The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Learning a Probabilistic Self-awareness Model for Robotic Systems

Raphael Golombek, Sebastian Wrede, Marc Hanheide, and Martin Heckmann

Abstract— **In order to address the problem of failure detection in the robotics domain, we present in this contribution a so-called *self-awareness model*, based on the system's internal data exchange and the inherent dynamics of inter-component communication. The model is strongly data driven and provides an anomaly detector for robotics systems both applicable in-situ at runtime as well as a-posteriori in post-mortem analysis. Current architectures or methods for failure detection in autonomous robots are either implementations of watch dog concepts or are based on excessive amounts of domain-specific error detection code. The approach presented in this contribution provides an avenue for the detection of more subtle anomalies originating from external sources such as the environment itself or system failures such as resource starvation. Additionally, developers are alleviated from explicitly modeling and foreseeing every exceptional situation, instead training the presented probabilistic model with the known normal modes within the specification of the robot system. As we developed and evaluated the self-awareness model on a mobile robot platform featuring an event-driven software architecture, the presented method can easily be applied in other current robotics software architectures.**

I. INTRODUCTION

Nowadays, autonomous robots are capable of performing an increasing variety of socially relevant tasks in merely unconstrained scenarios and in close interaction with humans [10], [19], [8]. This trend gets exemplified in the advancement of collaborative robot challenges such as the RoboCup@Home [15] initiative for service robots. Consequently, the inherent complexity of the tackled problems and the level of autonomy as well as the extended functionality of the underlying hardware and software architectures increases the overall complexity of robotic systems dramatically. Ultimately, robotic software developers are overburden by conceiving all possible interactions between the features within a robot system, in particular if it is deployed in a dynamic environment. Hence, it is often impossible to guarantee that robots operate in any circumstance within its specifications. However, as this is a critical precondition for robotic systems in close human-robot interaction, it remains an important challenge for robotic research to guarantee safe robot behavior at any time of operation.

Along these lines, a novel method for fault diagnosis in robotic systems is presented. It is based on a probabilistic model learned in a supervised fashion, providing an avenue to increase the safety and robustness of robot operation. The presented approach alleviates developers from the need to explicitly specify every exceptional or normal situation at design time of a to-be-controlled system, as usually done in model-based fault diagnosis [12].

Our approach is conceptually embedded in the framework of *Autonomic Computing* [14] (AC) that we want to employ for autonomous robots. AC describes the vision of software architectures embedding the necessary computational intelligence for anomaly detection and acting upon anomalies in the infrastructure itself. Here, the idea is to mimic the human nervous system thereby equipping the robot with abilities to self-heal, self-optimize, self-protect and self-configure. An essential precondition to realize these capabilities is the ability of the system to be aware of its own systemic condition i.e., the system has to be self-aware, which can also be viewed as kind of internal alarm mechanism as described by Sloman in the CoggAff architectural schema [20]. Hence, the self-awareness model provides the basis for closed-loop control schemes acting on autonomic (robotic) elements [14], which provide interfaces that allow the modification of system parameters at runtime to ultimately achieve the desired self-* attributes in autonomous robots.

Our contribution is organized as follows: Section II reviews related work on models and algorithms suitable for fault diagnosis. Section III describes the so-called *self-awareness model*, monitoring the system's internal data exchange and the inherent dynamics of inter-component communication. Section IV explains the integration of this model on our mobile robot *BIRON* [10]. This example illustrates the use of the model in a mobile robotic systems and outlines the required architectural support for its application. We argue that recent event-based and message-passing architectures as commonly found in many robotic systems [19], [8] are particularly well-suited to directly employ our approach. Following up, Section V describes experiments we conducted on BIRON while the robot performs tasks defined in the RoboCup@Home competition, see Figure 1, and discusses the performance of the self-awareness model. The evaluation is carried out based on several situations with decreasing discriminatory power outside of the robots functional specification. Finally, Section VI summarizes our contribution and provides an outlook on further challenges.

II. FAULT DIAGNOSIS AND ANOMALY DETECTION FOR ROBOTIC SYSTEMS

Current solutions to cope with errors arising in the context of robotic systems are mainly based on intensive specification [1], testing and exhaustive coding of recovery routines for known failures. However, this is a suboptimal solution

R. Golombek and S. Wrede are with the Research Institute for Cognition and Robotics, Bielefeld University, P.O. Box 100131, Bielefeld, Germany `[rgolombe,swrede]@cor-lab.uni-bielefeld.de`

M. Hanheide is with the School of Computer Science, University of Birmingham, UK `m.hanheide@cs.bham.ac.uk`

Martin Heckmann is with the Honda Research Institute Europe, Offenbach, Germany `Martin.Heckmann@Honda-RI.de`

for at least two reasons. First, to cover as many failures as possible requires expert knowledge concerning all of the individual system components as well as their interplay and non-functional characteristics. This is hard or even impossible if software re-use is envisioned. Secondly, it is not possible to foresee every exceptional situation which might lead to critical behavior, especially if non-expert users are interacting with these systems. Further approaches cope with errors by incorporating redundancy in the system thereby reducing the probability of a total fallout [5]. Such methods may be seen as supplementary solutions as they focus on the effect rather than the cause of the problem.

In this paper we propose a more flexible method to detect errors in robotic systems. Our approach is based on inter-component communication of the system and therefore can be learned from data. As model basis we choose the concept of anomaly detection. which refers to the problem of finding patterns in data that do not conform to expected behavior. Exploiting anomaly detection for the self-awareness model is reasonable as it does not demand for the specification of all possible faults which may occur while the robot is in field.

A plenty amount of work exist concerning anomaly detection. Existing approaches can be broadly subdivided into two groups, deterministic and stochastic approaches. Deterministic approaches like classification based approaches [21] assume that it is possible to cluster the feature space in order to separate normal classes from anomalous ones. Here, a pre-computed model is used which makes the testing phase very fast. To each test instance a definite label (class) is assigned which is disadvantageous when information about the degree of class-membership of an instance is desired rather than a concrete label. There are also Nearest Neighbor based approaches [4] which assume that normal data instances occur in dense neighborhoods, while anomalies occur far from their closest neighbors. This technique is purely data driven and unsupervised. The drawback is that the performance greatly relies on a distance measure.

Statistical techniques [9], [18], [3], [26] assume that normal data instances occur in high probability regions, while anomalies occur in low probability regions. Here, the idea is to fit a model to the given data and then apply a statistical inference test if an instance belongs to this model or not. Besides the inferred decision a statistical model provides a scoring of anomaly by its nature which can be interpreted as the confidence of the decision.

The aforementioned related work do not completely fit the needs and properties of data gathered from a complex robotic system. Such data results from intensive component interaction and communication. Therefore it is reasonable to treat is as a sequence and assume that information about the system state is also encoded in the order of occurrence of the data elements as well as in the time intervals between them. Additionally, in the context of robotic systems uncertainty occurs due to noisy sensor inputs, variable computation time of the components and the unforeseeable events in the environment which is reflected in the data, too. Regarding this, applying classification based approaches will be a challenging task as they operate only on sets of data thereby completely omitting temporal properties and the order of the elements in the set. Deterministic approaches in general fail to cope with uncertainty which makes them an inappropriate choice, too. Using Markov or HMM based approaches [9], [3], [26] seems to be a promising solution. These techniques are able to capture the frequencies and the temporal order in a sequence. Furthermore, there exist an extension [18] which allows to explicitly model the duration between consecutive sequence elements. However, as these approaches are based on the Markov assumption, they do not cover situations where temporal correlation exists between elements of the sequence which do not occur in consecutive order. In case of a communication pattern produced by a set of strongly interacting components the Markov assumption does not necessarily hold which makes the application of such techniques also challenging.

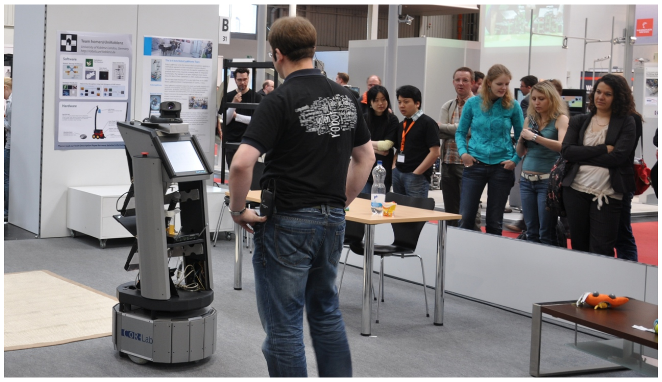
Fig. 1. BIRON performing at RoboCup@Home German Open 2009.

III. THE SELF-AWARENESS MODEL

In this paper we propose a different approach to build the self-awareness model. In contrast to the aforementioned probabilistic techniques, our approach is able to model durations between each combination of two elements of a sequence. By this means the model is able to capture more temporal correlations as it is not restricted to consecutive occurrences. Furthermore, our approach implicitly models the frequency of occurrence and the temporal order of the sequence elements.

As the basis for our anomaly detection we hypothesize the following: *The Components of a system communicate with each other to fulfill a given overall task. They produce and consume data which leads to complex communication patterns. If the communication is disturbed due to a component failure (i.e., missing data, delayed data) it is highly probable that the task won't be executed correctly. At the same time it is probable that the generated communication pattern will change, too. To preserve information encoded in temporal pattern the communication data is interpreted as a temporal ordered sequence.*

In our approach we exploit this hypothesis as follows:

A. Encoding Data

To build a model for the complete system we have to define a uniform representation of the recorded system data. This is necessary to process output from different components homogeneously in one model. Therefore, we define a transformation function f which operates on single data entity of the communication sequence as:

$$f(d_j) : \mathcal{D}_j \mapsto \mathcal{E} := e \tag{1}$$

where d_j denotes an element of the sequence with the specific domain space $\mathcal{D}_j$ and $\mathcal{E}$ denotes the common domain space. We call $\mathcal{E}$ the event space and the representation e of the data entity d_j in $\mathcal{E}$ an event.

B. Building The Self-awareness Model

The self-awareness model is build upon a training event sequence E_{train}. Let $U_E = \text{Set}(E_{train})$ be the unique set of all events in E_{train}. For each tuple $(e_i, e_j) \in U_E \times U_E$ a distribution $P_{(i,j)} := P(t|e_i, e_j)$ is derived from E_{train}. $P_{(i,j)}$ describes the probability that the event e_i occurs at time stamp t_i and that the event e_j follows this event with a delay of t, respectively occurring at time stamp $t_i = t_j + t$. Additionally, we constrain e_i to be the last seen occurrence of this type of event as we want to model temporal correlations only between the current event and the last occurrence of a given event.

We approximate P with a histogram based probability distribution. This discrete technique makes no assumption regarding the shape of the distribution over the durations between the events and features fast computation. The only parameter which has to be defined is the bin size s_b of the underlying histogram. s_b defines the resolution of temporal correlation modeled with the distribution $P_{i,j}$. The algorithm in listing 1 describes how to build the distribution for two events e_i, e_j.

Algorithm 1 calculateDistribution(e_i, e_j, s_b, E_{train})

```
$P_{i,j} \leftarrow 0$
$t_i \leftarrow 0$
for $e = e_1, \dots, e_n$ do
    if $e == e_j$ then
        $b = \lfloor \frac{t(e) - t_i}{s_b} \rfloor$
        $P_{i,j}(b) = P_{i,j}(b) + 1$
    else
        if $e == e_i$ then
            $t_i = t(e)$
return normalize($P_{i,j}$)
```

The algorithm traverses the whole sequence E_{train}. It permanently tracks the time stamp t_i of the last occurrence of the event e_i. Each time it discovers the event e_j in the sequence it calculates the timespan between the point in time t_i at which e_i has been seen last and the time stamp $t(e := e_j)$ of the just discovered event e_j and updates $P_{i,j}$. After the whole sequence has been traversed, $P_{i,j}$ is normalized to fulfill the requirements of a probability distribution.

Based on $P_{i,j}$, the self-awareness model is defined as $M = \{P_{i,j} | (e_i, e_j) \in U_E \times U_E\}$ i.e., the set of probabilities for all tuples in $U_E \times U_E$.

C. Evaluating The System State

The evaluation of the current system state corresponds to calculating a fitness value for a sequence of events E. We call this fitness value the *score* of the sequence and denote it with $\hat{s}_E$. In order to assess $\hat{s}_E$ we first calculate a score s_j for each event e_j in E and then average over the number of events:

$$\hat{s}_E = \frac{1}{\#E} \sum_{e_j \in E} s_j \tag{2}$$

Calculating the score s_j for an event e_j which occurred at the time stamp t comprises querying M to get all probabilities $p_{i,j} = P_{i,j}(\Delta t_i)$ and fusing them to a single score s_j. This again requires that the timestamps of the last occurrences of each event e_i which appeared during training are tracked to calculate the corresponding time spans Δt_i. The score value s_j is defined as:

$$s_j = \sum_{e_i \in E} w_{i,j} \cdot p_{i,j}. \tag{3}$$

Where E denotes the set of events tracked over the time. The weight $w_{i,j}$ is defined as:

$$w_{i,j} = 1 - \frac{h_{i,j}}{H_j}. \tag{4}$$

$h_{i,j}$ denotes the entropy of the distribution $P_{i,j}$ and $H_j = \sum_{e_i}^{E} h_{i,j}$ is the sum of all entropy values of the distributions $P_{i,j}$ which have to be considered when calculating the score of the event e_j. The reason to weight each $P_{i,j}$ with its entropy value $h_{i,j}$ is that it provides valuable information about the correlation of the two corresponding events at low computational costs. A high entropy value indicates high uncertainty in the distribution and consequently low correlation.

D. Assessing The System State

To assess abnormality of the system it is necessary to map the continuous score value for a sequence E to a binary decision (i.e., normal/abnormal). This is done by a threshold function:

$$\text{abnormal}(E) = \begin{cases} \text{True} & : \hat{s}_E < s^* \\ \text{False} & : \text{else} \end{cases} \tag{5}$$

If $\hat{s}_E$ of E is high enough it is declared as normal. Otherwise abnormality is assumed. The threshold s^* is determined by calculating the *receiver operating characteristic curve* [22] on test-data and finding the optimal cut-off in the curve.

IV. MODEL APPLICATION IN SERVICE ROBOTICS

The self-awareness model introduced in the previous section was continuously integrated and evaluated on our mobile robot *BIRON*. The BIRON efforts are aiming at developing a robot companion exhibiting social interaction capabilities [23]. The robot is i.e. able to focus and recognize its current interaction partner from a number of persons, which allows the robot to provide personalized services and to interact in a more efficient and natural means while carrying out useful tasks. One of the scenarios we aim at is the so-called home-tour scenario which is driven by the vision of future household robots being introduced for the first time use after purchase.

While we claim that the self-awareness model described in this paper is generally usable across a wide range of similar robotic systems, minimal requirements for its application are (i) a modular architecture, (ii) the capability to monitor internal component communication and (iii) the existence of an encoding function f. In the remainder of this section, these requirements and their fulfillment on BIRON are explained to exemplify the use of the self-awareness model on a recent service robot.

A. FUNCTIONAL MODULARITY

A necessary pre-condition most current robotics systems fulfill is the call for a modular system architecture. Despite other disadvantages for system development, the lack of functional structure in a robotic system makes the application of the self-awareness model challenging as the structure information is implicitly exploited in the model. Here, expert knowledge may be needed to instrument monolithic building blocks with additional measuring points in order to compensate the missing inherent functional decomposition.

In contrast, BIRON features a modular architecture. The number and type of active system components in a certain context depends on the task the system has to accomplish. In this work we focus on a specific task called "Follow Me" where approximately 15 components reaching from motor control and SLAM to face detection as well as person tracking and path planning are active.

Typically, exchanged data in these systems spans over a wide range of abstraction levels. On the one hand, components exist which produce raw data output like the laser scanner component. On the other hand, components such as multi-modal person tracking [13] generate high-level hypothesis about the presence and state of person in a scene. Hence, while modularity is a necessary pre-requisite for the self-awareness model, it must be able to process and monitor data from different abstraction levels.

B. ARCHITECTURAL SUPPORT

The self-awareness model has to be trained from internal information which requires architectural support for monitoring internal data flow. A conceivable worst case scenario is again a monolithic system without architectural support for introspecting the data flow in the system. In this case the monitoring of the communication has to be realized on the programming language or on the operating system level.

In contrast to this, *Event-Driven Architectures* [16] usually allow components easy monitoring of inter-component communication by subscribing to all advertised event types. Received event notifications are in turn be used to train the self-awareness models for certain operational modes of the system.

Information exchange on BIRON utilizes so-called active memories [24], [11]. These virtual shared memories are used for the integration of the different layers in BIRON's system architecture [7]. This concept puts forward the ideas of event-driven integration on the basis of flexible notification and a heterogeneous document-oriented data model. On BIRON, the self-awareness model thus just subscribes to all events generated through active memory instances.

C. ENCODING FUNCTION

Having access to the exchanged data, the encoding function f maps a recorded data point d to an event domain space $\mathcal{E}$. This is beneficial to process data from different components with different domains in one model.

In context of the BIRON system the mapping function is defined as follows:

$$f(d) : D \rightarrow SOURCE \times SCOPE \times TYPE$$

That is, each data point d is mapped to an event e which represents the producing component of the data point, the scope where this data point can be perceived by other components and the type. Here, the type of an event i.e. takes one of the values *insert* (new data), *delete* (data becomes invalid) or *update* (data changes). This extensible set of values maps to common processing possibilities of information.

This definition of f is very general and it is possible to define more sophisticated mapping functions which might take further information into account (e.g., the content of a data point or further properties of the generating component or the communication channel). However, we chose this definition of f to minimize the required knowledge about the functioning of the components in the system.

Furthermore, this definition of f can be applied straightforward to other robotic systems developed with similar approaches such as DORA built using the CAS toolkit where all processes are coupled by means of working memories through which information is mediated and exchanged. With little adaptation of the encoding function, the approach can be transferred to other architectures at least sharing a data-driven or event-based robotics middleware concept, e.g. ROS [17], Yarp [6] or OROCOS [2].

V. EVALUATION

A. Scenario

The self-awareness model was evaluated on data recorded during the interaction between a human and our robotic platform called BIRON. As a realistic interaction scenario the robocup@home task called "Follow Me' has been chosen. In the context of this task the human partner uses the commands *follow me*, *stop*, *turn right* and *turn left* to advice the robot what to do. In return the robot is allowed to ask questions (e.g. do you want me to follow you?) if it is unsure which action to perform. This task requires 15 different components which exchange information at an overall frequency of 163Hz. Thereby, the range of frequencies spawns between 0.5Hz and 86Hz.

B. Induced Error Cases

To determine the performance of the algorithm we simulate the occurrence of different types of errors. The first two failures are crashes of components which we denote with CC_1 and CC_2. For CC_1 we trigger the crash of a component called player which publishes laser data into the system. This crash results in the immobility of the robot. The second crash affects the slam component which generates hypothesis about the robot's position in the environment. Without the information from the slam component, the robot cannot navigate anymore. With the help of these two cases we test if the approach is capable of detecting errors on different levels of data processing i.e., barely processed sensory data and hypothesis of the robot's position. As third error we trigger a resource starvation (RS) failure by inducing a high CPU-load via a dedicated *busy worker* component. Resource starvation errors are of interest as they occur fairly often when integrating independently developed components into a robotic system. Another interesting abnormality occurs in the context of distributed systems and results from asynchronous clocks of the underlying operating system. This error affects all pairs components which rely on synchronization based on the system clock if they run on different hardware. We denote this error the asynchronous communication error (AC).

C. Results

For each error case we record 100 seconds of internal communication data shortly before and after the occurrence of the error. Additionally, we record four data sequences of normal system behavior where normal behavior is defined by the system designer[1]. The number of events in the records varies between 9000 and 16000. Two records of normal behavior are used to train the model, one to generate the anomaly threshold s^* and one for testing purposes. For each record a sequence S of score values is calculated by first splitting the record into sequences E_i of 100 milliseconds length and then applying the formula 2 to each E_i. The threshold is calculated with the ROC curve method and results in $s^* = 4.5$. The record of normal behavior which we dedicate for testing is used to calculate the False Positive rate resulting in 6.51%.

Figure 2 shows a qualitative overview of the score sequences calculated for the error records. In each plot the threshold s^* is drawn for comparison. Each plot displays two seconds of the recorded data nearby the point in time when the particular error has been triggered. This position is marked with a vertical bar in each plot. Each plot shows a degradation of the score value after the failure has been triggered. For CC_1 and CC_2 the degradation happens nearly instantaneously. This can be ascribed to the sudden omission of the events related to the crashed component. The timestamps of occurrence of the omitted events won't be updated in the algorithm and the durations between the last

[1]For the case of normal behavior it cannot be guaranteed that the recorded data do not contain any inconsistencies. However, the probabilistic model used in this approach compensates small variations in the data.

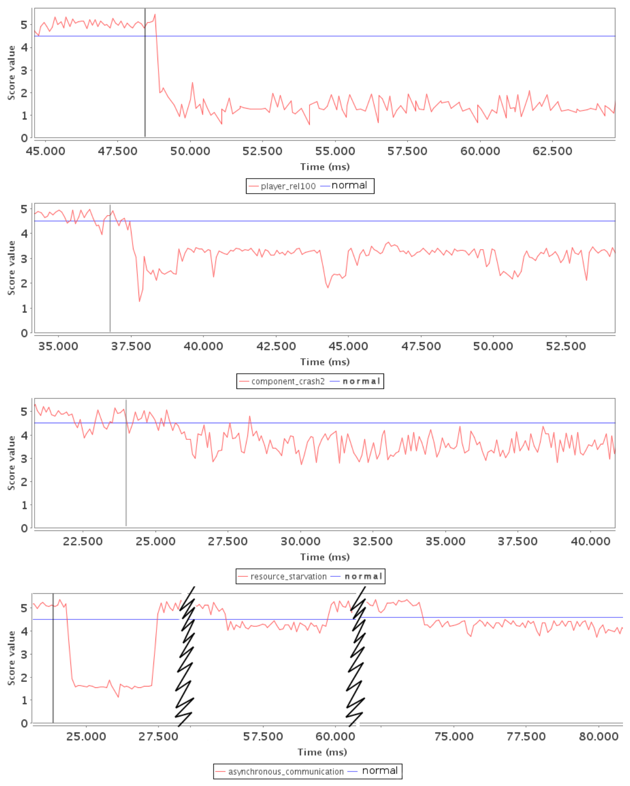

Fig. 2. Comparison of score values generated by the self-awareness model: The blue lines in each plot match the threshold $s^* = 4.5$. The red curves match diverse induced failures. Note note in the case of asynchronous communication the plot has been copied together to demonstrate the occasional occurrence of the error.

occurrence and the current timestamp increase. After some time the durations reach values which are not covered by the trained distributions and the score decreases. The algorithm is able to detect the occurrence of both errors. What is more important, in both cases the algorithm constantly reports the presence of the error without False Negatives. Regarding the resource starvation error RS the score curve shows a different behavior after inducing the failure. Instead to collapse the score degrades slowly over time. This results from slowly increasing intervals between event emissions due to continuously diminishing resources of the system. However, after successfully detecting the error the model constantly tracks it with a False Negative rate of 1.7%. The last case, the asynchronous communication error AC differs from the previous ones in terms of its occurrence. It does not manifest itself constantly over time after it has been induced in the system but rather appears sporadically. The reason for this behavior may be that the asynchronism of the two clocks becomes sensible to the self-awareness model only if components are effected that synchronize each other via the monitored communication channels. This happens in sporadic intervals which leads to discontinuous detection of this failure. An deeper analysis of the event stream reveals that during the detection of the error an expected event does not occur. Usually, the *person tracking and anchoring component* of the system updates its hypothesis about tracked persons by emitting a specific event. Such a hypothesis will be discarded if it is too old which is usually a normal behavior. In the case of the asynchronous communication error there exist time intervals where no single hypothesis is tracked which results in complete absence of the update event and a degradation of the score value. To cope with such errors the self-awareness model requires an additional reasoning mechanism which enables the linking of sporadic occurrences of the error to a coherent failure recognition.

In this section we evaluated errors with permanent impact on the performance of the system after occurrence. However, in real world scenarios the Quality of Service (QoS) delivered by the system may decrease only temporarily. We expect our approach to detect QoS-degradation given that the quality parameters are encoded in the attributes of the recorded events. An exemplary application which would directly benefit from detection of QoS-degradation is the approach for composition of robotic services for complex task solving based on the maximization of the QoS of a system [25].

VI. CONCLUSION AND OUTLOOK

In this paper we presented a novel method for fault detection in robotic systems constructed on the basis of discrete event-based data interchange. The introduced self-awareness model is strongly data driven and thus (i) can be trained from good examples of normal system behavior, (ii) is largely independent from specific scenarios and (iii) shows promising results even for transient malfunctions in system behavior.

The resulting self-awareness model provides a basis for a sophisticated autonomic computing architecture in the domain of robotic systems, enhancing safety and robustness of robot operation and ultimatively increasing the autonomoy of intelligent robot systems.

Future work will focus on a broader and more realistic daily life evaluation as well as an exhaustive complexity and performance analysis of the demonstrated approach. Over a long distance our goal will be closing the autonomic control loop, effectively allowing modification of relevant system properties upon detected anomalies, the integration of anomaly detection with behavioral control and the further exploration of fault diagnosis models for robotic applications.

VII. ACKNOWLEDGMENTS

The research project "An Autonomic Computing Approach for Systemic Self-Regulation" is supported by the Honda Research Institute Europe.

REFERENCES

[1] R. Bischoff and V. Graefe. Design principles for dependable robotic assistants. *International Journal of Humanoid Robotics*, 1:95–125, 2004.

[2] Herman Bruyninckx. Open robot control software, 2008. http://www.orocos.org.

[3] Marta Casar and Jos A. R. Fonollosa. Overcoming HMM time independence assumption using n-gram based modelling for continuous speech recognition. In *European Signal Processing Conference. EUSIPCO 2008*, pages 3–7. EURASIP, August 2008.

[4] Varun Chandola, Eric Eilertson, Levent Ertoz, Gyorgy Simon, and Vipin Kumar. *Data mining for cyber security*. Springer, 2006.

[5] Ada Diaconescu and John Murphy. A framework for using component redundancy for self-optimising and self-healing component based systems. In *WADS workshop, ICSE*, 2003.

[6] Paul Fitzpatrick, Giorgio Metta, and Lorenzo Natale. Towards long-lived robot genes. *Robotics and Autonomous Systems*, 56(1):29–45, 2008.

[7] Jannik Fritsch, Markus Kleinehagenbrock, Axel Haasch, Sebastian Wrede, and Gerhard Sagerer. A Flexible Infrastructure for the Development of a Robot Companion with Extensible HRI-Capabilities. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 3419–3425, Barcelona, Spain, April 2005.

[8] B. Graf, C. Parlitz, and M. H
"agele. Robotic home assistant care-o-bot® 3 product vision and innovation platform. *Proceedings of the 13th International Conference on Human-Computer Interaction. Part II: Novel Interaction Methods and Techniques*, page 320, 2009.

[9] Robert Gwadera, Mikhail J. Atallah, and Wojciech Szpankowski. Reliable detection of episodes in event sequences. In *Knowledge and Information Systems*, pages 67–74, 2004.

[10] A. Haasch, S. Hohenner, S. H
"uwel, M. Kleinehagenbrock, S. Lang, I. Toptsis, GA Fink, J. Fritsch, B. Wrede, and G. Sagerer. Biron–the bielefeld robot companion. In *Proc. Int. Workshop on Advances in Service Robotics*, 2004.

[11] Marc Hanheide, Sebastian Wrede, Christian Lang, and Gerhard Sagerer. Who am i talking with? a face memory for social robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, 2008. IEEE, IEEE.

[12] Rolf Isermann. Model-based fault detection and diagnosis - status and applications. In Alexander Nebylov, editor, *Automatic control in aerospace 2004 : proceedings from the 16th IFAC symposium*, Saint-Petersburg, Russia, 2005. Elsevier.

[13] Kai Jngling, M. Arens, Marc Hanheide, and Gerhard Sagerer. Fusion of perceptual processes for real-time object tracking. In *International Conference on Information Fusion*, 2008.

[14] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *IEEE Computer*, 36:41–50, 2003.

[15] RoboCup league's committees. Robocup@home rules & regulations. 2009.

[16] D.C. Luckham. *The power of events*. Addison-Wesley, 2002.

[17] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully B. Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *Proc. of the Int. Conf. on Robotics and Automation*, Open-Source Software workshop, 2009.

[18] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, 53(3):267–296, 1990.

[19] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. The intelligent asimo: System overview and integration intelligent robots and system. In *Proc. Of IEEE/RSJ International. Conference on Intelligent Robots and Systems*, 2002.

[20] A. Sloman. Varieties of affect and the cogaff architecture schema. In *Proceedings Symposium on Emotion, Cognition, and Affective Computing AISB*, 2001.

[21] Claudio De Stefano, Carlo Sansone, and Mario Vento. To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 30(1):84–94, 2000.

[22] A.R. van Erkel and P.M.T. Pattynama. Receiver operating characteristic (roc) analysis: basic principles and applications in radiology. *European Journal of radiology*, 27:88–94, 1009.

[23] Britta Wrede, Marcus Kleinehagenbrock, and Jannik Fritsch. Towards an integrated robotic system for interactive learning in a social context. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems - IROS 2006*, Bejing, 2007.

[24] Sebastian Wrede. *An Information-Driven Architecture for Cognitive Systems Research*. PhD thesis, Technical Faculty – Bielefeld University, 2009.

[25] A. Yachir, K. Tari, Y. Amirat, A. Chibani, and N. Badache. Qos based framework for ubiquitous robotic services composition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[26] N. Ye. A markov chain model of temporal behavior for anomaly detection. In *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, pages 171–174, 2000.