9] N. S. Nise. *Control Systems Engineering*. John Wiley and Sons Publishing, 2011.

[10] G. Tamura, N. M. Villegas, H. A. Müller, L. Duchien, and L. Seinturier. Improving Context-Awareness in Self-Adaptation using the DYNAMICO Reference Model. In *Proc. of the 8th Int. Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 153–162, Piscataway, NJ, USA, 2013. IEEE Press.

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

[12] N. M. Villegas, G. Tamura, H. A. Müller, L. Duchien, and R. Casallas. DYNAMICO: A Reference Model for Governing Control Objectives and Context Relevance in Self-Adaptive Software Systems. In *Software Engineering for Self-Adaptive Systems*, pages 265–293, 2010.

[13] 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.