The 7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
12-14 September 2013, Berlin, Germany

Policy-based Self-configuration of Autonomous Systems Information Inputs

Michal Podpora [1] , Aleksandra Kawala-Janik[2] , Mariusz Pelc[1, 2]
[1] Opole University of Technology, Faculty of Electrical Engineering, Automatic Control and Informaics, Opole, Poland, michal.podpora@gmail.com (www.po.opole.pl)

[2] University of Greenwich, School of Computing and Mathematical Sciences, London, United Kingdom, kawala84@gmail.com, m.pelc@greenwich.ac.uk (www.gre.ac.uk)

***Abstract* – In this paper the idea of Policy-based Input Selector, which is a framework of a highly flexible modular software interface was presented. The proposed Policy-based Input Selector allows run-time context-sensitive and self-acting reconfiguration of information inputs in cognitive systems. The proposed solution has great application capability in autonomous and intelligent systems with multiple data inputs, such as – inter alia – vision, Laser Range Finders (LRF) or audio. Potential implementation of policies enables efficient context switching of information input and this was in detail presented and explained in this work.**

***Keywords* – intelligent information systems; distributed systems; autonomous systems; autonomic computing**

I. Introduction

An example of implementation of autonomic politic system is a mobile robot. A well-designed autonomous mobile robot is able to navigate in environment, which enables recognition of basic shapes and objects. Autonomous decisions are made by the robot by choosing predefined actions and recovering from unexpected situations.

Modern mobile robots contain multiple sensors and data acquisition modules in order to provide all possible kinds of data information inputs, however they are only able to treat only one of the modules at time – the most important one. In cheaper hardware implementations these are most often – ultrasonic sensors, where in more expensive and more sophisticated products – laser range finders (LRF). In both cases, no matter if there is a simple, cheaper solution or a complex, sophisticated one, there is one thing in common – the favoured input is unable to fail, but it also cannot be neither substituted nor disabled. The robot is considered to be not operational without it. The vast majority of advanced autonomous robot implementations is designed in the way, that they use one of its inputs as the preferred, favoured one.

Robots, and in particular – autonomous and intelligent robots, would be much more fail-proof if they were able autonomously and in run-time mode to reconfigure (or replace) the input processing procedures in order to change the preferred data input source. This requires a cognitive system design that may be implemented for the memory and inference system of the robot. This system would contain sufficient level of information abstraction and enable to address the possibility of Invariant Representations stored in the information system [1].

The main goal of this research is to formulate and propose a framework for run-time reconfigurable inputs of an autonomous system.

II. Policy-based Input Selection

A. Policy-based Systems

Primarily, policy-based systems were designed to support run-time reconfigurability of systems decision making logic. Policy-based computing describes a methodology for embedding dynamic behavior into software components [2].

The most important advantage of this type policy-reconfigurable software component is that the malfunction of one of the data acquisition subsystems does not immobilise the whole system. Instead it also enables another policy loading, so the the incomplete input data can be managed. Such a system might not be the most optimal solution, but is far more reliable, as it could report the malfunction, while still running all critical functions.

In the Fig. 1 a simplified scheme of policy-based dynamically-configurable logic was illustrated. The applied logic can be replaced in a run-time mode. It is important to mention, that both inputs and outputs cannot be modified in any way.

B. Middleware Run-time Logic Replacement in Policy-based Control Systems

Over the last few years policy-based computing was applied to computer control systems, offering not only Boolean logic decisions, but also fuzzy inputs and outputs [3]. This in turn leaded to brand new applications of control systems with modular and dynamically-configurable logic. In the Fig. 2 block scheme of policy-based control

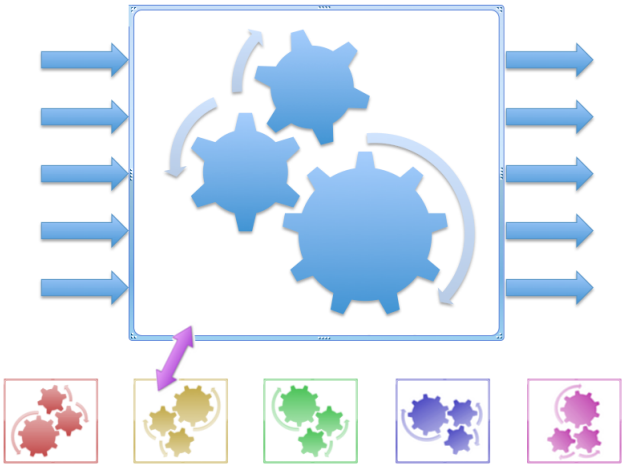

Figure 1. Policy-based dynamically-configurable logic.

systems framework was presented.

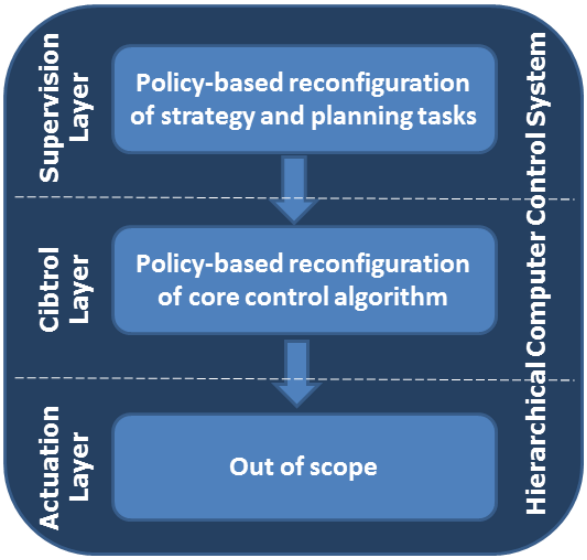

Figure 2. Policy-based control systems framework [3].

Policy-based control systems are more efficient with the implementation of combination of various different control strategies (policies). This makes them more reliable (fail-safe), what can be even more improved by the implementation of additional 'emergency policies' for incomplete data input [3].

The application of the policy-based computing in a control system enhances the reliability and stability of the system, but more significant is that it enables reconfiguration of system logic in a run-time mode. The system does not need neither recompilation nor reinitialisation.

C. Policy-based Input Selection in Autonomous Mobile Robots and Systems

An alternative definition of autonomic systems can be specified together with introduction of a new field of research, which is a certification of Autonomic Systems. Autonomic Systems as such are designed to mimic the capabilities of biological systems. They are programmed to evolve and adapt, and to evaluate the trust measure [4].

Real autonomous systems should be able to manage unexpected exceptions and failures of any subsystem and to deal with various kinds of interactions which may occur between these (sub)systems [5]. Decisions made by and behaviour of an autonomous system can be secured by implementation of actions for any exception (also for new, not managed, exceptions). Unfortunately the failure of data acquisition subsystem usually is unrecoverable. Although, if the sensors (data inputs) of an autonomous mobile robot system would be interfaced with a modular run-time replaceable interconnection, it would be possible to switch the data information input. This can be achieved by implementing a policy-based fuzzy control system outlines to a new middleware interfacing data inputs and system. Such a system becomes capable of temporarily turning off a malfunctioning acquisition subsystem for replacement and maintenance purposes or switching to a concurrent/alternative information inputs in order to complete navigation or any other action.

Policy-based input selection and switching can be easily achieved by implementation of the basic functionality of policy-based system wrapper to an interface between sensors and system (Fig. 3).

Figure 3. Policy-based Input Selector for Autonomous and Intelligent Systems.

The wrapper is responsible for loading and switching logic of the Input Selector, depending on system acquisition availability and quality of the input data. For instance, if the LRF data acquisition subsystem is defective, the Policy-based Input Selector can switch the preferred data input to vision. On the other hand, vision is not always the best source of data, depending on the environment

parameters. Smoke, fog or any obstacle can make the vision to become a useless data source, in such case the inferring system can force the Input Selector to change the current policy by loading another one, such as infrared camera data input.

D. Backside and Requirements

Policy-based Input Selector can be implemented in any programming language and in any hardware input configuration. The principles and design are similar to a wrapper in policy-based fuzzy control systems. However, design of an autonomous mobile robot system using Policy-based Input Selector is a difficult challenge, because of the data abstraction level. The inputs are not equal until the highest level of abstraction, e.g. 'a fast-moving object on the left, 60%: a ball'. If the whole object recognition together with the inferring algorithms are supposed to be runing on an embedded hardware, developing such a system will be undoubtedly technically possible. On the other hand, if the mobile platform uses embedded hardware only for critical tasks and the cognitive architecture is accessed remotely (Fig. 3), it might be extremely difficult to define the proper outpupts borderline of the wrapper (i.e. inputs of the cognitive system). In that case, the Input Selector can be implemented only when lowering the expected abstraction level towards the inputs of the system. This in turn implies limitation of universality. The essential issue while implementing the Policy-based Input Selector, is to find a compromise between limited universality of data inputs and lowered abstraction level. In practice, it is important to split the input data processing algorithm into the two following parts:

- hardware- and source-dependent processing,
- output-oriented processing.

The information, between these two parts, is hardware-independent enough to be transferred to remote system, although it is not universal yet. In practical the implementation data sources are grouped depending on information type:

- visual information,
- depth information,
- audio information,
- hardware parameters information of a robot.

The implemented Policy-based Input Selector is capable to switch a policy within a particular information type group (e.g. changing preferred input policy from 'visual-RGB-information' to 'visual - infrared - information' or from 'depth - LRF - information' to 'depth - sonar - information'). The outputs of Input Selector put out not only preprocessed data information, but also some basic knowledge inferred from the information using fuzzy values.

Of course, not all source combinations are rational, especially between different information type groups (e.g. there is no point in automatic replacing audio input with sonar input) and not all are computationally reasonable (e.g. replacing LRF depth data with stereo vision depth data, while it requires completely different input data processing procedures).

Figure 4. Policy-based Input Selector must be implemented in the robot embedded hardware to prevent off-line malfunction.

The embedded hardware of the mobile robot should not be used extensively – only the basic data manipulation procedures should be performed (i.e. preparing a universal-format data frame for transmission and performing a simple data quality evaluation, e.g. a brightness threshold for day/night detection or an image histogram for smoke/fog detection) (Fig. 4).

The input data quality evaluation procedures should be proposed, and implemented individually for every policy, with respect to the needs and specificity of the case. The PIS-based systems give the opportunity to use linguistic variables for describing input data quality, which extends their usability and flexibility by joining the fuzzy systems family.

III. Discussion

Policy-based computing and policy-based control systems can be applied to information inputs of a mobile robot as a Policy-based Input Selector. Policy-based Input Selector is capable of automatic policy swapping (information source switching) in run-time in case of input hardware malfunction or any acquisition subsystem error. Furthermore, Policy-based Input Selector is capable of automatic policy changing if the input signal is too noisy or clearly erroneous. Lastly, Policy-based Input Selector can switch policies 'on-demand' - if the inferring algorithm decides to do so. As of now, the last operating mode of Policy-based Input Selector has not been tested, because implementation of the inferring algorithm as well as cognitive distributed system is still in progress. Advantages of using Policy-based Input Selector instead of simply connecting inputs to system are significant:

- fail-safe inputs for hardware issues,
- fail-safe inputs for acquisition subsystems stability issues,

- run-time automatic emergency switching to other information source,
- run-time automatic switching to a better information source,
- run-time on-demand switching to a specific information source.

Authors have validated the Policy-based Input Selector framework conception by implementing a trial PIS System, containing two acquisition subsystems:

- computer vision (RGB color) acquisition subsystem,
- infrared vision acquisition subsystem.

The trial implementation included a simple policy (control strategy for the PIS) using constant threshold quality parameters for both inputs. Since the quality parameters and/or specific reconfiguration policies are completely user-definable, they are not in the scope of this paper. They are defined, implemented and evaluated by a PIS System developer – depending on a desired strategy.

However, the trial implementation was important for authors, while it showed that:

- the subjective 'quality' of an input can be initially evaluated in an embedded system,
- it is possible (and potentially beneficial) to implement the possibility to change the definition (algorithm) of 'quality',
- it is possible to provide the modularity of PIS System to enable the possibility to change policies in runtime.

IV. Future Work

The further step of this research involves the implementation of a distributed cognitive system. The distributed cognitive system will be able to store the hierarchical information regarding robot environment and semantic network of possible actions.

After carrying out some initial tests – full implementation will be applied.

References

[1] *https://www.numenta.com/htm-overview/education /HTM/CorticalLearningAlgorithms.pdf* Numenta Inc., Whitepaper – available on-line, *Hierarchical Temporal Memory Cortical Learning Algorithms*. Numenta Inc., 2011.

[2] Ward P., Pelc M., Hawthorne J., Anthony R., *Embedding dynamic behaviour into a self-configuring software system*. ATC 2008, Lecture Notes in Computer Science, Springer Verlag, 2008.

[3] Pelc M., *Policy-Based Reconfiguration of the Computer Control Systems*. Science Notes no. 335, Opole University of Technology, Opole – Poland, 2013.

[4] Shuaib H, Anthony R. J., Pelc M., *A Framework for Certifying Autonomic Computing Systems*. The Seventh International Conference on Autonomic and Autonomous Systems: ICAS 2011 Venice/Mestre, Italy, IARIA, 2011.

[5] Anthony R., Pelc M., Shuaib H., *An Interoperability Service for Autonomic Systems*. International Journal on Advances in Intelligent Systems, vol. 5, no. 3 & 4, 2012.