Skill-based Exception Handling and Error Recovery for Collaborative Industrial Robots

A. B. Beck, A. D. Schwartz, A. R. Fugl, M. Naumann, B. Kahl

***Abstract*— Moving robots from their carefully designed and encapsulated work cells into the open, less structured human workspace for collaboration with workers requires robust error detection and recovery strategies. Foreseeing all possible uncertainties and unexpected events and to program in recovery actions at setup time is unfeasible. Online learning of nominal execution behaviour and automatic detection of anomalies using an Extended Markov Model, combined with interactively trained Bayesian networks for mapping anomalies to error causes and recovery actions, enables automatic recovery from previously experienced errors. A three-layered user-friendly model of errors—causes—responses and a simple GUI allows non-expert user to define new recovery activities and error causes when not yet handled anomalies occur.**

I. Motivation

Today's robot systems for industrial applications rely on a structured environment to avoid errors. Parts, fixtures, tools and stations have defined positions and the workspace is encapsulated to avoid intruders that could possibly endanger this defined environment. Expected exceptions from the nominal case that were either foreseen during the planning of the robot system, or occurred during the setup phase of the system are coped with by integrating additional sensors, adapting tool-, fixture and part geometries and adding additional branches to the robot program to cope with these deviations. Furthermore, as many robotic systems are complicated, any exceptions and breakdowns occurring after system setup often require external technicians or engineers to diagnose and solve problems.

Such strictly controlled and carefully designed work cells are only economically feasible if the designed robot system will run unobstructed for a long time. Small and mid-sized enterprises (SMEs) are often characterized by a much more agile production style and consequently rely on human workspaces. Moving robots out of their strictly controlled and carefully designed spaces into human workspaces, which are by nature unstructured environments with a high degree of uncertainty, requires significantly enhanced robustness towards unforeseen events and geometric or other uncertainties. (The additional need for safety measures to protect the human co-worker from injuries is out of scope of this work, see e.g. [22], [12] and many others.) A SME suitable robot system therefore needs semi automatic exception handling and error recovery capabilities that allow non-expert users to manage exceptions (internally and externally triggered) occurring in daily operation. We propose a novel skill-based exception handling and error recovery approach that allows non-robot expert users to operate a robotic system embedded in a human-centric workspace. We briefly introduce our execution model, detail the Extended Markov Chain based Situation Awareness, which forms the base for Exception Handling, and the Error Recovery module employing a Bayesian network and beta-binomial inference algorithm. The prosed system has been implemented in a pick & place and in an assembly work cell, which are finally presented.

II. Related Work

Research in exception handling is related to the area of error or fault recovery [17]. Error recovery has been defined as "the process by which the system returns to a state where production can restart after an abnormal and disruptive condition has occurred" [23]. For a robot coworker to effectively handle an exception, whether through informing the human worker or resolving the problem by itself, the types of faults that typically occur in the manufacturing robotic assembly cases needs to be understood. Fault taxonomies have been presented in other related fields, including mobile robots [7], computing [3], autonomous robots in RoboCup [21], workflow systems [16], service-oriented architecture [6], and web service [8]. Reports show that many errors in manufacturing systems, including CNC machines, are hardware related and that approximately 60% of all stoppages are due to tool breakdown [23]. However, there has been a lack of study on the likelihood of common errors and exceptions occurring during assembly tasks involving collaborative robots. One of the reasons can be that robot coworkers have not yet proven to be robust enough for industry application to be studied and generalized based on real assembly cases [14].

III. Skill-based Execution Model

At the base of the system is a Skill Execution Engine, which allows a more goal-oriented task description than strict motion based programming or planning. Without going into details of the skill-model [1], we assume skills to be independent, sensor-based motion or handling primitives that adapt themselves to position uncertainties and other deviations from an ideal state using build-in sensing and monitoring as well as (limited) internal error recovery. Robot tasks are constructed by chaining skills and control flow instructions, forming a state machine [2] based on SCXML[1]. While skills detect deviations from their expected performance and report these, the skill executor by itself does not provide any error recovery functionality. Features

* The research leading to these results has been funded by the European Union's seventh framework program (FP7/2007-2013) under grant agreements #608604 (LIAA: Lean Intelligent Assembly Automation) and #287787 (SMErobotics: The European Robotics Initiative for Strengthening the Competitiveness of SMEs in Manufacturing by integrating aspects of cognitive systems).

A. B. Beck and A. R. Fugl are with the Danish Technology Institute. E-mail: anbb@dti.dk and arf@dti.dk

N. Naumann is with the Fraunhofer Institute for Production Systems and Automation. E-mail: Martin.Naumann@ipa.fraunhofer.de

B. Kahl is with the Gesellschaft für Produktionssysteme GmbH Stuttgart. E-mail: bjoern.kahl@gps-stuttgart.de

[1] Apache Commons SCXML executor, http://commons.apache.org/proper/commons-scxml/.

of the skill executor that allow the implementation of error recovery functionality at higher layers are:

- The skill executor knows and publishes the current state of the system at any time. This allows an error recovery module to relate errors on the one hand to specific skill models and on the other hand to specific application steps and therefore to draw conclusions like "this is an error that is very typical for a pick operation" or "this is an error that occurred already in the past at this specific execution step of the application".
- The skill executor has an interface for an error recovery module to stop and later continue the execution of the skill based application program thereby allowing worker interaction to recover from errors detected by an error recovery module.
- The skill formalism used by the skill executor is built on the concept of reusable hierarchical skills that are easy to enhance or adapt. It is therefore easily possible to include additional mechanisms into an existing skill model to cope with errors that could be detected by the system but just have not been considered yet.

The Situation Assessment (SA) constantly monitors the overall situation (robot task execution) using data published by the skills executor as well as by additional sensors dedicated for situation assessment. Deviations flagged by the SA are further examined by the Exception Handling (EH), which devises a possible cause and corrective measure, potentially involving user interaction. The whole system of skill executer, situation assessment and exception handling is collectively referred to as "Exception Handling Framework" or "EHF".

IV. Situation Assessment

The role of Situation Assessment (SA) is to learn and monitor the (correct) skill execution and detect non-nominal conditions. Deviations from the learned, nominal behaviour are interpreted as Anomalies, which are passed on to the Exception Handler (section V). Our implementation of SA is based on prior work by [4] and [5], where SA was applied to mobile robotics. We implemented and expanded SA to learn skill based execution in a collaborative robotic system. To learn how to perform a skill correctly, SA captures the essence of the skill by learning the timing and sequence of events that make up the skill. Our approach is to generate one parameterized model that includes parameters in the space and time domain. SA learns the sequence of events within a skill execution by learning a set of parameters with a temporal component, recording the transition from one instantiation of the parameters to the next:

$$p(X) = p(X_1, X_2, \dots, X_n) \qquad (1)$$

where X, a *Situation Model*, denotes a set of parameters (a state), n denotes a discrete step in time, and p, a *Situation*, denotes the complete distribution of all the states within a skill. Each state X_i of X is parameterized:

$$X = [d_1, d_2, \dots, d_m] \qquad (2)$$

where d_i are data components such as sensor values or robot's internal state values.

A. Situation Model

Situation Assessment uses a *Situation Model* as a template description to fuse together the different data points for learning a Situation. The components of the Situation Model (d_i in (2)) are real number data, which can come from any source and have any meaning. In our experience, combining space and time is critical to the success of learning a skill. For instance, learning a skill using a 6D F/T sensor, the Situation Model s could be defined as in (3).

$$s_a = [primitive, F_x, F_y, F_z, T_x, T_y, T_z] \qquad (3)$$

The component $primitive$ of s_a is a data point that uniquely identifies the current primitive being executed in the skill. In this case, the unique primitive ID provides the understanding of time while the understanding of space is provided by the F/T data. By using the primitive ID we can learn a skill time invariantly. This means that SA will only learn the sequence of the events and is invariant towards the duration of the execution of specific primitives. We have found this feature particularly useful when the duration of the primitives or skills is stochastic. Should it be necessary to catch anomalies in relation to when events occur (e.g. too early or late), the primitive ID in (3) can be substituted with a time data point. Throughout our research, we have successfully applied SA to monitoring digital inputs, such as the state of one or more grippers. Through the rest of the paper, we will use the following Situation Model for implementation and testing:

$$s_b = [primitive, gripper_{open}, gripper_{closed}] \qquad (4)$$

where $gripper_{open}$ and $gripper_{closed}$ are binary outputs of reed switches of the gripper: $gripper_{open}$ is $true$ when the gripper is fully open and $gripper_{closed}$ is $true$ when the gripper is fully closed. Our assumption is that the gripper is grasping an object when both readings are $false$, indicating the gripper is neither fully open nor closed.

B. Data Processing and Clustering

The Situation Model serves as a template describing which sensors SA should fuse together into one single state. In general, all data points in the Situation Model have to be real numbers. This allows the computation of one single metric for each X_i in (1). We have so far used the Jaccard similarity coefficient as a method for clustering similar states. Through experimentation, we have found the algorithm to be useful despite its simplicity.

C. Dynamic Learning in Situation Assessment

SA can autonomously learn a skill without the user having to manually specify the states of a skill. We have implemented a spatiotemporal model that allows for online dynamic learning of states over time. For this purpose, we are currently using the Extensible Markov Model (EMM) as it is useful for online learning of sequences of states [10]. An example of a dynamically learned model using the EMM algorithm can be seen in Figure 1. In this example, a robot is picking up a nut from a table and placing it on a pipe in a single nonrecurring operation (therefore an open-ended chain). The EMM is also useful in learning looped tasks.

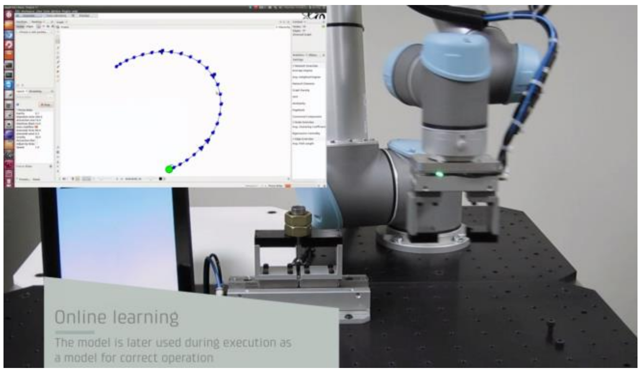

Online learning
The model is later used during execution as a model for correct operation

Figure 1. Example of learning a task. The upper left corner shows the temporal sequence of states the skill consists of. These states were learned online while the robot performed the task. A full video is available at https://www.youtube.com/watch?v=-CKbdQ3ocQo.

D. *Anomaly detection*

SA has two modes of operation: learning and detection during execution. In the learning mode, SA monitors the data points specified in the Situation Model and builds the Situation for the skill that is being learned. During execution of the same skill, SA loads the saved Situation and applies the same clustering process as during learning. However, should the clustering of the data result in a new state in (1), then SA will interpret that as an anomalous state has occurred and issue an Anomaly warning. Processing and handling the Anomaly is the task of the Exception Handler (EH) module.

V. EXCEPTION HANDLER

The task of EH is to receive an Anomaly from SA and provide a suggested solution that is most likely to solve the problem. For each robotic system, EH maintains a hierarchical four-layered Bayesian network with all exceptions and solutions relevant to that cell. The hierarchical structure allows EH to reason about the most suitable solution to a problem. EH provides the suggested solution to the user along with all other possible solutions. The user is free to select the suggested solution, any other solution or to create a new solution. The selection is stored in EH as a sample of user solution preference. Such samples are used in priming the network for inference with future anomalies. With the feedback of user samples, a closed preference-learning loop is formed to provide suggestions for solutions to future anomalies. In this section, we provide a detailed description of EH and begin with the role of the Exception Scenario ES in EH.

A. *Exception Scenario*

The Exception Scenario (ES) is designed as a four-layered model consisting of Anomaly, Error, Fault and Response, inspired by work in [18]. The hierarchy is a four-layered binary Bayesian network that facilitates inferring the most likely Response (solution) to an Anomaly (a deviation), an example is shown in Figure 2. At the lowest level of the network, Anomaly nodes model anomalies detected by SA. Each Anomaly node corresponds to a data component (d_i of (2)) in the Situation Model. Above Anomaly, the Error node models which kind of error the Anomaly is and if the Anomaly should even be considered an error. The Fault node models the root cause of the Anomaly and the Response node models the solution to the Fault. This model resembles the diagnosis model used by physicians when examining a patient: Based on symptoms (here: the detected error) an illness is inferred (here the fault) and a therapy decided (here the response). The intermediate step of a fault is necessary, since one and the same observed error (symptom) can have multiple causes. For example an unexpected gripper state can be due to a failed grasping operation, a missing object at the pickup position or a defective gripper itself.

B. *Bayesian Network*

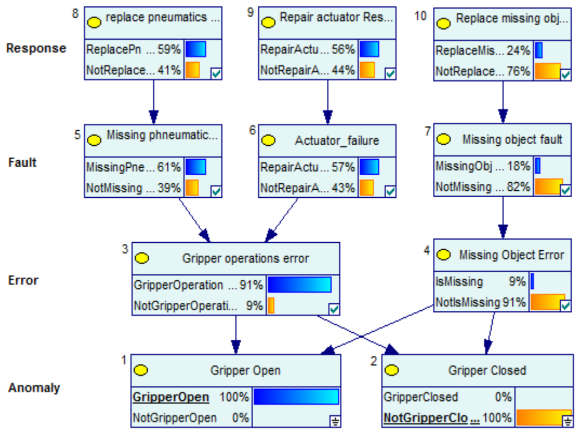


Figure. 2. Inference of cause and solution to *Gripper Open* anomaly. Both error nodes are dependent on both anomaly nodes, allowing the Bayesian network to further strengthen the belief about the cause of an anomaly. Simulated in GeNIe[1].

Figure 2 is an example of a Bayesian network with three Exception Scenarios for the two Anomaly nodes of the sensors $gripper_{open}$ and $gripper_{closed}$ in (4). Both Error nodes are dependent on both Anomaly nodes, allowing the Bayesian network to further strengthen the belief about the cause of an Anomaly (simulated in GeNIe[2]). The first two scenarios with nodes $s_1 = \{1,3,5,8\}$ and $s_2 = \{1,3,6,9\}$, offer two Responses to the Gripper Open Anomaly while the third scenario $s_3 = \{2,4,7,10\}$, offers a single Response to a Gripper Closed Anomaly. The numbers in curly braces indicate the node number in Figure 2. In this example we are modelling two faults {5,6} and Responses {8, 9} for the Gripper Open Anomaly. If a gripper is unexpectedly open (Gripper Open = true, Gripper Closed = false), we could interpret that as either a pneumatics failure (e.g. loss of air pressure) that can be solved by checking and replacing the air supply {5,8}, or an actuator failure (e.g. broken gripper) that can be solved by repairing the gripper {6,9}. In the reverse case of a closed gripper, we could interpret the failure as there was no object to grip and the solution is simply to replace the missing object. In Figure 2, the Faults {5,6} are modelled as belonging to the same Error, Gripper Operations Error {3}. This allows the network to learn user selections for a specific Fault, Response pair over other pairs belonging to the same Error node. The network is thereby able to encode knowledge specific to individual user environments.

[2] Figure 2 shows a screen capture of GeNIe, a Bayesian modelling environment developed by the Decision Systems Laboratory of the University of Pittsburgh. Available at http://genie.sis.pitt.edu

C. *Inference*

The process of inferring a Response to an Anomaly in the Bayesian network, is the inference process of the EH. This process is an implementation of Bayes' theorem:

$$posterior \propto prior \cdot likelihood \quad (5)$$

We have implemented Bayes' theorem in three steps:

- Calculate prior probabilities
- Introduce evidence to network
- Infer posterior probabilities

In the following, we describe each of these steps.

D. *Inference*

A prerequisite for performing inference is the calculation of prior probabilities. As described in section IV, a feature of the EHF is to learn the user-preferred solution of a given anomaly. When the user selects a specific Exception Scenario (i.e. an Error, a Cause and a Solution) to solve a problem, it is fed back to the database as a sample of the user selection, thus learning the preference of selecting this Exception Scenario for a specific Anomaly. The sample data is used to calculate the prior probability for each node of the network. We treat calculating the node's prior probability as an inference process that adds another layer of Bayesian inference as described in (5). We introduce the sample data from user selection of Exception Scenarios as the evidence to infer each node's posterior probability. Each node of the Bayesian network is a binary random variable modelling an event that either occurs or not. For instance, if the user selects the ES {1,3,5,8} in Fig. 2, then the user is confirming that the specific ES solved the problem (e.g. that a Gripper Open Anomaly did happen, it was caused by missing air pressure and the solution was to resupply the air). At the same time and equally important, the user is also confirming that alternative events {4,6} to ES {1,3,5,8} did not occur. Thus, with every selection of an ES, EH registers the confirmed nodes on all levels of the ES, as well as the rejected nodes. The process of selecting any node in the Bayesian network over time, can be viewed as a Bernoulli process following a binomial distribution as in (6).

$$X \sim \mathrm{Binom}(n, p) \quad (6)$$

where X is the number of times a specific node has been selected. n is the number of samples drawn in the sequence. If this process is sampled sufficiently, a distribution reflecting the user selection can be inferred from the sample set. However, in many cases it is not possible to provide a sample set of sufficient size and inference will be subject to uncertainty. To model this uncertainty, we model the user selection for each node as a hyper-parameter p, thereby modelling the user selection as a random variable itself and creating a hierarchical Bayesian model for calculating the prior probability [11]. This approach uses the samples of user selection as a likelihood function providing evidence to the inference process. Given the binomial likelihood, we have chosen the Beta distribution as the prior distribution (7).

$$(p \mid \alpha, \beta) = Be(\alpha, \beta) \qquad (7)$$

In (7), the user selection is modelled as the hyperparameter p, drawing samples from the Beta distribution. α and β is respectively the number of samples confirming and rejecting the selection of the node. The Beta distribution is a conjugate distribution to the Binomial distribution, thereby offering analytical tractability of the Bayesian inference process. The conjugate property ensures that when updating the prior Beta distribution (7) with new evidence following the Binomial distribution, the resulting posterior distribution is also a Beta distribution (8).

$$(p \mid \alpha*, \beta*) = Be(\alpha*, \beta*) \quad (8)$$

$\alpha*$ and $\beta*$ is respectively the new number of selections and rejections for the specific node. Thus, obtaining the posterior distribution in (8) becomes simply a matter of adding new confirmations to the existing, and then calculating the mean (μ) and variance (var) (9,10).

$$\mu = \frac{\alpha}{\alpha+\beta} \qquad (9)$$

$$var = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \qquad (10)$$

In Figure 3, examples of Beta distributions for different values of α and β are shown. Distribution 1: $Be(\alpha = 1, \beta = 1)$ is a uniform distribution offering an uninformative prior with a mean, $\mu = 0.5$ and a high variance (uncertainty) due to the low sample size. In this case, the posterior will largely be determined by the data. Distribution 4: $Be(\alpha = 30, \beta = 5)$ has $\mu = 0.86$ and a smaller variance, thus providing a comparably less uncertain estimate of the user selection preference, p. The sequence of graphs 1-4 in Figure 3, can be seen as an example of a continuous learning cycle, starting with no knowledge of user selection (a uniform distribution with no samples) towards more informative distributions 2-4 as the sample size increases. When a new node is created with no samples available ($\alpha = \beta = 1$), the Beta distribution is uniform. However, to avoid the uninformative uniform distribution we propose to query the user to provide a subjective estimate of the selection (the mean) of this node along with a confidence level (the variance). Using the equations for the mean (9) and variance (10), suitable values for α and β can then be calculated.

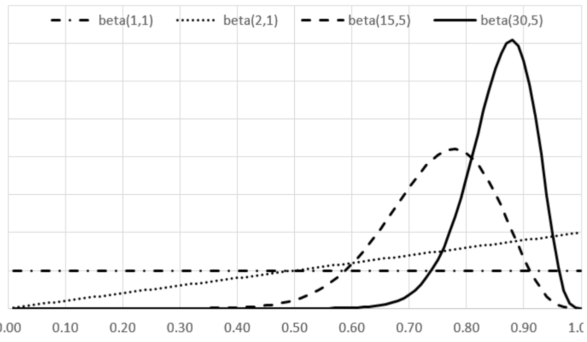


Figure 3. Four $Be(\alpha, \beta)$ distributions for different values of α, β. Note that $\alpha, \beta > 0$, thus $\alpha = \beta = 1$ is equal to no samples. 1: $Be(1,1)$, $\mu = 0.50$, $var = 0.083$. 2: $Be(2,1)$, $\mu = 0.67$, $var = 0.056$. 3: $Be(15,5)$, $\mu = 0.75$, $var = 0.0089$. 4: $Be(30,5)$, $\mu = 0.86$, $var = 0.0034$.

E. *Introducing evidence*

The Bayesian network described in section V.B and Fig. 2 receives evidence in the form of Anomaly information gathered by SA. In the example shown in Figure 2, SA has detected that the gripper was unexpectedly fully open (thus providing evidence that Gripper Open = true, Gripper Closed = false. The evidence is in practice introduced to the network by clamping the two nodes to their respective values.

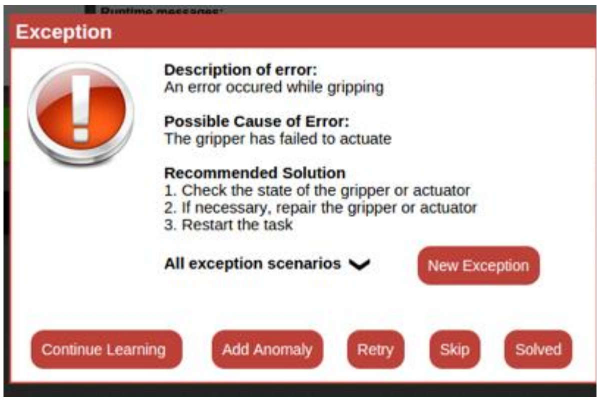
F. *Posterior probabilities*

After introducing evidence, posterior probabilities for all nodes are calculated. We have used the SMILE reasoning engine [9] for inference. The Response having the highest posterior probability is selected as the suggested solution. For each Response, the tree is descended towards the root Anomaly nodes, thus mapping out each possible path towards the root. The resulting list will have the most probable ES listed first with all other less likely alternative ES following in descending probability.

VI. User Interface for Error Recovery

While section V discussed the inner working of the actual mapping process, we focus on a more user-centric view in this section.

Whenever an anomaly is detected, the error layer classifies it into an error cause. If no cause is found the user is inquired and given the option to assign an existing cause, dismiss the anomaly as not indicating an error or to create a new cause (including a resolution, if known). Figure 4 shows the dialog box after successfully mapping an anomaly to an error and further to a recovery action. The user can accept


Exception

Description of error:
An error occured while gripping

Possible Cause of Error:
The gripper has failed to actuate

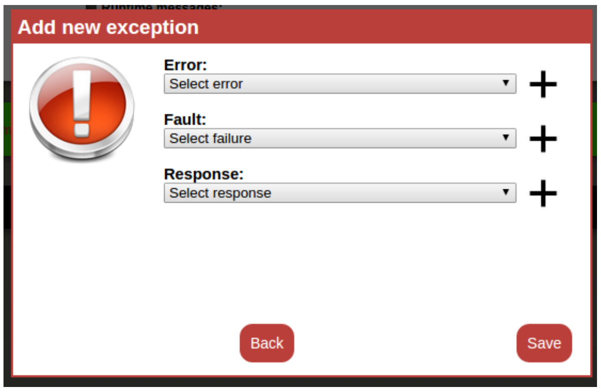
Recommended Solution
1. Check the state of the gripper or actuator
2. If necessary, repair the gripper or actuator
3. Restart the task

Figure 4 The system identified an error including a recovery action. In case of misclassification the user can add a new exception or dismiss the anomaly as not indication an error (button "Continue Learning").

this solution or add a new solution. Figure 5 shows the corresponding dialog box for adding a new triplet of error, error cause and recovery action. The dialog boxes shown in Fig. 4 and 5 are designed for use at system runtime and therefore as simplistic as possible. A more elaborated interface for managing the entire network of anomalies, errors, causes and recovery actions is also provided and targeted at specifically trained users that setup a new robot application.

VII. Experimental Evaluation

Within the scope of SMErobotics, this framework has been intensively evaluated using various experiments. A detailed example is the failure to grasp as described in the following section. We have tested the system's ability to learn the preference of selecting a solution by manually introducing the Gripper Open (GO) Anomaly, shown in Fig. 6, during the execution of a skill. In this test, we have tested the system's ability to learn the user preference of selecting the Repair Actuator (RA) Response over the Replace Pneumatics (RP) Response. For the purpose of the test, the system had initially no knowledge of user selections (samples), except for five samples confirming the choice of the RP Response as the user preferred solution to the GO


Add new exception

Error: Select error

Fault: Select failure

Response: Select response

Figure 5 Adding a new error cause or fault to the system.

Anomaly. Hereafter, we introduced the GO Anomaly repeatedly, selecting the RA Response as the solution each time. This process was repeated until EH started to suggest the RA Response, thus demonstrating EHF's ability to learn the user preference of selecting the RA Response over the RP.

Test results are shown in Fig. 6. Initially, EH has five samples confirming the selection of the RP Response for the

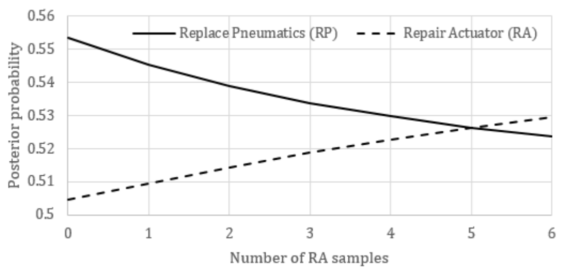


Figure 6. Posterior probabilities for solution nodes Replace Pneumatics (RP) and Repair Actuator (RA) to a Gripper Open Anomaly. The solid line represents the posterior probability of the RP and the dotted line represents posterior probability of RA. For one sample of RP, it takes EH two samples of RA to learn the user preference of selecting RA.

GO Anomaly. Thus, when the GO Anomaly is introduced, EH suggests RP as the most suitable Response to the GO Anomaly with probability ~ 0.553. However, the user ignores the EH suggested RP Response and instead selects RA. Thus, when the GO Anomaly is introduced again, EH now has six samples (five for RA and one for RP), computing the most likely Response to be RP with probability ~ 0.545, and so on. At RA sample 5, EH computes the probability for each Response being identical (~ 0.526). Again, the GO Anomaly is introduced and this time EH suggests the RA Response with posterior probability ~ 0.535. Thus, with five samples confirming RP, it took six samples of RA for EH to suggest RA.

VIII. Conclusion and Future Work

Through the test results in section VII, we showed that EH is able to learn the user preference of selecting a solution, even when it had learned a different preference earlier. As the user selects a specific solution to an Anomaly, the solution becomes more probable for future selection. This is normally helpful, but can be problematic if the user wishes the system to select a different solution, since learning a new preference can take several iterations, as the test results showed. This is especially true when the sample count for the prior solution is high. A possible future solution could be to introduce additional information in the Error Layer, e.g. condition the error cause not only on the counting of user selections, but also on the state of various system variables at time of the user selection.

The system is currently being integrated in further demonstrators in the context of the SMErobotics project and will see more in-depth testing and possibly enhancements in these demonstrators. Concept videos of these showing the SMErobotics vision of future industrial robotics are available at http://video.smerobotics.org; especially the D2 and D3 videos are relevant in the context of this work.

References

[1] R. H. Andersen, “Definition of Hardware-Independent Robot Skills for Industrial Robotic Co-Workers”, *ISR*, 2014

[2] R. H. Andersen , L. Dalgaard , A. B. Beck, J. Hallam, “An Architecture for Efficient Reuse in Flexible Production Scenarios”, Accepted for IEEE Int. Conf. on Automation Science and Engineering (IEEE CASE 2015)

[3] A. Avizienis, et al, “Basic concepts and taxonomy of dependable and secure computing,” *IEEE Transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.

[4] A. B. Bech, “Situation Assessment for Mobile Robots”. PhD thesis, DTU Electrical Engineering, Danish Technological Institute, 2012.

[5] A. B. Beck, C. Risager, N. A. Andersen, O. Ravn, "Spacio-Temporal Situation Assessment for Mobile Robots," in 14th International Conference on Information Fusion (FUSION), 2011.

[6] S. Bruning, et al, "A Fault Taxonomy for Service-Oriented Architecture," *High Assurance Systems Engineering Symposium,* 10th IEEE , vol., no., pp.367,368, 14-16, Nov. 2007

[7] J. Carlson, R. R. Murphy, "How UGVs physically fail in the field," *IEEE Transactions on Robotics*, vol.21, no.3, pp.423,437, 2005

[8] K.S.M Chan, et al., “A Fault Taxonomy for Web Service Composition,” *Service-Oriented Computing - ICSOC 2007 Workshops*, *Lecture Notes in Computer Science*, pp 363-375, 2009

[9] J. M. Druzdzel, "SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: a development environment for graphical decision-theoretic models." AAAI/IAAI. 1999.

[10] M. Dunham et al, ”Extensible Markov Model”. *Fourth IEEE International Conference on Data Mining*, 371-374, 2004

[11] N. Fenton, M. Neil, ”Risk Assessment and Decision Analysis with Bayesian Networks”, First edn., CRC Press, 2013

[12] T. Gecks, D. Henrich, "Human-robot cooperation: safe pick-and-place operations.", *IEEE International Workshop on Robots and Human Interactive Communication*. 2005.

[13] S. Haddadin, et al. ”Towards the Robotic Co-Worker,” *The 14th International Symposium ISRR*, pp 261-282, 2011.

[14] J. Huckaby, H. I. Christensen, “Toward a knowledge transfer framework for process abstraction in manufacturing robotics,” in *ICML Workshop on Theoretically Grounded Transfer Learning*, 2013.

[15] C. Kemp, et al, "Challenges for robot manipulation in human environments [Grand Challenges of Robotics]," *Robotics & Automation Magazine, IEEE* , vol.14, no.1, March 2007

[16] M. Klein, C. Dellarocas; “Knowledge-based Approach to Handling Exceptions in Workflow Systems”, *Computer Supported Cooperative Work,* Volume 9, Issue 3-4, pp 399-412, 2000

[17] P. Loborg, “Error recovery in automation: An overview,” in *Proc. AAAI.*, Stanford, CA, pp. 94–100, 1994

[18] J. Pearl, “Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference”, First edn, Morgan Kaufmann, 1998.

[19] M. D. Schmill, et al, “The Role of Metacognition in Robust AI Systems,” *AAAI-08 Workshop on Meta-reasoning*, Chicago, 2008

[20] J. Shah, et al, “Improved human-robot team performance using chaski, a human-inspired plan execution system,” in *Proceedings of the 6th international conference on Human-robot interaction*, 2011.

[21] G. Steinbauer, “A Survey about Faults of Robots Used in RoboCup,” *RoboCup 2012: Robot Soccer World Cup XVI, Lecture Notes in Computer Science Volume*, pp 344-355, 2013

[22] D. Stengel et al. "An Approach for Safe and Efficient Human-Robot Collaboration.", *The 6th International Conference on Safety of Industrial Automated Systems*. 2010.

[23] C. Syan, Y. Mostefai, "Status monitoring and error recovery in flexible manufacturing systems", *Integrated Manufacturing Systems*, Vol. 6 Issue 4, pp.43 – 48, 1995