High Dynamic Range Image Based on Multiple Exposure Time Synthetization

Annamária R. Várkonyi-Kóczy

Dept. of Measurement and Information Systems, Budapest University of Technology and Economics, Budapest, Hungary,
Integrated Intelligent Systems Japanese-Hungarian Laboratory
koczy@mit.bme.hu

András Rövid

Dept. of Measurement and Information Systems, Budapest University of Technology and Economics, Budapest, Hungary,
Integrated Intelligent Systems Japanese-Hungarian Laboratory
Dept. of Electrical and Electronics Engineering, Shizuoka University, Japan
tarovid@ipc.shizuoka.ac.jp

Szilveszter Balogh

Dept. of Measurement and Information Systems, Budapest University of Technology and Economics, Budapest, Hungary,
Integrated Intelligent Systems Japanese-Hungarian Laboratory

Takeshi Hashimoto, Yoshifumi Shimodaira

Dept. of Electrical and Electronics Engineering, Shizuoka University, Japan

Abstract: High dynamic range of illumination may cause serious distortions and other problems in viewing and further processing of digital images. In this paper a new tone reproduction preprocessing algorithm is introduced which may help in developing hardly or non-viewable features and content of the images. The method is based on the synthetization of multiple exposure images from which the dense part, i.e. regions having the maximum level of detail are included in the output image. The resulted high quality HDR image makes easier the information extraction and effectively supports the further processing of the image.

Keywords: high dynamic range images, image reproduction, image processing

1 Introduction

Digital processing can often improve the visual quality of real-world photographs, even if they have been taken with the best cameras by professional photographers in carefully controlled lighting conditions. This is because visual quality is not the same thing as accurate scene reproduction. In image processing most of the recently used methods apply a so-called preprocessing procedure to obtain images which guarantees – from the point of view of the concrete method – better conditions for the processing. A typical example is noise elimination from the images which yields much better results as else. There are many kinds of image properties to which certain methods are more or less sensitive [1]. The different image regions usually have different features. The parameters of the processing methods in many of the cases are functions of these image features.

The light intensity at a point in the image is the product of the reflectance at the corresponding object point and the intensity of illumination at that point. The amount of light projected to the eyes (luminance) is determined by factors such as: the illumination that strikes visible surfaces, the proportion of light reflected from the surface and the amount of light absorbed, reflected, or deflected by the prevailing atmospheric conditions such as haze or other partially transparent media [2].

An organism needs to know about meaningful world properties, such as color, size, shape, etc. for the interpretation of the view. These properties are not explicitly available in the retinal image and must be extracted by visual processing.

In this paper we will deal with the reproduction of images when the high dynamic range of the lightness causes distortions in the appearance and contrast of the image in certain regions e.g. because a part of the image is highly illuminated looking plain white or another is in darkness.

The dynamic range in photography describes the ratio between the maximum and minimum measurable light intensities. High dynamic range (HDR) imaging covers a set of techniques that allow a far greater dynamic range of exposures than normal digital imaging techniques [3]. HDR images enable to record a wider range of tonal detail than the cameras could capture in a single photo.

Recently HDR imaging techniques have come into the focus of research because of their high theoretical and practical importance. The application of HDR capable sensors in cars may get an important role in traffic safety, because in many of the cases the environment around the car has high dynamic range (dark and bright) illumination. (Just consider the case when a car leaves a (dark) tunnel and enters the (bright) sunshine.) An HDR sensor, in contrast to a linear sensor, can detect details that bright environment washes out and it misses fewer details in dark environment [3].

Another example can be the application of HDR techniques in the preprocessing phase of images, by which the performance of different image processing algo-

rithms like corner and edge detectors or scene reconstruction algorithms can be improved.

Several methods can be found in the literature addressing the problem. Each of them tries to compress the high dynamic range of luminance values to a displayable range keeping as much amount of information as possible. Just to mention some characteristic approaches, Rubinstein and Brooks's method in [4] is based on fusion in the Laplacian pyramid domain. The core of the algorithm is a simple maximization process in the Laplacian domain. Wide dynamic range CMOS image sensors play also very important role in HDR imaging (see e.g. Kawahito's work in [5]). The sensor introduced in [5] uses multiple time signals and such a way extends the image sensor's dynamic range. Reinhards in [6] applies a so-called zone system. The authors of this paper also have introduced new fuzzy-based tone reproduction algorithms in [7], [8].

2 Gradient Based Multiple Exposure Time Synthetization Algorithm

2.1 The Basic Concept

Given N images of a static scene obtained at different exposures using a stationary camera. The introduced method combines the images into a single image in which each of the input image information is involved without producing noise. The main idea is the following:

Each image is segmented into small local regions, each of the same size. The shape of these regions is rectangular. The proposed method selects the most informative image for each local image region. The main task is to identify the image that contains the highest density of information within a local region.

When the dynamic range of the scene is high taking just one photo using a normal camera is not enough for producing a HDR image. In such cases several pictures are needed to capture all the scene details. These images should be merged together such a way, that all the involved information should be preserved.

If the scene contains regions with high luminance values, then in that highly illuminated region it is necessary to take a picture with low exposure time for the visualization of the details. On the other hand, if the scene contains very dark areas, then the exposure time should be possibly much higher. Approaching from the opposite site, if we have images with different exposures we have to decide somehow which exposure contains the maximum level of information in case of a certain region. There are existing methods, which use statistical elements for se-

lecting the most informative image region while others apply the histogram of the luminance values of the processed region.

According to our experiments, the level of the details in a region can be measured based on the sum of gradient magnitudes of luminance in that region. The complexity of this approach is lower than that of the other ones, thus the processing time can also be reduced. As detailed is the information as higher the sum of the gradient values is in that region.

The main step is to find the segments with the highest sum of gradients of luminance values. After merging the segments, as the output of this task we get an image, which contains the maximum amount of information in each region.

Another important step is smoothing, because we have to eliminate the sharp transitions, which arise at the borders of the regions. A monotonically decreasing blending function is centered over the selected regions of the input images and the pixel intensities are weighted according to the corresponding blending function values. The blending function assigns the maximum weight to the pixel located at the center of the considered region. To the other pixels, weights inversely proportional to the distances from the center of the region are assigned. After this procedure we get the output image which will not contain discontinuities along the image regions.

2.2 Measuring the Level of the Detail in an Image Region

For extracting all of the details involved in a set of images of the same scene made with different exposures, it is required to introduce a factor for characterizing the level of the detail in an image region. For this purpose we propose the gradient of the intensity function corresponding to the processed image and a linear mapping function, which is applied for setting up the sensitivity of the measurement of the detail level. In the followings the description of the estimation of the mentioned factor is introduced.

Let $I(x,y)$ be the pixel luminance at location $[x, y]$ in the image to be processed. Let us consider the group of neighboring pixels which belong to a 3x3 window centered on $[x, y]$. For calculating the gradient of the intensity function in horizontal ΔI_x and vertical ΔI_y directions at position $[x, y]$ the luminance differences between the neighboring pixels are considered:

$$\begin{aligned} \Delta I_x &= \left| I(x+1, y) - I(x, y) \right| \\ \Delta I_y &= \left| I(x, y-1) - I(x, y) \right| \end{aligned}. \tag{1}$$

For the further processing the maximum of the estimated gradient values should be chosen, which solves as the input of the normalized linear mapping function P defined as follows:

$$P(v) = v / I_{\max}, \tag{2}$$

where $I_{\max}$ is the maximum luminance value. (For 8 bit grayscale images it equals 255.)

Let **R** be a rectangular image region of width *rw* and height *rh*, with upper left corner at position $[x_r, y_r]$. The level of the detail inside of region **R** is defined as

$$M_D(\mathbf{R}) = \sum_{i=0}^{rw} \sum_{j=0}^{rh} P\left(\max\left(\Delta I_x(x_r + i, y_r + j), \Delta I_y(x_r + i, y_r + j)\right)\right) \tag{3}$$

As higher is the calculated M_D value as detailed the analyzed region is. In the followings we will use this parameter for characterizing the measure of the image detail.

2.3 HDR Image Synthetization

Let I_k denote the intensity function of the input image with index *k*, where *k*=1,…,*N* and *N* stands for the number of images to be processed, each of them taken with different exposure. Each image contains regions, which are more detailed as the corresponding regions in the other *N*-1 images. Our goal is to produce an image, which is the combination of the *N* input images and contains all details involved in all of the images without producing noise. Using such detailed image, the most of the feature detection methods can be improved and can effectively be used even if the lighting conditions are not ideal.

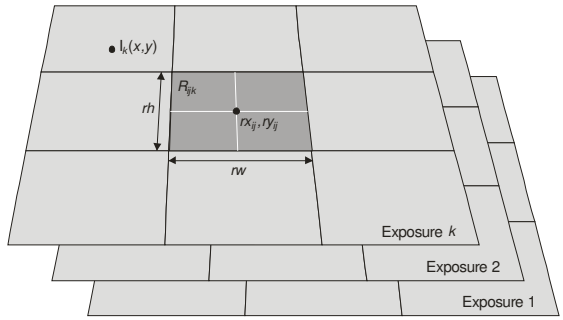


Figure 1
Image regions

The first step of the processing is to divide the pictures into *n* rows and *m* columns, which yields *nxm* rectangular image regions. The regions in the images are of the same size with height *rh* and width *rw* (see Figure 1). (In Figure 1 you can see a 3x3 division.) Let $\mathbf{R}_{ijk}$ denote the region in the *i*th row and *j*th column of the image with index *k*. Let rx_{ij}, ry_{ij} denote the horizontal and vertical coordinates of the center of the region in the *i*th row and *j*th column. $I_k(x,y)$ stands for the intensity of the pixel at position (*x*,*y*) in the image with index *k*.

For each image, the level of the detail has to be estimated inside every region $\mathbf{R}_{ijk}$. This information helps us to select the most detailed regions among the corresponding image parts (indexed by the same *i* and *j* values).

Let **D** denote the matrix of regions with the highest level of detail. Let d_{ij} be the element in the *i*th row and *j*th column of **D**, which stands for the index of the image, which has the most detailed region in the *i*th row and *j*th column, i.e.

$$M_D\left(R_{ijl}\right) > M_D\left(R_{ijk}\right) \mid k = 1...,l-1,l+1,...,N; l = d_{ij}\,. \tag{4}$$

The next step is to merge the regions $\mathbf{R}_{ijl}$. together, where $l=d_{ij}$ $i=1..n$, $j=1..m$. Merging the selected regions together results in an image that contains every detail involved in the *N* input images. Unfortunately, the resulted image usually contains sharp transitions along the borders of the regions. These sharp transitions should be eliminated. For this purpose the Gaussian blending function [4] can be applied advantageously, which has the form of

$$B_{ij}\left(x,y\right) = \frac{e^{-\left(\frac{\left(x-rx_{ij}\right)^2}{2\sigma_x^2}+\frac{\left(y-ry_{ij}\right)^2}{2\sigma_y^2}\right)}}{\sum_{p=1}^{m}\sum_{q=1}^{n} e^{-\left(\frac{\left(x-rx_{pq}\right)^2}{2\sigma_x^2}+\frac{\left(y-ry_{pq}\right)^2}{2\sigma_y^2}\right)}}\,. \tag{5}$$

Here *i* and *j* stands for the row and column indices of the region over which the Gaussian hump $G_{ij}(x,y)$ is centered (see Figure 2), *m* denotes the total number of columns and *n* the total number of rows in the input images. σ_x and σ_y stand for the standard deviation of the 2D Gaussian function. The values rx_{pq} and ry_{pq} represent the coordinates of the center of the region in *p*th column and *q*th row.

Let *U* be a function defined as

$$U(x,y) = \begin{cases} 1 & |x| \wedge |y| \le \varepsilon \\ 0 & \text{else} \end{cases}. \tag{6}$$

Function *U* is used for eliminating the influence of those segments, who's center points fall outside a predefined ε environment of the actually processed pixel.

Using the blending function (eq. (5)) and function *U* the luminance value of the output image can be evaluated according to

$$I_{out}\left(x,y\right) = \sum_{i=1}^{n}\sum_{j=1}^{m} B_{ij}(x,y)U(x-rx_{ij}, y-ry_{ij})I_{d_{ij}}(x,y). \tag{7}$$

The output luminance can be influenced by changing the size of the regions and the standard deviation of the Gaussian functions. As smaller is the standard deviation as higher influence the regions with low detail level onto the result have.

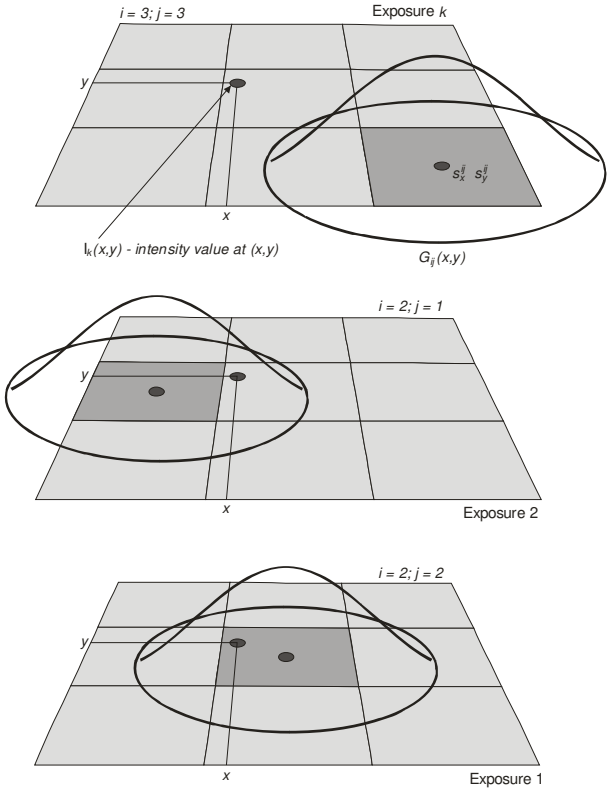


Figure 2
Illustration of the regions covered by Gaussian hump

Using such detailed image the edges can also be effectively extracted and advantageously be used by further processing, by object recognition, scene reconstruction etc.

In Figure 2 in case of exposure 1 the region in the 2nd row and 2nd column has the maximum level of detail compared to the other images, in case of exposure 2 the region in the 2nd row and 1st column while in case of exposure k the region in the 3rd row and 3rd column is the most detailed. The Gaussian functions are centered at (rx_{ij}, ry_{ij}) of the maximum level regions.

3 Illustrative Examples

The effectivity of the proposed algorithm is illustrated by two examples. In both of the examples the width and height of the regions are chosen to be the 1/15 part of the image width and height, respectively. Deviations σ_x and σ_y have the same value as the width and height of the regions, respectively.

Example 1: In Fig. 3 the original under-exposed image and its edge map can be seen. Fig. 4 represents the original over-exposed image and its edge map. In Fig. 5 the resulted HDR image using the proposed method and its edge map can be followed.

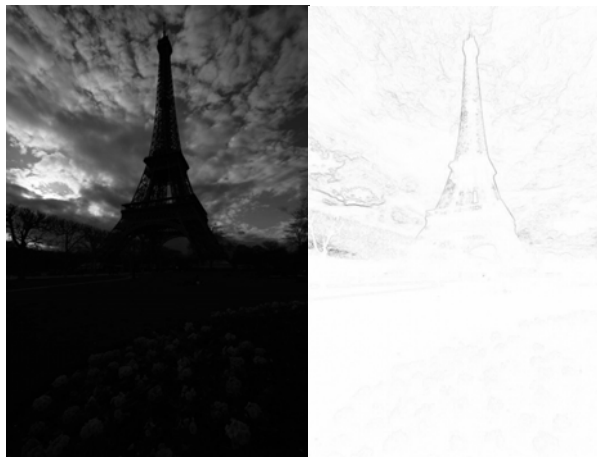


Figure3
Under-exposed image: Original image (left) and its edge map (right)

Figure 4
Over-exposed image: Original image (left) and its edge map (right)

Figure 5
Obtained HDR image: The resulted high dynamic range image (left) and its edge map (right)

Example 2: In Figs. 6(a)-(e) five different exposure images can be seen. Fig. 7 stands for the resulted HDR images using the proposed method.

(a) (b)

(c) (d)

(e)

Figure 6
Multiple exposure images

Figure 7
The obtained output HDR image (a) output image

Conclusions

In this paper a new gradient based approach is introduced for extracting image details. The method uses multiple exposure images of the same scene as input data. During the processing the images are divided into regions and always the most detailed region is chosen from the different exposure versions for the output image. With the help of this technique it becomes possible to produce a good quality HDR image from a set of bad quality photos taken by a range of exposures.

Acknowledgment

This work was sponsored by the Hungarian Fund for Scientific Research (OTKA T049519) and the Structural Fund for Supporting Innovation in New Knowledge and Technology Intensive Micro- and Spin-off Enterprises (GVOP-3.3.1-05/1.2005-05-0160/3.0).

References

[1] Russo, F.: Recent Advances in Fuzzy Techniques for Image Enhancement, IEEE Transactions on Instrumentation and Measurement, Vol. 47, No. 6, Dec. 1998, pp. 1428-1434

[2] Adelson, E. H., A. P. Pentland: The Perception of Shading and Reflectance, In D. Knill and W. Richards (eds.), Perception as Bayesian Inference, New York: Cambridge University Press, 1996, pp. 409-423

[3] Li, Y., L. Sharan, E. H. Adelson: Perceptually-based Range Compression for High Dynamic Range Images, Journal of Vision, Vol. 5, No. 8, August 2005, p. 598

[4] R. Rubinstein, A. Brook: Fusion of Differently Exposed Images, Final Project Report, Israel Institute of Technology, p. 14, 2004

[5] M. Sasaki, M. Mase, S. Kawahito, Y. Wakamori: A Wide Dynamic Range CMOS Image Sensor with Multiple Exposure Time Signals and Column-Parallel Cyclic A/D Converters, IEEE Workshop on Charge-Coupled Devices and Advanced Image Sensors, 2005

[6] Reinhard, E., R. E. Stark, M. Shirley, P. J. Ferwerda: Photographic Tone Reproduction for Digital Images, In Proc. of the 29th Annual Conference on Computer Graphics and Interactive Techniques, 2002, San Antonio, Texas, pp. 267-276

[7] Várkonyi-Kóczy, A. R., A. Rövid, P. Várlaki: Fuzzy-based Brightness Compensation for High Dynamic Range Images, In Proc. of the 9th International Conference on Intelligent Engineering Systems, INES 2005, Cruising on Mediterranean Sea, Greece, Turkey, Sept. 16-19, 2005, pp. 245-248

[8] Várkonyi-Kóczy, A. R., A. Rövid: High Dynamic Range Image Reproduction Methods, In CD-Rom Proc. of the 2006 IEEE Instrumentation and Measurement Technology Conference, IMTC/2006, Sorrento, Italy, April 24-27, 2006

New Design Objective and Human Intent-based Management of Changes for Product Modeling

László Horváth

Institute of Intelligent Engineering Systems, John von Neumann Faculty of Informatics, Budapest Tech, Bécsi út 96/B, H-1034 Budapest, Hungary
horvath.laszlo@nik.bmf.hu

Abstract: This paper is concerned with the evaluation of effects of modeled object changes at development of product in virtual space. Modeling of engineering objects such as elements and structures of products, results of analyses and tests, processes for production, and customer services has reached the level where sophisticated descriptions and modeling procedures serve lifecycle management of product information (PLM). However, effective utilization of highly associative product models is impossible in current modeling because techniques are not available for tracking and evaluation of high number of associative relationships in large product model. The author analyzed the above problem and considered inappropriate organization of product information as its main cause. In order to gain a solution in current modeling systems, a new method is proposed in this paper for change management. In this method, joint modeling of design objectives and human intent is applied for shape-centered products. As background information for the proposed modeling, paper discusses research results in change management for product development in modeling environments. Following this, integrated modeling of closely related engineering objects is proposed as extension to current industrial PLM systems. Next, design objective driven product change management is detailed. Finally, virtual space is outlined as a possible advanced application of the proposed change management with the capability of representation of human intent.

Keywords: product modeling, virtual space, change management, behavior-based modeling

1 Introduction

The success of industrial product modeling has led to highly integrated engineering systems called as product lifecycle management (PLM) systems. Design, analysis, production planning, marketing, and customer service applies the same software and data base system. PLM systems are being moved into Internet portals and they act as integrators in extensive project work at extended companies. This new style of engineering is based on integrated description of engineering objects in product model. While high-level software tools process

product models, effective communication between human and modeling procedure is not possible in current industrial modeling systems. Product models reached a complexity where application of conventional human-computer procedures does not provide means to survey consequences of a change in the highly associative product information. Engineers suffer from low-level support of decision-making processes. As a consequence, they do not able to survey huge number of related decisions, human intents, knowledge, and legislations.

The contribution of this paper is a new method for human – computer communication at decisions in engineering. Two essential model entities are introduced as means for representation of design objectives and human intents in their background. Based on these new elements of product model, new process is proposed for handling of changes of modeled objects during development and application of products. The proposed method is highly relied upon earlier researches by the author and other researches in product modeling and intelligent computing.

As background information for the proposed modeling, paper discusses research results in change management for product development in modeling environments. Following this, integrated modeling of closely related engineering objects is proposed as extension to current industrial PLM systems. Next, design objective driven product change management is detailed. Finally, virtual space is outlined as a possible advanced application of the proposed change management with the capability of representation of human intent.

2 Background

Product modeling can be summarized as definition, generation, and handling information for engineering activities during the entire lifecycle of products. Product model is heterogeneous because it describes different engineering objects such as components, analysis models, manufacturing processes, production schedules, and so on. For a consistent product model, it is essential to know the variation of modeled object parameters due to variations in other parameters. Typically, an engineering object parameter depends on high number of parameters of other objects as well as specifications. A great deal of study has been devoted to identify characteristics of engineering objects and current techniques in engineering modeling [1].

Important advancement towards efficient human-computer interaction (HCI) was application of form features at the definition of shape of mechanical parts during the nineties. Mechanical parts have shape-centered characteristic because any other information can be mapped to shape information. The author introduces research results for the application together with proven shape-centered modeling. Several researches were considered. The most relevant ones are cited as follows.

Form features are defined during construction a shape or recognized on a previously created shape. A method is given for multipurpose application oriented recognition of features in [2]. Other important area of model development is integrated product modeling for concurrent engineering. Typical product related engineering activities are integrated in concurrent integrated engineering in [3]. Reference models, resources, and application protocols are essential tools for the definition of integrated product models [4]. Description of behaviors of products in models represents initial efforts towards intelligent engineering modeling [5]. Broadening the application of computer systems at engineering was stimulated by advancements in digital computer principles [6].

In [7] Petri net is proposed as representation for design and implementation of an execution control, which, through suitable graph-search algorithms, generates sequences of task activation and deactivation operations, which execute the desired commands maintaining the system in admissible configurations. Environment composed by known and unknown elements are typical at certain applications. Availability of machine learning is essential in the handling of unforeseen environmental conditions. Robot controller can learn on-line about its own capabilities and limitations when interacting with its environment. A method is proposed in [8] where off-line supervised neurofuzzy learning and on-line unsupervised reinforcement learning, and unsupervised/supervised hybrid learning are applied at control of gripper. Application of Fuzzy methods is of great importance [9], among others at reduction of rule sets for the representation of corporate knowledge. Authors of [10] demonstrate that knowledge level based explanations of cognitive processes provided by traditional artificial intelligence and approach of embodied systems interacting with the real world in new AI can be unified. Author of [11] examined how UML, as the most widespread modeling tool for object-oriented software development, supports practical user interface development. He proposed application of the usage interaction model and the usage control model, each of which can be described by supplementing well-known UML diagrams. Modeling often serves special applications such as geometric modeling in reconstructing surgery [12].

Researches with participation of the author are cited as antecedents of the contribution in this paper below. The authors analyzed possibilities for extended application of the feature principle, adaptive processes and associative object definitions [13]. Other relevant researches are as follows.

- Application of product and other engineering object behavior definitions and adaptive actions in order to enhanced decisions [14].
- Modeling of human intent as background of engineering decisions [15].
- Integration of human intent descriptions in product models [16].
- Modeling for the handling of changes of engineering objects [17].

3 Closely Related Sets of Engineering Objects

The proposed change management requires modeling of human intent and product behaviors. This is evident because humans define model information according their intent and the aim of product development is realization of design objectives as behaviors. Moreover, humans apply knowledge according to their intent or execute intents of other humans of higher hierarchical position. For this purpose, three extensions to modeling in current industrial PLM systems are proposed (Figure 1) as follows.

- Integration of descriptions of closely related engineering objects.
- Extensions to object descriptions for human intent, associative object connections, adaptive actions, and product behaviors.
- Extensions to modeling procedures for the handling of HCI specific views in product data, collaborative connections, adaptive actions, and engineering specific browsing.

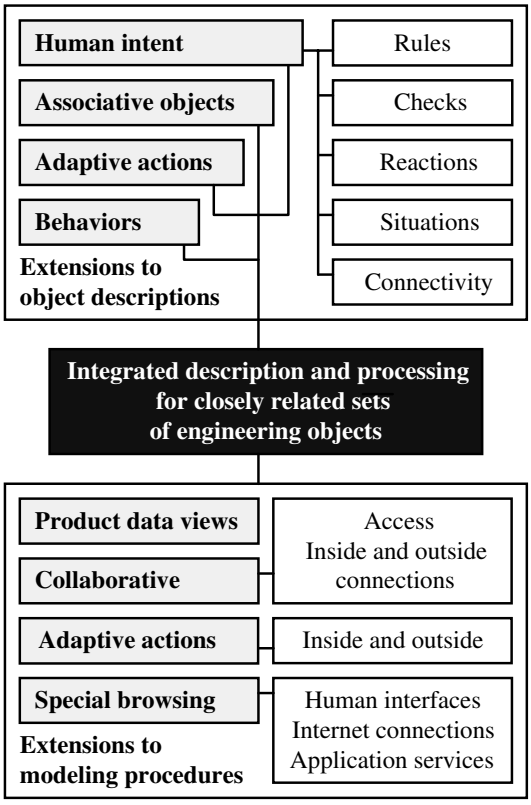


Figure 1
Essential approach to product modeling

Associative definitions connect objects inside and outside of a set of engineering objects. Human intent is described by using of one of the well-proven knowledge representations. Engineer-friendly representations are preferred such as rules, checks, reactions, situations, and networks.

Functional sectors of product information are connected, as it is outlined in Fig. 2. Outside world includes objects outside of an actual set of engineering objects. Changes of engineering objects are sensed from the outside world through associative object connections then are recorded as senses. Product model integrates object descriptions. Object descriptions are connected with behaviors. Behaviors are in close connection with knowledge for human intent and objectives for product development. Fulfilling the specified behaviors needs actions on inside and outside engineering objects.

In accordance with essential connections in Figure 2, essential modeling procedures are outlined in Figure 3. Information exchange processes serve communication between a set of closely related engineering objects and the world outside of it. Associative object definition manager is in connection with creation of model entities and action manager. Associative definition builder establishes new connections for behavior analysis driven action definition. Action definition initiates creation or modification of product model entities. Humans control behavior analysis and model creation procedures through human–computer interaction (HCI) procedures. All actions are coordinated and executed by action manager including actions concerning the outside world. Humans control HCI procedures. Internet portal organizes group work of engineers and provides means for collaboration amongst humans independent of geographical position.

4 Product Change Management for Design Objectives

Any attempt for changing any modeled product or product related engineering object must be evaluated for consequences on all associative objects before deciding its acceptance or rejection. This evaluation requires tracking along chains of associative objects in the product model. In this context, any modification of a product for its development, variant creation and correction is considered as change. Engineering on a product is a sequence of changes. Evaluation of a change analyzes appropriateness of a new proposed parameter value of a modeled engineering object for appropriate design objectives. This passive evaluation can be replaced by active evaluation to calculate parameters of modeled objects that results in behaviors in accordance with specified engineering objectives. Change of an engineering object parameter may affect one or more behaviors of the same and associative engineering objects. The author introduced the concept of change affect zone (CAZ) in order to define the set of potentially affected engineering

objects for change of an engineering object,. CAZ of an engineering object includes a set of engineering objects that may require modifications as a consequence of its change.

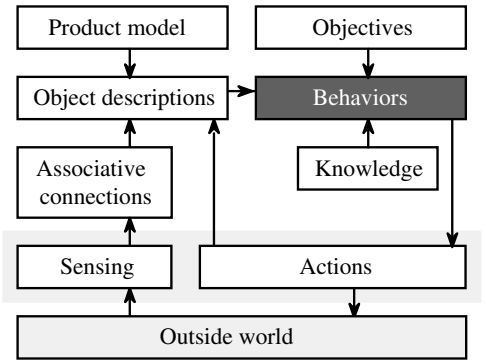


Figure 2
Essential connections of model information

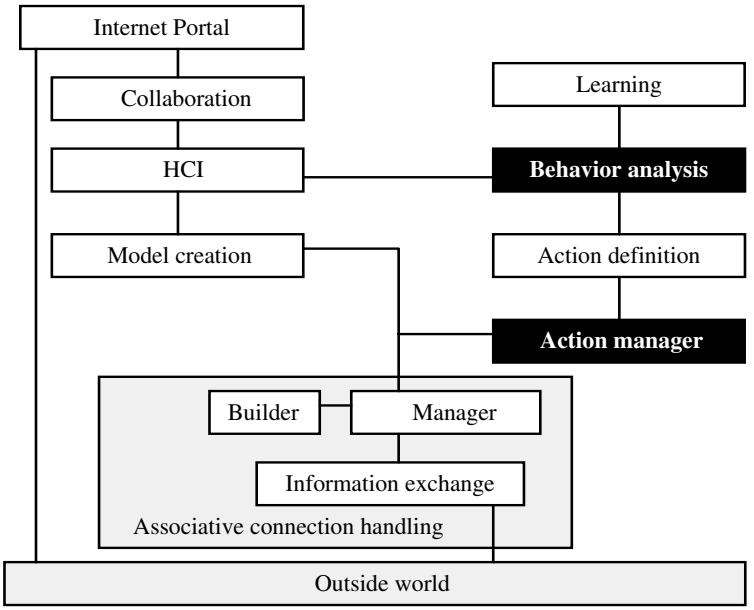


Figure 3
Essential modeling procedures

A behavior of an engineering object is defined by a set of its attributes. Any combination of values of these attributes results in a behavior with allowable or not allowable characteristics. Required or allowed combinations of parameters affecting characteristics of a behavior are specified as situations. A situation is

defined by a set of attribute values called as circumstances. Specifications of behaviors as design objectives for an engineering object consider customer demands, technical requirements, experiences, standards, legislations, and personal intents.

The author applies an extended definition of behavior for engineering objects. Figure 4 shows an example for the extension from shape centered product modeling. Characteristics of the shape became important factor during the past decade. Design objectives for shape, placing, continuity, and locality of a swept surface are specified by four behaviors, accordingly. Shape behavior is defined by situation for shape control. This situation comprises attributes of the swept surface serving modification of the shape. Other behaviors characterize placing of surface in a solid shape and its structural environment, continuity at its connections, and parametrically controlled local characteristics of the surface.

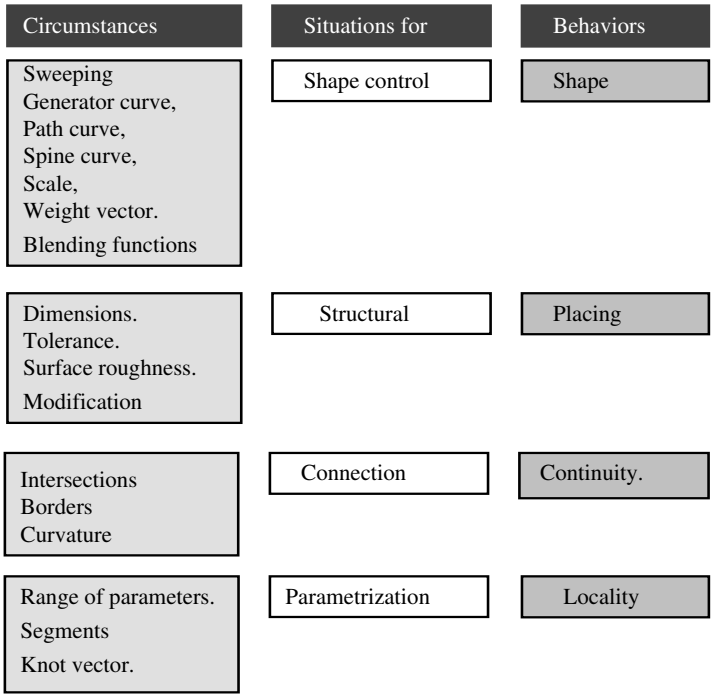


Figure 4
Example of surface behaviors

Essential management of object changes is outlined in Fig. 5. A set of closely related engineering objects are integrated in a complex engineering object. This object constitutes the inside world. All other engineering objects are in the outside world. Configuration of inside world is more or less flexible according to actual measures in the engineering organization. Model of a complex engineering object

includes information for elements, composition, situations, associative relations, and behaviors. Change management is supported by affect analysis. It handles adaptive actions. An adaptive action carries information for a change and may have four states. Change management receives information from interfaced humans and procedures about proposed and accepted changes. Its communication with outside world is about attempted and accepted changes. Change management generates new adaptive actions as consequences of proposed, attempted, and accepted changes. Adaptive actions with received, attempted and generated states are considered as conditional ones. Accepted adaptive actions are executed by inside and outside procedures.

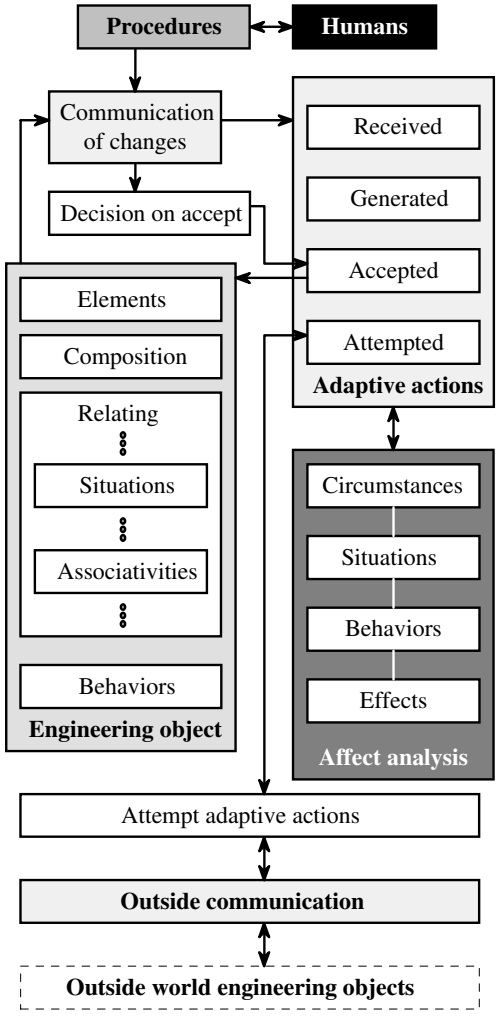


Figure 5
Management of changes of modeled objects

During acceptance procedure, changes are analyzed for their affects on attributes and behaviors of associative engineering objects. New adaptive actions are generated and affect analyzed along chains of associative engineering objects.

Acceptance procedure is continued during the entire product development and it is under control of responsible humans. Engineers have much more chance to find a conflict free solution than in conventional modeling. Change management acts as advanced navigator rather than design automata. The consequences of changes are often calculated directly, as modifications of elementary or composition objects. Sometimes simple changes are abandoned due to improper changes of behaviors. Alternatively, behaviors can be changed.

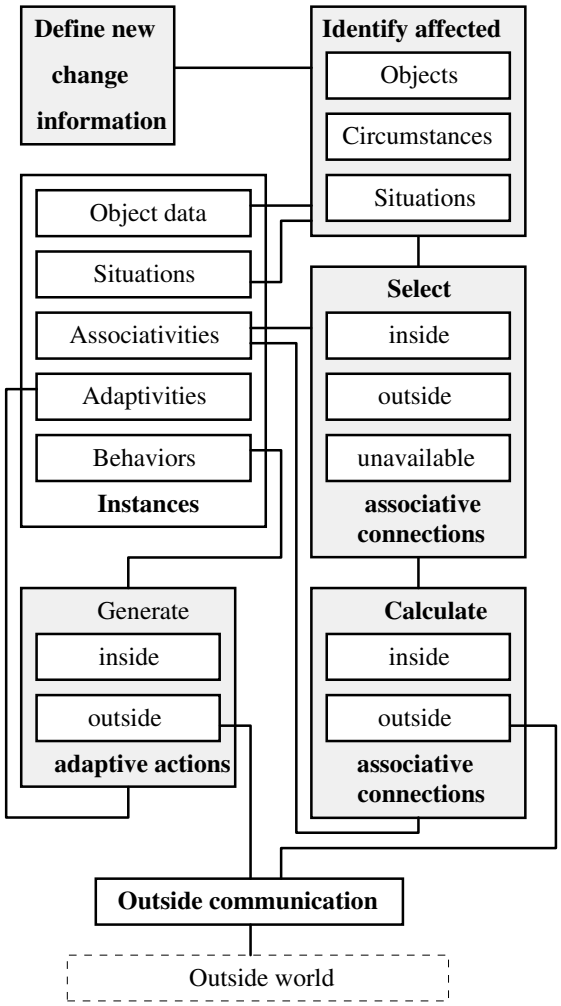


Figure 6
Processing of change information

Some details of change management are shown in Figure 6. Instances of generic descriptive object entities, situations, associative connections, adaptive actions and behaviors are defined. Process is briefed in the following.

- Engineering objects, circumstances, and situations are identified for behavior analysis.
- Inside, outside and unavailable associative connection definitions are selected for the identified model entities.
- Values for inside and outside associative connections are calculated and adaptive actions are generated.

Unknown associative connections must be considered and possibly defined by responsible humans.

The above discussion interpreted the product development as a series of decisions on changes towards product with specified behaviors. Decision support is summarized in Figure 7. It is connected to human sources, modeling procedures, and outside world items. Behavior analysis, creation of views, combination of intents, and change management are essential methods. Decision process uses conventional object data and extensions for the proposed modeling. These extensions are situations for behavior analysis, combined intents, effectivities for views, and types of effects. Additional items carry information about humans and outside links.

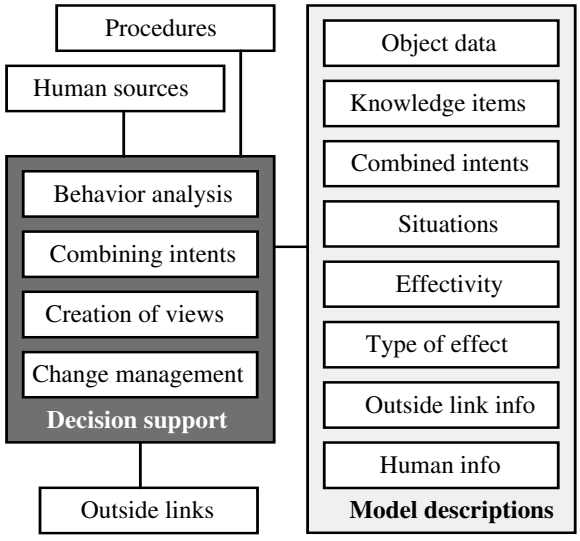


Figure 7
Support of decision-making

Decision on attributes of an engineering object may have complex human and knowledge background. It is often controlled by intent of two or more humans. Responsible engineer considers intent from research results, standards, legislation, local instructions, customer demands, and decisions of engineers on higher levels of hierarchy. At making these decisions, intents should be combined.

5 Concept of Virtual Space

Change management in the previous sector of this paper can support activities in a virtual space that is capable to accommodate computational intelligence as representation of human intent. Product model is developed in a virtual space where a development sector is responsible both for development of product descriptions and modeling capabilities (Figure 8). Other sectors of a virtual space are responsible for behavior, interface, and learning related activities. In the behavior sector, situations are generated, behaviors are analyzed, and rules are applied. Behaviors are created as engineering objectives under control of the development sector. Interface sector has the ability to receive and reacting effects. In the meantime, patterns and rules are learned for later application.

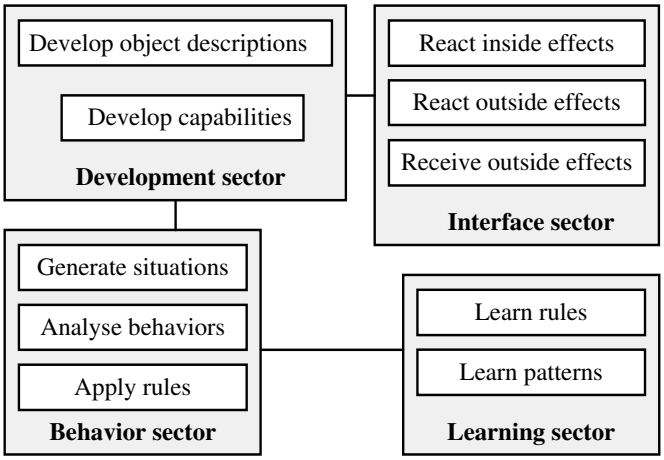


Figure 8
Sectors in virtual space

Virtual space characterized by actual state, responses for change attempts and sensitive to changes in the outside world. Development sector constructs space by using of subspaces (Figure 9). A subspace is defined for a set of closely related engineering objects as it was explained in sector 3 of this paper (Figure 1). Empty and imported subspaces are available for development of an actual space. Development effects are generated by adaptive actions. Separated development

protects the virtual space against undesired modifications. Effects are exchanged with interfaced subspaces.

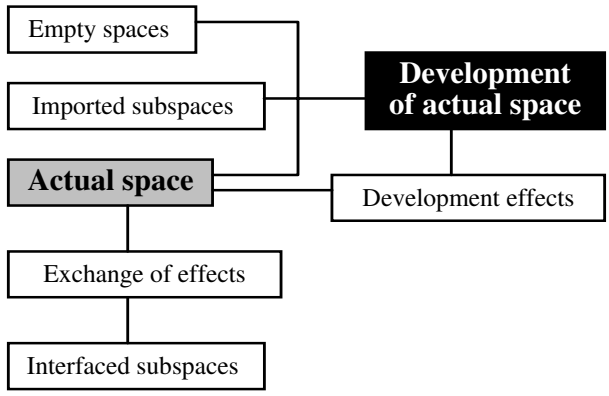


Figure 9
Development of virtual space for engineering

Conclusions and Implementation

Very complex structure of associative connections amongst high number of engineering objects in product models cause serious problems at management of frequent changes during development of the modeled product and its production. Solution for this problem is urgent because product modeling has been extended to definition, generation, and handling all information for engineering activities during entire lifecycle of products. Consistency of the heterogeneous product model requires information about the variation of modeled object parameters due to variations in other parameters. The paper introduced a possible solution in which three extensions are proposed for modeling in current industrial PLM systems. Extensions are integration of descriptions of closely related engineering objects, new model entities for description of engineering objectives, associative relations, and adaptive actions, and new methods for change management.

Method to reveal change affect zone (CAZ) for changed objects and tracking chains of associative engineering objects are essential elements of the proposed modeling. Change management can support activities in a virtual space that can accommodate computational intelligence as representation of human intent. In this case, product model is developed in a virtual space where a development sector is responsible both for development of product descriptions and modeling capabilities. In addition, sectors are defined in the virtual space for behavior, interface, and learning related activities.

An experimental virtual engineering space is under development at the Laboratory of Intelligent Engineering Systems (John von Neumann Faculty of Informatics, Budapest Tech). It integrates subsystems in order to establish a typical PLM environment. Leading PLM software is applied for digital product definition,

analysis, intelligent computing, product data management, multi-site type of group work, and Internet portal purposes.

References

[1] L. Horváth, I. J. Rudas: *Modeling and Problem Solving Methods for Engineers*, ISBN 0-12-602250-X, Elsevier, Academic Press, 2004

[2] Kim, Y. S., Wang, E.: Recognition of Machining Features for Cast then Machined Parts, *Computer-Aided Design,* Vol. 34, No. 1, (2002): pp. 71-87

[3] Zha, X. F., Du, H.: A PDES/STEP-based Model and System for Concurrent Integrated Design and Assembly Planning, *Computer-Aided Design*, Vol. 34, (2002)

[4] Mannistö, T., Peltonen, H., Martio, A. Sulonen, R.: Modeling Generic Product Structures in STEP, *Computer-Aided Design*, Vol. 30, No. 14, 1998, pp. 1111-1118

[5] Yasuhisa Hasegawa, Toshio Fukuda: Motion Coordination of Behavior-based Controller for Brachiation Robot, In *Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetic, Human Communication and Cybernetics*, IEEE, Tokyo, Vol. 6, 896-901, 1999

[6] Vokorokos, L.: *Digital Computer Principles*. Typotex Ltd. Retek 33-35, ISBN 96-39548-09-X., Budapest, p. 230, 2004

[7] M. Caccia, P. Coletta, G. Bruzzone, G. Veruggio: Execution Control of Robotic Tasks: a Petri Net-based Approach, *Control Engineering Practice*, Volume 13, Issue 8, August 2005, pp. 959-971

[8] J. A. Domínguez-López, R. I. Damper, R. M. Crowder, C. J. Harris: Adaptive Neurofuzzy Control of a Robotic Gripper with On-Line Machine Learning, *Robotics and Autonomous Systems*, Volume 48, Issues 2-3, 30 September 2004, pp. 93-110

[9] Da Ruan, Changjiu Zhou, Madan M. Gupta: Fuzzy Set Techniques for Intelligent Robotic Systems, *Fuzzy Sets and Systems,* Volume 134, Issue 1, 16 February 2003, pp. 1-4

[10] Paul F. M. J. Verschure, Philipp Althaus: A Real-World Rational Agent: Unifying Old and New AI, *Cognitive Science* Volume 27, Issue 4, July-August 2003, pp. 561-590

[11] József Tick: Software User Interface Modelling with UML Support, in *Proc. of the IEEE International Conference on Computational Cybernetics, ICCC 2005*, Mauritius, 2005, pp. 325-3

[12] Hermann Gy.: Geometric Modeling in Reconstructing Surgery, in *Proc. of the 6th International Conference on Intelligent Engineering Systems 2002 (INES 2002)*, Opatija, Croatia, 2002, pp. 379-382

[13] L. Horváth, I. J. Rudas: Virtual Technology-based Associative Integration

of Modeling of Mechanical Parts, *Journal of Advanced Computational Intelligence*, *Intelligence*, Vol. 5, No. 5, 2001, pp. 269-278

[14] L. Horváth, I. J. Rudas, G. Hancke: Feature Driven Integrated Product and Robot Assembly Modeling, in *Proc. of the The Seventh International Conference on Automation Technology, Automation 2003,* Chia-yi, Taiwan, 2003, *pp.* 906-911

[15] L. Horváth, I. J. Rudas: Modeling of the Background of Human Activities in Engineering Modeling, *Proceedings of the IECON '01, 27th Annual Conference of the IEEE Industrial Electronics Society,* Denver, Colorado, USA, 2001, pp. 273-278

[16] L. Horváth, I. J. Rudas, C. Couto: Integration of Human Intent Model Descriptions in Product Models, *In book Digital Enterprise - New Challenges Life-Cycle Approach in Management and Production*, Kluwer Academic Publishers, pp. 1-12

[17] L. Horváth, I. J. Rudas: Possibilities for Application of Associative Objects with Built-in Intelligence in Engineering Modeling, in *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Tokyo, Vol. 8, No. 5, pp. 544-551, 2004

Edge Detection Model Based on Involuntary Eye Movements of the Eye-Retina System

András Róka, Ádám Csapó, Barna Reskó, Péter Baranyi

Computer and Automation Research Institute,
Hungarian Academy of Sciences
E-mail: baranyi@sztaki.hu

Abstract: Traditional edge-detection algorithms in image processing typically convolute a filter operator and the input image, and then map overlapping input image regions to output signals. Convolution also serves as a basis in biologically inspired (Sobel, Laplace, Canny) algorithms. Recent results in cognitive retinal research have shown that ganglion cell receptive fields cover the mammalian retina in a mosaic arrangement, with insignificant amounts of overlap in the central fovea. This means that the biological relevance of traditional and widely adapted edge-detection algorithms with convolution-based overlapping operator architectures has been disproved. However, using traditional filters with non-overlapping operator architectures leads to considerable losses in contour information. This paper introduces a novel, tremor-based retina model and edge-detection algorithm that reconciles these differences between the physiology of the retina and the overlapping architectures used by today's widely adapted algorithms. The algorithm takes into consideration data convergence, as well as the dynamic properties of the retina, by incorporating a model of involuntary eye tremors and the impulse responses of ganglion cells. Based on the evaluation of the model, two hypotheses are formulated on the highly debated role of involuntary eye tremors: 1) The role of involuntary eye tremors has information theoretical implications 2) From an information processing point of view, the functional role of involuntary eye-movements extends to more than just the maintenance of action potentials. Involuntary eye-movements may be responsible for the compensation of information losses caused by a non-overlapping receptive field architecture. In support of these hypotheses, the article provides a detailed analysis of the model's biological relevance, along with numerical simulations and a hardware implementation.

Keywords: Image contour detection, Non-Overlapping receptive field, Artificial involuntary eye-movements

1 Introduction

The subjects of artificial vision and image processing are two of the most important areas in the development of robot technologies. Due to the overall failure of classical computational methods to provide general purpose applications in these areas, soft computing techniques and biologically inspired approaches to computation have become more and more popular in the last decade. The aim of these novel paradigms is to provide digital implementation of a variety of operations ranging from edge detection to more complex visual tasks, while guaranteeing the efficiency of biological visual systems [9,15]. In this paper we propose a novel edge-filtering method, in accordance with the latest findings in cognitive research. The paper is structured as follows: in a preliminary section we briefly present the goals of the edge-detection model. This will be followed by an overview of its biological background. Further sections treat the details of the model and discuss the hypotheses postulated based on its evaluation. Finally, test results and a hardware implementation are also presented.

2 Motivation

One thing in common between the majority of classical and biologically inspired edge detection methods used in image processing is that the output image is computed as the convolution of the input image and a filtering operator [9,7]. This has two important implications as regards the output image:

- Input and output images have the same spatial resolution.
- A given pixel value on the input image influences several pixel values on the output image.

According to findings in retinal research, the convolution-based computational philosophy is biologically inadequate. This is well demonstrated by the fact that the number of retinal cones (photoreceptors sensitive to color) is about 6 million and the number of rods (photoreceptors sensitive to light regardless of its color) is about 125 million, while at the same time the number of ganglion cells sending the visual information to the brain is only about 1.2 million. This suggests that image processing as well as massive image compression (also referred to as convergence) takes place in the retina [1,12]. In terms of image processing, the above facts also imply the following:

- Input and output images do not have the same resolution.
- The influence of a given pixel value in the input image on a given pixel value on the output image is described by a complex temporal and spatial relationship.

In this paper we propose an edge detection method that implements data convergence and several dynamic properties of ganglion cells. At the same time, the proposed algorithm is characterized by low complexity and is therefore easily implementable on hardware.

3 Biological Overview

3.1 Structure of the Retina

The retina is the first stage in the visual processing hierarchy. Besides being responsible for the detection of light intensities, the retina also carries out preprocessing tasks and filters visual information that is important in later stages of feature extraction.

The retina is composed of several types of cells: cones, rods, bipolar cells, horizontal cells, amacrine cells, and ganglion cells. All of these cells can be categorized into numerous subtypes based on their structure and functionality [5,11].

Information processing in the retina ends with a layer of ganglion cells. The axons of ganglion cells constitute the optic nerve which leads to the LGN and visual cortex, where higher-level visual processing takes place. Scientists distinguish between at least 11 kinds of ganglion cells based on the types of information they provide, as well as the way they react to stimuli (impulse response) [7].

Before providing a further description of the functionality of the retina, it is important to define the term receptive field. The state of a nerve cell is affected by every photoreceptor cell that provides it with input (either directly or indirectly). This set of cells providing input is referred to as the receptive field of the nerve cell. The structure of receptive fields can be observed from the firing patterns of cells when they are stimulated in an artificial environment.

The key information processing activity of the retina deals with the detection of light, a task which is carried out by rods and cones. These photoreceptor cells reflect the intensity of incoming light through their membrane potentials [17]. The information provided by photoreceptor cells is primarily processed by the complex structure of horizontal and bipolar cells. The co-operation of these two cell types creates the well-known center-surround receptive fields of ganglion cells [2,3,4]. The structure of center-surround receptive fields comes in two types: on-centered off-surround and off-centered on-surround receptive fields. On-centered off-surround cells are depolarized when there is light at the center of their receptive fields and less light at the periphery, while off-centered and on-surround

cells prefer more light at the periphery of their receptive field, and less at the center.

From an engineering point of view, the center-surround receptive field structure is sensitive to changes of light intensity, a property crucial for contour detection. On-centered off-surround cells are sensitive to the 'light edge' of a light-to-dark transition, while off-centered off-surround cells are sensitive to the 'dark edge' of the same transition [2,3,4].

3.2 Non-Overlapping Receptive Fields

For a long time, experiments have shown extensive overlaps between receptive fields on the retina. However, when comparing the number of axons of photoreceptor cells to the number of axons in the optic nerve, it was discovered that the 130 million axons of rods and cones are condensed into 1.2 million axons in the optic nerve. Because of this fact, it was assumed that the retina performs some kind of information compression [14].

With the evolution of experimental methods, it was possible to make a distinction between many kinds of ganglion cells. Devries and Baylor [6] were able to distinguish between 11 kinds of ganglion cells, based on their receptive fields and response characteristics. Different kinds of ganglion cells provide different kinds of information, such as contour information, intensity information, motion information, as well as information on uniformly lighted image segments. It was also shown that receptive fields of ganglion cells of the same type do not overlap in the central fovea; the center of these receptive fields are located at a distance of one diameter (Figure 1). These measurements were confirmed by Packer and Dacey [16].

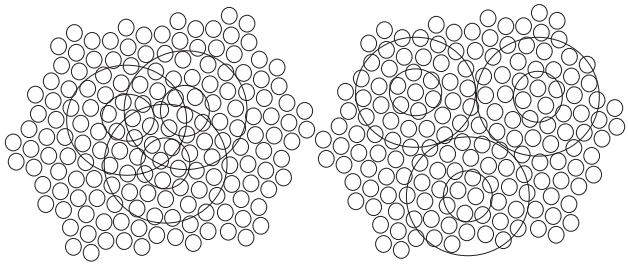


Figure 1

Contrary to the previous theory, it was shown that receptive fields of ganglion cells of the same type do not overlap in the mammalian central fovea

The scientific literature contains much contradicting data as regards the size and spatial organization of receptive fields, because the retina shows significant differences in various animal species. The model we propose takes into

consideration only the foveal areas of the mammalian retina, where the size of receptive fields is relatively small and uniform. Because of the lack of overlaps between receptive fields in the fovea, only intensity transitions that fall in the center of the receptive fields can be detected. Applying a non-overlapping architecture in today's well-known edge-detection methods leads to considerable losses in contour information. The proposed model provides a possible solution to the conflict between the continuous perception of contours and the non-overlapping receptive field architecture in the central fovea.

3.3 Involuntary Eye-Movements

The three major kinds of involuntary eye movements that occur during fixation are microsaccades, drifts and tremors (also sometimes referred to as nystagmuses) [13,10].

Of the three eye movements, later sections of this paper concentrate on tremors. Tremors are involuntary, rhythmic oscillations of the eye that have frequencies of about 90 Hz and amplitudes of roughly the diameter of a cone on the fovea (therefore the diameter of the smallest of photoreceptor cells). There is currently no conclusive evidence on the functional role of tremors, however, artificially eliminating tremors, researchers have found that vision faded away.

The model for edge-detection proposed in this paper uses non-overlapping receptive fields, but also incorporates tremors in order to achieve the effects of overlapping receptive fields through time. It will be shown that besides following the structure of human visual perception, the model accounts for the 130:1 information reduction ratio characteristic to the pathway between photoreceptor cells of the retina and ganglion cells [18].

4 The Proposed Retina-Inspired Model for Edge Detection

4.1 Filters Used for Edge Detection

The model proposed in this paper performs edge detection based on the center-surround structure of receptive fields. Despite the fact that in the retina, the size of these receptive fields increases from the fovea towards the peripheral areas, the model we propose considers only the foveal areas of the retina, where the size of receptive fields is relatively small and uniform. In the fovea, receptive fields are so small that the central area of the receptive fields consist of only one cone [16,8]. Taking into account the diameter of the photoreceptors and receptive fields

in the mammalian retina, receptive fields are represented by a 3-by-3, two-dimensional filter matrix F, which resembles the Laplace-operator (Figure 2). Each matrix value represents an input weight with which the corresponding stimulus is multiplied.

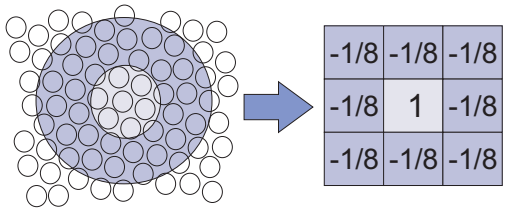


Figure 2

Receptive fields in mammalian fovea small enough to be represented by 3-by-3 pixel filter operators, as was done in many previous models

The configuration of weights depends on the concrete type of receptive field being modeled; on-centered fields have positive values in the central area, surrounded by all negative weights, while off-centered fields contain a central negative value, surrounded by all positive weights.

4.2 Non-overlapped Edge Filtering Model

If we are to accept the fact that receptive fields do not overlap in the fovea, we have to consider its implications.

The proposed model uses no overlaps between filter matrices (Figure 3). The input image is tiled using a mosaic arrangement of the filter matrix F in a non-overlapped manner. The center of each F filter is at a distance of three pixels from each of the four closest filter matrix centers. The image pixel values are multiplied by the corresponding F values and a weighted sum is calculated for each filter matrix center. The matrix composed of the filter centers provides the output matrix of the non-overlapped filtering operation. Figure 4 shows this processing structure.

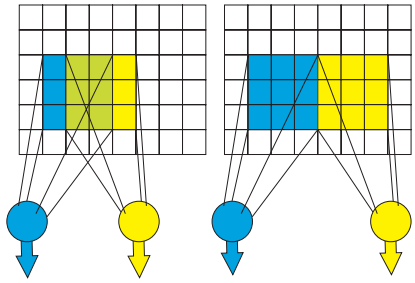


Figure 3

Overlapping and non-overlapping filtering architecture

In the model with no overlaps, it can easily occur that a change in contrast falls precisely in between two filter matrices, thus remaining undetected. Figure 5 shows an example of such information loss. Note that the output of this processing structure is identical to the 3-by-3 subsampling of the convolution of the input image with the same filter matrix F, and therefore the spatial resolution of the output is $1/9^{th}$ the spatial resolution of the output obtained using convolution. In order to make the two results comparable, the size of the input image used in convolution-based filtering is also reduced by a factor of $1/3$ along each dimension.

Figure 4
Non-overlapping processing structure. The input image is tiled using a mosaic arrangement of the filter matrix F in a non-overlapped manner.

The non-overlapped image filtering method can be formalized as follows. Let I denote the input image, F the filter mask, J and K the output images obtained using the classical convolution-based method and the non-overlapping filtering method, respectively. In classical convolution-based filtering, the output can be expressed as the convolution of I and F over a discrete 2D space:

$$J(x,y) = F \otimes I = \sum_{j=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} F(j,k) \cdot I(x-j, y-k) \tag{1}$$

The result of the non-overlapped filtering K can be obtained from the result of the overlapped filtering J, as follows:

$$K_{i,j} = J_{m \cdot i, n \cdot j} \tag{2}$$

where n and m indicate the size of matrix F (in the current implementation, n = m = 3). It can be seen from equation 2 that the number of elements in K is m x n times smaller than the number of elements in J. The model presented above takes into account the convergence between photoreceptor cells and ganglion cells, but does not incorporate temporal properties of ganglion cells, and involuntary eye-movements are also disregarded. Receptive fields were modeled using 3-by-3 matrices. It is clear that the lack of overlaps result in a deteriorated image; certain sections of line segments are blurred, or are missing altogether.

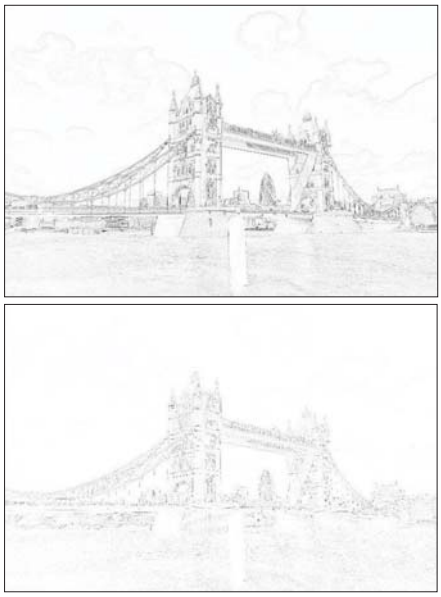


Figure 5
In the model with no overlaps in case of a change in contrast falls in between two filter matrices, thus remaining undetected, information loss occurs

4.3 Virtual Receptive Fields

The non-overlapping retina model used to generate the above images performs too poorly to be considered biologically valid. The literature does not explain how the quality of images can be guaranteed by the retina, even with the difficulties caused by this non-overlapping architecture. This section provides a possible solution to the problem.

The starting point of the model is that high-quality edge detection can be achieved only when assuming that receptive fields overlap. Taking into consideration the fact that ganglion cells of the same type do not physically overlap on the fovea, but also considering that such overlaps would otherwise be required, it is plausible to assume that the necessary overlaps are achieved through time. The large-frequency and small-amplitude involuntary eye-movements known as tremors (see section 4.5) could be capable of causing the necessary overlaps through time. Such small tremors with frequencies of 90 Hz and amplitudes of 2-3 cones are sufficient for the correction of the minor gaps between non-overlapping receptive fields, and could reveal the presence of the previously unperceived line segments. Due to tremors, the image that is projected onto the retina suffers random displacements with an amplitude of 2-3 cones. This extension of non-overlapping receptive fields covers larger parts of the scene than the receptive fields did in the

previous model, and virtual receptive fields are created through time. Such virtual fields are larger in size compared to the original receptive fields, and also show considerable overlaps. For instance, assuming a physical receptive field with a diameter of 6 cones, and tremors with an amplitude of 3 cones, the emergent virtual receptive fields will have a diameter of 12 cones and the produced overlap will be enough to perform edge filtering without any information loss.

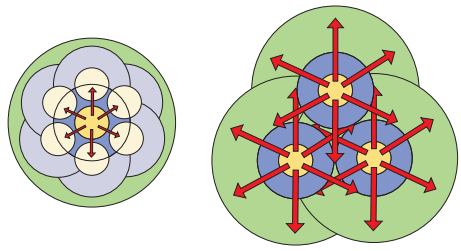


Figure 6

Due to the involuntary tremors and the impulse response of ganglion cells, virtual receptive fields are created. Virtual fields greater than real ones, and show notable overlap.

The formation of such virtual receptive fields would be impossible without taking into account the impulse responses of ganglion cells.

4.4 Temporal Model of Ganglion Cells

It is generally a common property of ganglion cells that they produce increased activity when stimulated, and that their activity decreases only gradually when the stimulus either disappears or remains stagnant. However, if stimulating effects increase, the output of ganglion cells increases even more, and finally declines to 0 (Figure 7).

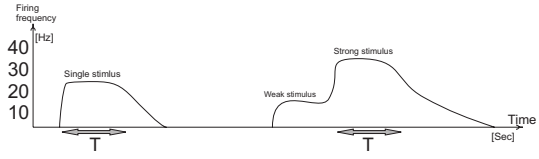


Figure 7

The general impulse response of a ganglion cell.

In the proposed model, the stochastic response properties of ganglion cells are approximated using a window function (Figure 8). This simplification is justified because it conserves the essence of the ganglion cell's functionality (in terms of its 'temporal memory' or 'time constant'), and it is also convenient because it ensures that unnecessarily complex computations are not brought into the model.

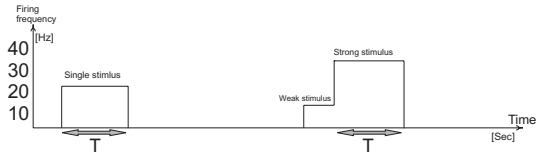


Figure 8
Modeled response of ganglion cells in time.

The proposed model calculates the sum of the inputs of ganglion cells, and holds their output for T time frames. If, during this time, their inputs are stimulated even more, then their output will increase. Otherwise, the output level will return to 0 after T time frames.

4.5 Artificial Tremors

As described above, tremors could be capable of causing the necessary overlaps between receptive fields through time. This is the reason why artificial tremors were introduced to the proposed model. Artificial tremors can be simulated by software, and can also be implemented by adding mechanical vibrations to the image sensor equipment (camera). In order to create an artificial tremor that is as similar as possible to its biological counterpart, the physical properties of tremors have to be considered.

Eye tremors have frequencies of about 90Hz, and amplitudes of about the diameter one cone. Based on this measurement data, the parameters of the artificial tremors incorporated in the non-overlapped filtering model were chosen such that:

- Amplitude: A = 2 pixels
- Frequency: f = 90 frame^{-1}
- Direction: Random

The simulation of tremors is done by shifting the input image I in a random direction by a maximum of 2 pixels. Mechanical implementations of artificial tremors can be realized by computing the summation of the effect of two perpendicular mechanical vibrations of different frequencies.

4.6 Proposed Tremor-based Dynamic Retina Model

As described in section 4.2, the input image is tiled using a mosaic arrangement of the filter matrix F in a non-overlapped manner. This means that the sets of pixels seen by each receptive field are disjoint, and that their union covers the whole image. Each ganglion cell computes the sum of its inputs to produces its output.

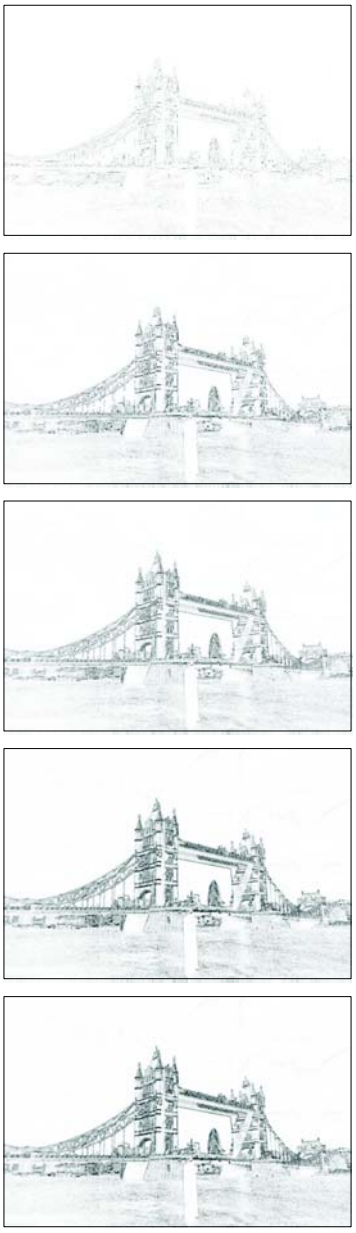


Figure 9
The obtained image using T = 1, 5, 10, 30, 100, from top to bottom

The difference between the non-overlapping edge filtering method (Section 4.2) and the proposed model lies in the use of artifical tremors as well as the temporal model of ganglion cells in the latter model. In the non-overlapping method, the output is a static function of the input, while in the tremor-based model, ganglion cells produce their output according to the window function described in section 4.4. Virtual receptive fields can therefore be developed thorough time, based on the artificial tremors. The notion of virtual receptive fields does not appear directly in the mathematical formulation of the model, but is useful in its interpretation.

At any given moment, the output of each modeled ganglion cell is determined as the maximum of the sum of its inputs within the last T time frames. From a biological point of view, the value of T is between 15 and 30 Ttr, where Ttr denotes the period time of artificial tremors. Based on simulation results, it is clear that the growth in image quality slows down as T increases. Figure 9 demonstrates the relationship between image quality and the value of T. In mathematical terms, the proposed model can be formulated as follows: Let $Is1$ be the tremor-based shifted image, and F be the filter matrix. The shifted image, $Is1$, is then filtered in a non-overlapped manner using filter F, yielding $Ks1$. This process is done as many times as defined by the T

time constant parameter, resulting in images Ksi, where $i = 1..f$ and $f = T / T_{tr}$. The f images are then integrated into one image K. The integration is computed using a pixel-wise maximization:

$$\forall x, y : K(x, y) = \max_i K_{Si}(x, y) \tag{3}$$

The resulting image K is one output frame of the non-overlapped filtering system. Using this approach, each output pixel aggregates edge information from its neighborhood as defined by the properties of the artificial tremor. Unlike the static (tremorless) non-overlapped case, contour information contained in the input image will not be lost, but aggregated into the output pixel. The authors were not able to find any existing theory that described such a close link between tremors and the non-overlapping receptive field architecture on the retina. According to the theory proposed here, the retina does not perform data compression (as suggested by others in the past) in order to achieve the convergence between photoreceptor cells and ganglion cells, but uses the mechanism described above. Such a mechanism allows only the most important features to be considered in each disjoint subset of the visual field.

4.7 Advantageous Properties of the Proposed Model

In the proposed model, simple implementation is coupled with fine output image quality. Through its data convergence, output images are 9 times smaller than input images (in the case of using 3-by-3 pixel filter matrices – Figure 10. For instance, a 800-by-600 input image would result in an output image with the same resolution using classical edge-detection methods 480 Kbytes). However, the tremor-based method would generate an output of only 266-by-200 pixels in the same case (53 Kbytes), and the output image still contains the necessary contour information for further image processing tasks.

In hardware implementation, from the point of view of speed rate and edge board complexity, it is important to use built-in memory if possible. The convergence of the proposed model provides an appreciable reduction of almost one order of magnitude. The output image of 53 Kbytes can easily be stored in either a PIC microcontroller, while 480 Kbytes of data generally require external memory.

5 Hardware Implementation

The tremor-based non-overlapping retina model was implemented on a Virtex II. FPGA (Figure 11). An important goal was to make the implementation independent of the type of camera used.

Figure 10
Some advantages of the proposed model are its simple implementation, fine output image quality, and its convergence of almost one order of magnitude

For this reason, the FPGA implementation receives its input from a desktop monitor, through a VGA interface. In this way, the implementation can be used both to evaluate real-time performance through a camera connected to the computer, as well as to obtain results from pre-existing video files.

Figure 11
An image of the camera, equipped with two, asymmetrically weighted vibrating motors

The camera used for the recording of input has two vibrating motors attached to it, in order to simulate tremors. In industrial applications, artifical tremors would be implemented in the reading-in method of the FPGA, but in this case mechanical vibrations were applied for the purposes of illustration. The two motors induce vibrations in a horizontal and vertical direction. Because the tremors need to be as delicate as possible, the authors decided to use a PWM remote controller to

control the motors. In this way, direct contact with the camera can be avoided, and the frequency and amplitude of tremors can be adjusted without interfering with the system's functionality.

The implementation uses the 3-by-3 F matrices presented above. This means that information can only be processed when three pixel rows are received. The first two rows modulo 3 are stored in a 2 Kbyte dual-port memory, and processing is pipelined with the reading of the third row. Although the responses of ganglion cells are calculated serially, one after the other, the final output is calculated using a pipelined arithmetic unit. The correctness of the input, as well as the commencement of calculations, is guaranteed by a control unit. The calculated output values are also stored in an internal, 53 Kbyte dual-port memory. A separate unit is used to calculate the dynamical functions of ganglion-cells based on these values. As a trade-off between biological relevance and low-complexity implementation, a moving-maximum procedure is used for these purposes. Because of the non-overlapping architecture, output images are smaller than input images. For this reason, output images are rendered to the center of the display, with a black background.

Results obtained using the hardware implementation are very similar to the software-based results seen in figure 9. The difference between the two is that the current FPGA implementation provides lower resolution, but real-time results.

6 Hypotheses

Based on recent findings in biology and the experimental results presented in this paper, the following hypotheses are postulated:

Hypothesis 1: The role of involuntary eye tremors has information theoretical implications. This hypothesis is supported by recent findings in biology, which claim that receptive fields cover the surface of the fovea in a mosaic arrangement with minimal overlaps, and also by the fact that the output of retinal edge filtering on the axons of the ganglion cell layer has a much lower resolution than the input resolution on the photoreceptor cells.

Hypothesis 2: From an information processing point of view, the functional role of involuntary eye-movements extends to more than just the maintenance of action potentials. Involuntary eye-movements may be responsible for the compensation of informationlosses caused by a non-overlapping receptive field architecture. This hypothesis is supported by the experimental results presented in section 4.6, which show the difference between the edge-detected images obtained by using non-overlapped filtering with and without artificial tremors. The experiments show that the application of tremors considerably enhances image quality. This can be explained by the integrative and compressive effects of tremors within a

certain locality, in contrast with the compression principle of the tremorless, non-overlapped filtering method, which completely disregards certain aspects of locality.

Conclusions

A novel, biologically inspired image filtering method was proposed. The method implements the non-overlapping mosaic arrangement of retinal receptive fields, as well as a certain kind of non-voluntary eye-motions, called tremors. The characteristics of the proposed method are different from those of convolution-based image filtering methods. In a way similar to the retina, the non-overlapped filtering implements image information compression, as a direct result of the use of non-overlapping receptive fields and artificial tremors. This has advantageous effects on the temporal and spatial requirements (execution time and memory) of a hardware implementation that needs to satisfy real-time constraints.

Based on the model, two hypotheses were formulated about the role of involuntary tremors. The first hypothesis states that convolution-based edge-detection algorithms are inadequate if our goal is to provide a biologically valid model. It is interesting to note that for the human observer, the edge-detected image using the proposed method looks more expressive than convolution-based results, which might indicate that the human vision system prefers the biologically more relevant way of producing contour images. Experimental results show that small vibrations dramatically improve the quality of edge detected images. This is generalized in the second hypothesis, which claims that a similar phenomenon may exist in the eye-retina system, which would explain the heretofore unknown role of eye tremors. The two hypotheses proposed in this paper need further support, from both biology and computational experiments.

Acknowledgements

This research was supported by the János Bolyai Postdoctoral Scholarship.

References

[1] H. Barlow, *"Summation and inhibition of the frog's retina"*, Journal of Physiology, 119:69–88, 1953

[2] J. Bilotta and I. Ambrov, *"Spatial properties of goldfish ganglion cells"*, J. Gen. Physiol., 93(6):1147–1169, 1989 June

[3] S. Brown, S. He, and R. Masland, *"Receptive Field microstructure and dendritic geometry of retinal ganglion cells"*, Neuron, 27(2):371–383, 2000 Aug.

[4] D. Burkhardt and P. Fahey, *"Contrast enhancement and distributed encoding by bipolar cells in the retina"*, J. Neurphysiol., 80(3):1070–1081, 1998 Sept.

[5] D. Dacey, *"Primate retina: cell types, circuits and color opponency"*, Prog Retin Eye Res., 18(6):737–763, 1999 Nov.

[6] S. Devries and D. Baylor, *"Mosaic arrangement of ganglion cell receptive fields in rabbit retina"*, J. Neurphysiol., 78(4):2048–2060, 1997 Oct.

[7] S. H. DeVries and D. A. Baylor, *"Mosaic Arrangement of Ganglion Cell Receptive Fields in Rabbit Retina"*, The Journal of Neurophysiology, 78(4), 1997

[8] L. Diller, O. Packer, J. Verweij, M. McMahon, D. Williams, and D. Dacey, *"L and M cone contributions to the midget and parasol ganglion cell receptive fields of macaque monkey retina"*, J. Neurosci., 24(5):1079–1088, 2004 Febr.

[9] C. Grigorescu, N. Petkov, and M.Westenberg, *"Contour and boundary detection improved by surround suppression of texture edges"*, Image and Vision Computing, 22(8):609–622, 2004

[10] Hennig and Worgotter, *"The effect of fixational eye movements on hyperacuity"*. Computational Neuroscience, Dept. of Psychology, University of Stirling

[11] D. Hubel, *"Eye, Brain and Vision"*, W H Freeman, 1988

[12] S. Kuffler, *"Discharge patterns and functional organization of mammalian retina"*, Journal of Neurophysiology, 16:37–68, 1953

[13] S. Martinez-Conde, S. L. Macknik, and D. H. Hubel, *"The Role of Fixational Eye Movements in Visual Perception"*, Nature Reviews Neuroscience, 5(3):229–240, 2004

[14] M. Meister, *"Multineural codes in retinal signaling"*, Proc Natl Acad Sci USA, 93(2):609–614, 1996 Jan.

[15] V. Navalpakkam and L. Itti, *"An integrated model of top-down and bottom-up attention for optimal object detection"*, In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2049–2056, 2006

[16] O. Packer and D. Dacey, *"Receptive field structure of H1 horizontal cells in macaque monkey retina"*, Journal of Vision, 2(4):279–292, 2002

[17] D. Schneeweis and J. Schnapf, *"The photovoltage of macaque cone photoreceptors: adaptation, noise, and kinetics"*, J. Neurosci., 9(4):1203–1216, 1999 Febr.

[18] R. Sekuler and R. Blake, *"Perception"*, Mc Graw Hill, 1994

[19] J. Thiem, C. Wolff, and G. Hartmann, *"Biology-Inspired Early Vision System for a Spike Processing Neurocomputer"*, In Biologically Motivated Computer Vision, pp. 387–396, 2000

Geometric Error Correction in Coordinate Measurement

Gyula Hermann

BrainWare Ltd.
Völgy utca 13/A, H-1021 Budapest
Hungary

Abstract: Software compensation of geometric errors in coordinate measuring is hot subject because it results the decrease of manufacturing costs. The paper gives a summary of the results and achievements of earlier works on the subject. In order to improve these results a method is adapted to capture simultaneously the new coordinate frames in order use exact transformation values at discrete points of the measuring volume. The interpolation techniques used have the draw back that they could not maintain the orthogonality of the rotational part of the transformation matrices. The paper gives a technique based quaternions which avoid this problem and leads to better results.

1 Introduction

Three dimensional coordinate metrology is a firmly established technique in industry. Their universal applicability and high degree of automation accounts for it's success in the last 30 years. In order to full-fill its task to verify the geometry of products on the basis of the measured results CMM-s must be in principally be an order of magnitude more accurate than the machine tool used to manufacture the part.

Over the last 50 years one can observe enormous enhancement in positioning and measuring accuracy. The main portion of this enhancement is the result of improved knowledge about high precision machine design [18]. A fundamental principle was recognized by professor Abbe already in the 1890's about the alignment of the displacement measuring system with the distance to be measured. Another fundamental principle is the separation of the structural and measuring functions in a machine. Already in the 1880's measuring equipment was built in witch the measuring system was attached to a separate metrology frame. The third important factor, to be concerned, is the thermal distortion of the metrology system. A short overview of novel constructions for high precision coordinate measuring machines is given in [8].

As mechanical accuracy is costly, whereas repeatability is not expensive, software techniques were used from the beginning to compensate for the systematic errors in order to keep manufacturing costs low.

One of the earliest papers on error compensation of coordinate measuring machines is by Zhang et al. [25]. They describe the compensation of a bridge type industrial three-coordinate measuring machine, which resulted in an accuracy improvement by approximately a factor 10. The machine consists of only translational axis and the infinitesimal rotation errors are described by the rotation matrix where the trigonometric functions are replaced by the first term in their Taylor series. The correction vectors are determined at equally spaced points in the measuring volume and are stored in the memory of the computer in the form of look-up table. The correction vectors at intermediate point are calculated simply by linear interpolation.

An analytical quadratic model for the geometric error of a machine tool was developed by Ferreire and Liu [6] using rigid body kinematics. They introduced the notion of shape and joint transform. The former describes the transformation between the coordinate system on the same link and latter the transformation across a joint. To represent the transformations they introduced the use of homogeneous transformation in matrix form. A quadratic expression was developed for the case where the individual joint errors vary linearly with the movement along the joint or axis. The global error description was obtained by concatenating these matrices.

Duffie and Yang [2] invented a method to generate the kinematic error functions from volumetric error measurements. To represent the displacement error a vectorial approach was followed. The rotational errors were described by matrices in which, taking into account that the angular error are small, the cosine and the sine terms can be approximated with the lowest order terms of their Taylor series. The model was used to describe the error of a measuring probe neglecting the rotation. The translational error components were approximated by cubic polynomials. To find the coefficients least square fit was applied.

Teeuwsen [20] described the error motion of the kinematic components of a coordinate measuring machine by using homogeneous transformations and concatenating these transformations to calculate the resulting global error. Assuming that the rotational errors are very small, he neglected the second order term. Hereby he could ensure the commutativity of the matrices, but at the same time the orthonormality of these matrices was lost, which means that they do not represent a pure rotation anymore. The error motions of the probe displacements were also handled by transformation matrices. In order to establish the error map in the form of correction vectors the various error components were measured on a semi automatic way at discrete points of the measuring volume. To obtain a continuous description of the correction vector, between these points, regression was used to establish a piecewise polynomial representation.

Ruijl [15] has built a high precision coordinate measuring machine with a measuring uncertainty of 50 nm in a 100x100x40 mm measuring volume. The machine has a novel construction, where the air bearing table performs the measuring motion in all the three principal directions. It was derived that if the measuring systems are aligned with the centre of the probe tip the relationship between the position of the measuring system and the contact point on the workpiece is unique. This means that the functional point is the centre of the probe tip and hence it is possible to comply with the Abbe principle.

The nano measuring machine of SIOS [17] is based on the same principles. This machine is currently applied as the stage for a long range scanning microscope.

Kim et al. [12] have constructed an unusual machine. One attempt to get rid of the parallax error of orthogonal type coordinate measuring machines is the application of the so-called multilateration. It is to measure the diagonal distances of the probe using tracking laser interferometers with retro-reflectors. The paper describes a scheme of multilateration based on a single volumetric interferometer system. The volumetric interferometer generates two spherical wavefronts from the probe by using diffraction point sources. The emanated wavefronts interfere within the measuring volume, while two dimensional array of photodetectors mounted on the machine frame capture the interferometric intensity field. Phase information is used, from which the coordinates of the probe are determined. A second interferometer is installed to measure the x and y position of the machine table.

Kim and Chung [11] also applied infinitesimal matrix transformation to correct the position error due to geometric imperfections and transient thermal errors of a machine tool to improve on machine measurement accuracy. Thermal errors were derived from the thermal drift of the spindle in the three principle directions.

The static and transient thermal errors and their compensation are discussed by Kruth et al. [13]. Capturing temperature distribution of the machine structure the thermal deformations can be calculated using the linear thermal expansion coefficients of the individual machine components. However the determination of sensor positions in a cumbersome ‘trial and error’ task.

Recently in a paper Tan and his coauthors [19] describe the application of neural networks for the error compensation of single-axis, a gentry and X-Y stage. The advantage of using neural networks is in the followings: they could be used to approximate any continuous mapping. This mapping can be achieved by learning. Parallel processing and nonlinear interpolation can also be performed. Using this technique the authors could improve the positioning accuracy depending on the configuration investigated by a factor between two and three.

2 Overview of the Errors and their Sources

When considering the mechanical accuracy of coordinate measuring devices three primary sources of quasi-static errors can be identified:

- Geometric errors due to the limited accuracy of the individual machine components such as guideways and measuring systems.
- Errors related to the final stiffness of those components, mainly by moving parts.
- Thermal errors as expansion and bending of guideways due to uniform temperature changes and temperature gradients.

Geometric errors are caused by out of straightness of the guideways, imperfect alignment of the axis and flatness errors.

Deformations in the metrology frame introduce measuring errors. During measurement the deformation of the metrology frame is caused by the probing force. Its effect can be predicted with relatively high precision if the probing force is known and therefore it easily incorporated into the model.

The static deformation of the table is caused be gravity forces. It manifests itself as a contribution to out of flatness error. That means it can be handled on a similar way.

The largest deformations of the metrology frame are thermally induced. The main sources of the thermal disturbance are:

- heating and cooling source in the environment, like lighting, air conditioning, people around the machine etc.,
- heat generated by the machine itself,
- thermal memory: heat stored in the machine components from a previous thermal state.

The compensation of thermally induced errors is rather cumbersome, because of the complexity of the problem [13]. Based on results from the literature a linear thermal compensation model can be used.

3 Geometric Error Model

A coordinate measuring machine is a multiaxis machine consisting of a chain of translational and/or rotational kinematic components. The geometric deviation of a CMM is originating from the geometric deviations of its components. In order to discuss a general model the error model of the components are discussed.

The measuring loop of a coordinate measuring machine in general is given in Figure 1. Depending on the configuration of the machine all measuring motions are performed by the probe or by the table, or divided between the two.

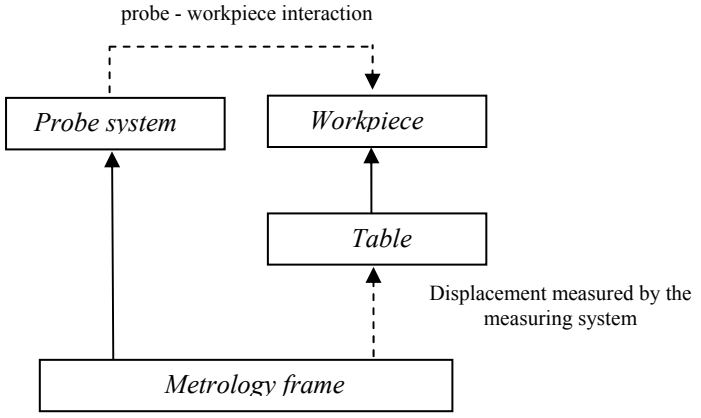


Figure 1
Schema of the metrology loop

A linear stage of precision machinery is expected to travel along a straight line and stop at a predefined position. However in the practiced the actual path deviates from the straight line due to the geometric errors of the guideways and it results also in angular errors as it is given in Fig. 2.

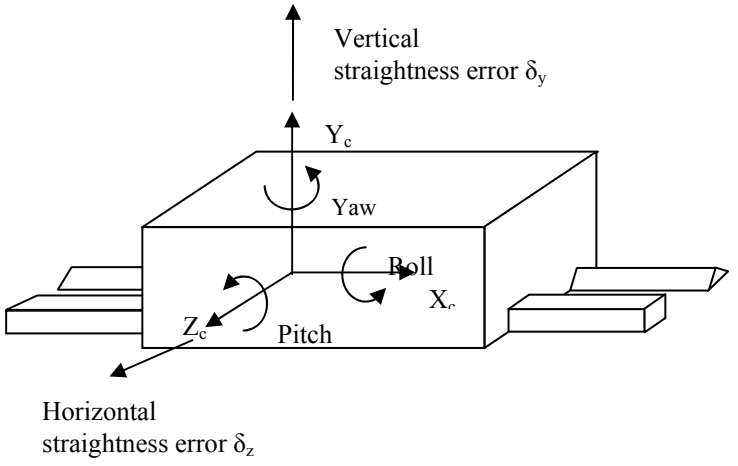


Figure 2
Representation of the six deviations of a translational kinematic component

For each axis a transformation matrix can be used to describe in homogeneous coordinates the deviations from the ideal motion. The general form of a transformation is given by:

$$T_{err} = \begin{bmatrix} c(\theta_y)c(\theta_z) & -c(\theta_y)s(\theta_z) & s(\theta_y) & \delta_x \\ c(\theta_x)s(\theta_z)+s(\theta_x)s(\theta_y)s(\theta_z) & c(\theta_x)c(\theta_z)-s(\theta_x)s(\theta_y)s(\theta_z) & -s(\theta_x)c(\theta_y) & \delta_y \\ s(\theta_x)s(\theta_z)-c(\theta_x)s(\theta_y)c(\theta_z) & s(\theta_x)c(\theta_z)-c(\theta_x)s(\theta_y)s(\phi_z) & c(\theta_x)c(\theta_y) & \delta_x \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where δx, δy and δz are the translational and θ_x, θ_y and θ_z are the rotational components and s respectively c are short for sin and cos.

In case of the coordinate table the angular errors are very small, and all the errors are position dependent the following approximation can be made:

$$T_{err} = \begin{bmatrix} 1 & -\theta_z(x) & \theta_z(x) & \delta_x(x) \\ \theta_z(x) & 1 & -\theta_x(x) & \delta_y(x) \\ -\theta_y(x) & \theta_x(x) & 1 & \delta_z(x) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Analog results can be derived for the y and z axis. Rotational components can presented on the same way and results in rather similar matrix.

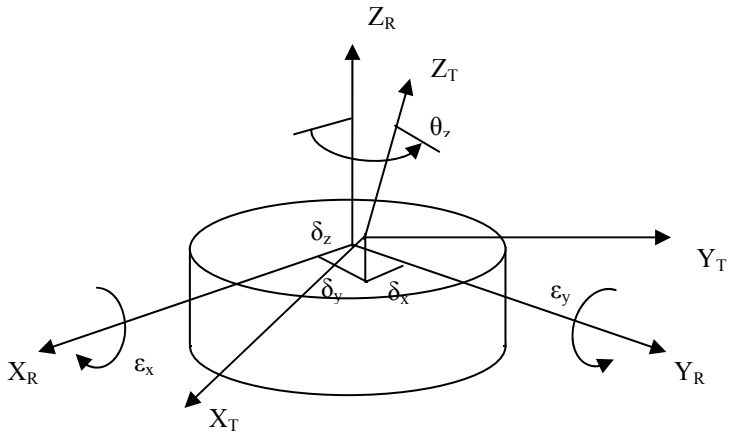


Figure 3
Representation of the six deviation of a rotational kinematic component

These components are called guided element. The guided moving elements are linked by socalled connecting elements, which can be represented by matrices with similar structure with only constant elements.

The resulting error matrix can be obtained by multiplying the individual matrices in the sequence as they follow each other in the kinematic chain.

A traditional co-ordinate measuring machine consists of three translational components x, y and z, and a probe is attached to the end of the z component. Usually the probe can be considered as a constant translational transformation.

In case of a measuring probe its error components can be handled on the similar way as it was done in case of a carriage and a rotational element.

4 Errors of a Coordinate Measuring Machine with three translational components

In order to illustrate the application of the technique described in the previous paragraph, let us consider a coordinate measuring machine with three translational components, given in Fig. 4.

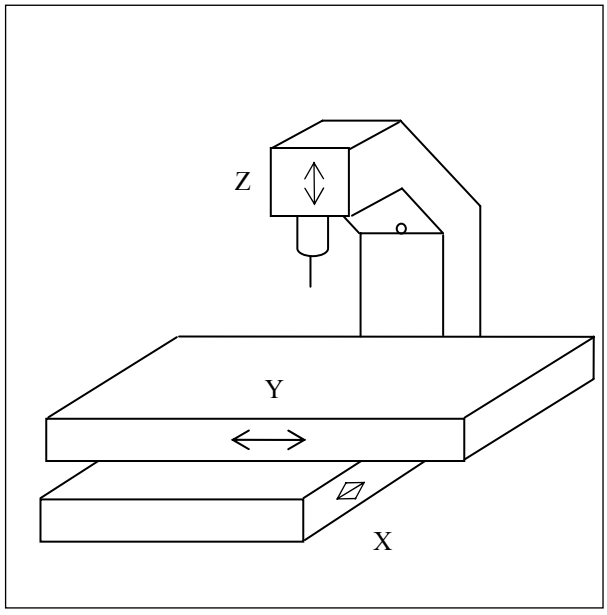


Figure 4
The investigated coordinate measuring machine

The carriage consists of two stacked tables; each of them turn rests on vacuum preloaded air bearings. The position are determined by incremental two optoelectronic measuring system, having 0,05 μm resolution. This results in a lightweight construction, which in turn ensures fast and accurate positioning.

The probe is attached to a pinole running in an air bushing. A counter weight minimizes the force needed for lifting the probe. The displacement is measured with a linear optoelectronic scale having a resolution of 0,05 μm.

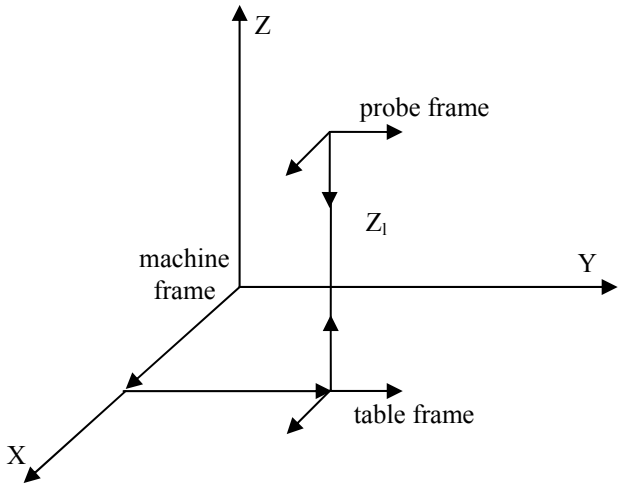


Figure 5
The frames of the CMM

The position dependent transformation matrices for the table (T_{table}) and the probe (T_{probe}) coordinate frames are given below:

$$T_{table}=\begin{bmatrix} c(\theta_y)c(\theta_z) & -c(\theta_y)s(\theta_z) & s(\theta_y) & \delta_x \\ c(\theta_x)s(\theta_z)+s(\theta_x)s(\theta_y)s(\theta_z) & c(\theta_x)c(\theta_z)-s(\theta_x)s(\theta_y)s(\theta_z) & -s(\theta_x)c(\theta_y) & \delta_y \\ s(\theta_x)s(\theta_z)-c(\theta_x)s(\theta_y)c(\theta_z) & s(\theta_x)c(\theta_z)-c(\theta_x)s(\theta_y)s(\phi_z) & c(\theta_x)c(\theta_y) & \delta_x \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & X_c \\ 0 & 1 & 0 & Y_c \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{probe}=\begin{bmatrix} c(\varphi_y)c(\varphi_z) & -c(\varphi_y)s(\varphi_z) & s(\varphi_y) & \varepsilon_x \\ c(\varphi_x)s(\varphi_z)+s(\varphi_x)s(\varphi_y)s(\varphi_z) & c(\varphi_x)c(\varphi_z)-s(\varphi_x)s(\varphi_y)s(\varphi_z) & -s(\varphi_x)c(\varphi_y) & \varepsilon_y \\ s(\varphi_x)s(\varphi_z)-c(\varphi_x)s(\varphi_y)c(\varphi_z) & s(\varphi_x)c(\varphi_z)-c(\varphi_x)s(\varphi_y)s(\varphi_z) & c(\varphi_x)c(\varphi_y) & \varepsilon_x \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & X_c \\ 0 & -1 & 0 & Y_c \\ 0 & 0 & -1 & Z_{ref}-Z_l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Here again δx, δy and δz and ε_x, ε_y, ε_z stand for the translational and θ_x, θ_y and θ_z and φ_x, φ_y, φ_z for the rotational errors of respectively the table and the probe. The constant values X_c, Y_c and Z_c are the offset coordinate distances between the machine coordinate system and the coordinate system attached to the workpiece to be measured. Z_{ref} is the probe reference point and Z_l is the probe length.

The coordinate values captured by the CMM are:

$$T_{meas} = \begin{bmatrix} x_{meas} \\ y_{meas} \\ z_{meas} \\ 1 \end{bmatrix} = T_{table} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} + T_{probe} \begin{bmatrix} x \\ y \\ z \\ y \end{bmatrix}$$

If for accuracy reason one may not substitute α for sinα and 1 for cosα than the components of each matrix describing the transformation should be captured in simultaneously. In the subsequent paragraph a suitable method is outlined.

5 Determination of the Geometric Errors by Measurement

For the calibration of coordinate measuring machines Zhang et al. [23] proposed to determine the angular errors by measuring the displacement errors along two parallel lines to the axis of motion but separated by a distance in the appropriate orthogonal direction.

In a more recent paper Zhang and Fu [24] describe the calibration of optical CMM-s using an uncalibrated reversible grid plate in three positions. In the initial position the plate is aligned with the machine coordinate system Next it is reversed about Y axis of the machine. In the third position the grid is rotated 90º about the Z axis. To determine the scale error one of the machine axis should be calibrated by a laser interferometer.

A simple measuring technique was invented by Fan et al. [3] to determine the motion accuracy of a linear stage. The idea is based on the fact that the position and orientation of a rigid body can be determined by appropriately selected six points. They measure the displacement of these points and calculate from them the rotational and translational error components. The displacement in the motion direction and the angular errors (pitch and yaw) perpendicular to this direction are measured by three laser interferometers. The roll and the straightness errors are captured by an optical setup containing two quadrant photo detectors. Taking again into considerations that the angular errors are small their values are replaced by their tangent. The invention initiated the development a dual and triple beam interferometers.

The above mentioned authors published a paper [5] about the measurement to determine the accuracy of a high precision wafer stage. Therefore 6-DOF errors of its positioning accuracy are significant. An improved version of the above described system was used. The moving part of it is an L-shaped mirror and on top of one leg a long right angle mirror. The stationary part consists of four laser

heads, two beam splitters and two quadrant photo detectors. The laser heads use four laser Doppler scales, three of which are parallel to each other. The upper two laser beams can be reflected by the long right-angle mirror and the lower one by the Y leg of the L shaped plane mirror. The fourth laser beam is aligned in the X-axis and reflected by the X mirror. Comparing the four linear measurements by four laser doppler scales, the X and Y positioning error of the moving table and its pitch and yaw errors can be determined. The upper two reflecting beams are split and each split beam is received by a quadrant photo detector. Comparing these signals the vertical straightness and the roll error can be derived at the same time. In order to minimize the cosine errors among the three displacement measurement and to ensure angular accuracy of the pitch and yaw measurements the parallelism of these beam should be precisely adjusted. The system takes into consideration the squareness alignments and the flatness error of the plane mirrors.

In their paper Gao et al. [7] describe the measurement straightness and rotational error motions of a commercially available linear airbearing stage actuated by a linear motor. The pitch and yaw errors were measured by an autocollimator. For the roll error measurement two capacitive displacement probes scan the flat surface in the XZ plane. The probes with their sensing axis in the Y direction were aligned with a certain spacing. The roll error is obtained by dividing the difference of the outputs of the two probes by the spacing between them. The horizontal and vertical straightness errors were measured by using the straightness kit of a laser interferometer.

The setup to detect motion errors of the linear stage uses two laser interferometers [16] and three capacitive sensors [14] is given in Fig. 4. The stationary part consists of a single and a dualbeam laser interferometer and three capacitive sensors perpendicular to each other. Both translational and rotational errors can be derived out of the displacement values captured by the transducers.

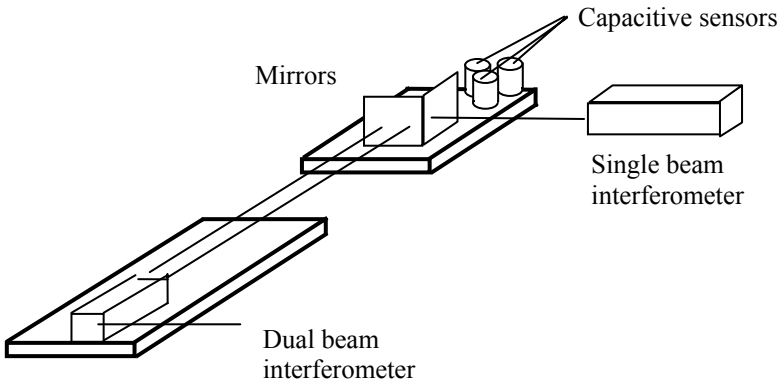


Figure 6
Set up for determining the motion error of a linear stage

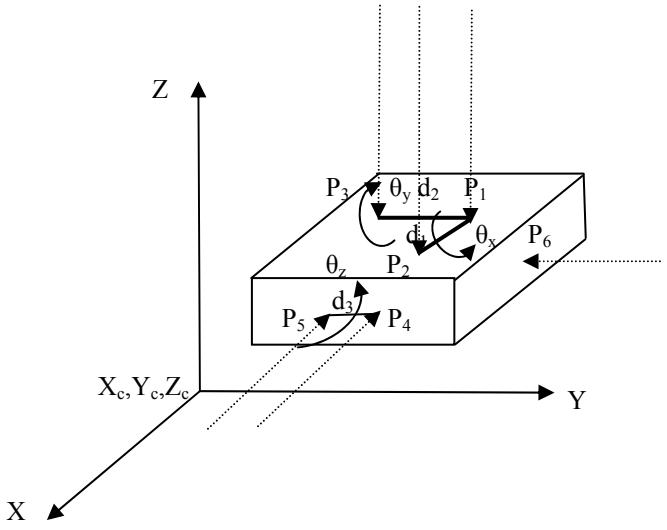


Figure 7
The measuring points on the surface of the artifact and their relations to each other

Where d_1, d_2 and d_3 represent the distance between the laserbeams, respectively the capacitive sensors, parallel to the coordinate axis.

By expanding the following determinants the equations of the table's boundary planes can be determined. These equations can be used to compute the origo of the new coordinate system and its principal axis. Having these values one can directly draw up the transformation matrix.

$$\begin{vmatrix} x-x_1 & y-y_1 & z-z_1 \\ d_1 & 0 & d_1 tg\,\theta_y \\ 0 & d_2 & d_2 tg\,\theta_x \end{vmatrix} = 0 \qquad \begin{vmatrix} x-x_4 & y-y_4 & z-z_4 \\ d_3 tg\,\theta_z & d_3 & 0 \\ tg\,\theta_y & tg\,\theta_x & -1 \end{vmatrix} = 0$$

$$\begin{vmatrix} x-x_6 & y-y_6 & z-z_6 \\ tg\,\theta_y & tg\,\theta_x & -1 \\ -1 & tg\,\theta_z & tg\,\theta_z tg\,\theta_x - tg\,\theta_y \end{vmatrix} = 0$$

6 Error Compensation Scheme

The error matrices are captured at discrete points of the measuring volume. To be able to model the behavior of the machine in between the matrices should be interpolated. Componentwise interpolation leads to a non-orthonornal rotational

part which should be avoided. We may consider these matrices as a Lie group and try to do the interpolation using techniques from their theory. However this is a tedious job. Let us decompose the matrix into a rotational part and a displacement vector describing the motion of the machine components as follows:

$$T = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} \delta_x \\ \delta_y \\ \delta_z \end{pmatrix}$$

Here the position dependent displacement vector points can be interpolated by a piecewise polynomial space curve while the rotational matrix remains to be interpolated. However there are simple techniques available based on the application of quaternion.

Quaternion [1] was invented by Sir William Hamilton in 1843. He realized that four numbers are needed to describe a rotation followed by a scaling. One number describes the size of scaling, one the number of degrees to be rotated and the lst two numbers give the plane in which the vector should be rotated. Quaternions consist of a scalar part s$\in\mathbb{R}$ and $\mathbf{v} = (x,y,z)\in\mathbb{R}^3$:

$$q \equiv [s,\mathbf{v}] \equiv [s,(x,y,z)] \equiv s + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z$$

where

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1,\ \mathbf{ij} = \mathbf{k} \text{ and } \mathbf{ji} = -\mathbf{k}$$

If q is a quaternion with $q = [\cos\theta,\sin\theta\mathbf{n}]$ and p is a quartenion $p = [0,\mathbf{r}]$ then $p' = qpq^{-1}$ is p rotated 2θ about the axis **n**.

Given a transformation matrix M the corresponding unit quaternion is can be calculated in two steps: first we must find s which is equal to:

$$s = \pm \frac{1}{2}\sqrt{M_{11} + M_{22} + M_{33} + M_{44}}$$

Now the other values follow:

$$x = \frac{M_{32} - M_{23}}{4s}$$

$$y = \frac{M_{13} - M_{31}}{4s}$$

$$z = \frac{M_{21} - M_{12}}{4s}$$

Where the M-s are the determinants of the respective submatrices.

A so-called spherical linear quaternion interpolation (Slerp) can used to compute the intermediate quaternions. The quaternions generated by Slerp are unit

quaternions, which means that they represent pure rotation matrices. The formula for Slerp is:

$$\cos(\Omega) = q_o \bullet q_1$$

$$Slerp(q_0, q_1, h) = \frac{q_0 \sin((1-h)\Omega) + q_1 \sin(h\Omega)}{\sin(\Omega)}$$

where $\bullet$ stands for the inner product defined as $q \bullet q' = ss' + xx' + yy' + zz'$

An even better (smoother) interpolation can be formulated which is the spherical cubic equivalent of a Beziér curve. This is called Squad and this defined by:

Let $q_1, ..., q_n$ point on the unit sphere. Find the cubic spline which interpolate the points in the given sequence. This can be achieved by the following formula:

Squad $(q_i, q_{i+1}, s_i, s_{i+1}, h)$ = **Slerp**(**Slerp**(q_i, q_{i+1}, h), **Slerp**(s_i, s_{i+1}, h), 2h(1-h))

where s_i are

$$s_i = q_i \cdot \exp\left(-\frac{\log \cdot (q_i^{-1} q_{i+1}) + \log \cdot (q_i^{-1} q_{i-1})}{4}\right)$$

where logq and exp q are defined as follows:

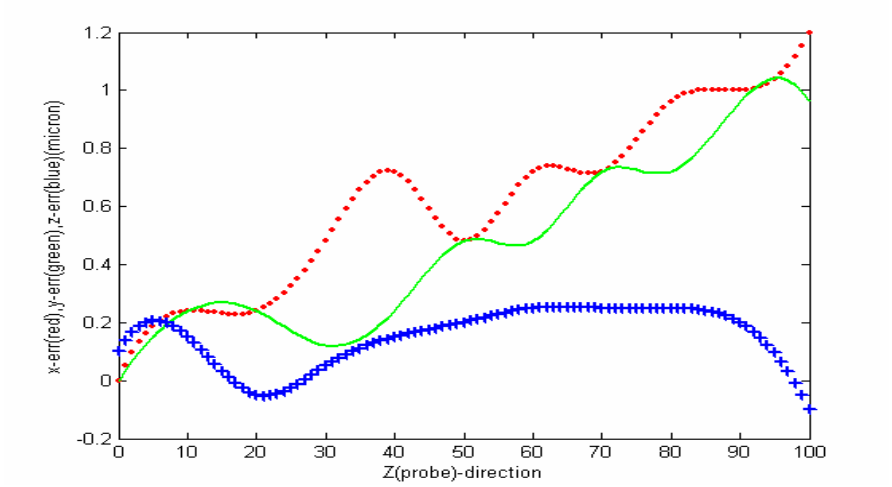
if q = [cosθ,sinθ**v**] then logq ≡ [0,θ**v**] and if q = [0,θ**v**] then expq ≡ [cosθ,sinθ**v**]

The suggested procedure to find the intermediate rotation matrices consists of the following steps:

- Convert the matrices captured by the procedure described in paragraph 4. in quaternions
- Find the **Squad** interpolation of these points
- Convert the quaternion splines back into matrix form

On this way a spherical spline representing pure rotation of the object will be obtained. The translational error can be handled by finding a spatianel interpolation spline function using for example least square fit. In the possession of these functions one can easily reconstruct the real coordinate values of the measured point.

In most of the practical applications, due to the small angular errors spherical linear interpolation is sufficient. In these cases the position originating from straightness and perpendicularity errors are dominant.



In Figure 8 the components of the error vector calculated on the above described way are given for the measuring machine presented in paragraph 4. Similar results are available for the x and the y axis.

Conclusions

The paper presents a new approach to the compensation of geometric errors in coordinate measuring machines. It consists of a measuring procedure which captures simultaneously the six error components of moving rigid body. The transformation matrices obtained on this way are interpolated by using quaternion representation. Hereby the orthonormality of the rotation matrices are maintained. Simulation values with randomly generated error components showed that the intermediate values lead to accuracy improvement.

Acknowledgment

The author gratefully acknowledges the support obtained within the frames of National Research Fund (OTKA) T 0429305.

References

[1] E. B. Dam, M. Koch, M. Lillholm: Quaternions: Interpolation and Animation, Technical Report DIKU-TR-98/5, Department of Computer Science, University of Copenhagen

[2] N. A. Duffie, S. M. Yang: Generation of Parametric Kinematic Error-Correction Function from Volumetric Error Measurement, Annals of the CIRP Vol. 34/1/1985, pp. 435-438

[3] K. C. Fan, M. J. Chen, W. M. Huang: A Six-Degree-of-Freedom Measurement System for the Motion Accuracy of Linear Stages, Int. J. Mach. Tools Manufact. Vol. 38, No. 3, pp. 155-164

[4] K. C. Fan, M. J. Chen: A 6-Degree-of-Freedom Measuring System for the Accuracy of X-Y Stages, Precision Engineering 24(2000) pp. 15-23

[5] G. Farin: Curves and Surfaces for Computer Aided Geoemtric Design: A Practical Guide, Academic Press, Boston, 3rd ed. 1993

[6] P. M. Ferreira, C. R. Liu: An Analytical Quadratic Model for the Geometric error of a Machine Tool, Journal of Manufacturing Systems, Vol. 5, No. 1, pp. 51-62

[7] W. Gao, Y. Arai, A. Shibuya, S. Kiyono, C. H. Park: Measurement of multi-degree-of-freedom error motion of a precision linear air-bearing stage, Precision Engineering 30(2006) pp. 96-103

[8] Gy. Hermann: Design consideration for a Modular High Precision Coordinate Measuring Machine, ICM 2006, IEEE International Conference on Mechatronics, July 3-5, 2006, Budapest, Hungary, pp.161-165

[9] Gy. Hermann: Volumetric Error Correction in Coordinate Measurement. 4th Serbian-Hungarian Joint Symposium on Intelligent Systems (SISY 2006) September 29-30, Subotica, Serbia, pp. 409-416, ISBN 963 7154 50 7

[10] P. S. Huang, J. Ni: On-line Error Compensation of Coordinate Measuring Machines, International Journal of Machine Tools and Manufacturing 1995, No. 3, pp. 725-738

[11] K. D. Kim, S. C. Chung: Accuracy Improvement of the On-Machine Inspection System by Correction of Geometric and Transient Thermal Errors, Transactions of NAMRI/SME Vol. XXXI, 2003, pp. 209-216

[12] S. W. Kim, H. G. Rhee, Ji-Young Chu: Volumetric Phase Measurement Interferometer for Three Dimensional, Precision Engineering 27(2003), pp. 205-215

[13] J.-P. Kruth, P. Vanherck, C. Van den Bergh: Compensation of Static and Transient Thermal Errors on CMMs, Annals of the CIRP Vol. 50/1/2001, pp. 377-380

[14] Lion Precision:

[15] T. A. M. Ruijl: Ultra Precision Coordinate Measuring Machine, PhD. Thesis TU Delft 2001

[16] SIOS: Messtechnik GmbH: Miniatur interferometer mit Planspiegel-reflektor SP 2-12/50/2000

[17] SIOS: Messtechnik GmbH: Nano measuring machine

[18] A. H. Slocum: Precision Machine Design, Englewood Cliffs, NJ: Prentice Hall, 1992

[19] K. K. Tan, S. N. Huang, S. Y. Lim, Y. P. Leow, H. C. Liaw: Geometric Error Modeling and Compensation Using Neural Networks, IEEE Transaction on Systems, Man and Cybernetics, Vol. 36, No. 6, Nov. 2006, pp. 797-809

[20] J. W. M. C. Teeuwsen, J. A. Soons, P. H. J. Schellekens: A General Method for Error Description of CMMs Using Polynomial Fitting Procedures, Annals of the CIRP Vol. 38/1/1989, pp. 505-510

[21] S. M. Wang, K. F. Ehmann: Measurement Methode for Position Error of a Multi-axis machine – Part I: Principle and Sensitivity Analysis, International Journal of Machine Tools and Manufacturing 39(1999) 951-964

[22] S. M. Wang, K. F. Ehmann: Measurement Methode for Position Error of a Multi-axis machine – Part II: Application and and Experimental Results, International Journal of Machine Tools and Manufacturing 39(1999) 951-964

[23] G. Zhang, R. Ouyang, B. Lu: A Displacement Method for Machine Geometry Calibration, Annals of the CIRP Vol. 37/1/1988, pp. 515-518

[24] G. X. Zhang, J. Y. Fu: A Methode for Optical CMM Calibration Using a Grid Plate, Annals of the CIRP Vol. 49/1/2000, pp. 399-402

[25] G. Zhang, R. Veale, T. Charlton, B. Borchardt, R. Hocken: Error Compensation of Coordinate Measuring Machines, Annals of the CIRP Vol. 34/1/1985, pp. 445-448

Navigation of Mobile Robots Using Potential Fields and Computational Intelligence Means

Ján Vaščák

Centre for Intelligent Technologies, Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University in Košice
Letná 9, 042 00 Košice, Slovakia
jan.vascak@tuke.sk

Abstract: A number of various approaches for robot navigation have been designed. However, practical realizations strike on some serious problems like imprecision of measured data, absence of complete knowledge about environment as well as computational complexity and resulting real-time bottlenecks. Each of proposed solutions has some of mentioned drawbacks. Therefore one of possible solutions seems to be in combining several means of computational intelligence. In this paper a combination of harmonic potential fields, neural networks and fuzzy controllers is presented. Also some simulation experiments are done and evaluated.

Keywords: harmonic potential fields, neural networks, fuzzy controller

1 Introduction

Theoretically the problem of finding the shortest or the safest path between two points in an environment with obstacles can be well defined and solved. Basically, there are three groups of approaches: heuristic, exact and grid algorithms (a good overview can be found in [4]). The first group is represented mainly by Bug algorithms, which are simple and suitable for environments with static obstacles. Exact algorithms, mainly visibility graphs and Voronoi diagrams, enable a mathematically correct way for finding the best solution. If there is no possible path (too restrictive obstacles) then they will be able to give a definite answer 'no'. However, they require precise sensing of obstacles and are suitable above all only for static environments. Algorithms based on grid description are more convenient for practical use because the precision of sensing is limited.

Potential fields are a part of grid algorithms. They use the metaphor of magnetic field or gas spreading, firstly mentioned in [5]. It is possible to determine the precision of the grid or to design multi-layered grids, which enable only roughly

to navigate a robot in a simple environment with only few obstacles and in the case of a more complicated area to switch to a more detailed grid description. There are also methods, which do not require having a complete map a priori. It can be constructed in steps, too [2]. It means they can be used also in dynamic environments. The only difference between static and dynamic environment is that in the second case the environment is changing during the movement of a robot, e.g. we consider vehicles or walkers.

However, also potential fields have some lacks, especially their computational complexity. Therefore, in this paper a structure combining potential fields with a neural network and fuzzy controller is proposed. The next section will describe basic principles of potential fields. Then the proposed structure will be described in detail and finally, some experiments and conclusions will be mentioned.

2 Potential Fields

In general, there are two principal objects in each environment: goals (we will further consider only one goal) and obstacles. The goal should attract a robot to its position and obstacles should repulse it from them. Therefore we will speak about attractive $\vec{F}_G$ and repulsive $\vec{F}_O$ strengths, respectively. Considering a two-dimensional space *X x Y* the total strength $\vec{F}$ in a certain point will be given as a vector sum:

$$\vec{F}(x,y) = \vec{F}_G(x,y) + \sum_i \vec{F}_{O_i}(x,y). \tag{1}$$

A set of all strengths for all combinations of *(x, y)* creates a *vector field* $\vec{F}$ (see Fig. 2).

Potential $U(x,y)$ is numerically equal to the work done between this point and a point with a zero potential (x_0, y_0) along the path *l*, i.e.:

$$U(x,y) = \int_{(x_0,y_0)}^{(x,y)} \vec{F}.dl\,. \tag{2}$$

Again, the depiction for all combinations of *(x, y)* creates a *potential field* U (see Fig. 1).

Let us consider an imagination of a landscape. We let to go down an element. Due to gravity it will move to a point with the minimum height level (a ditch) along a path with the maximum descent (a valley) avoiding all areas with higher levels (hills). Now we can define the goal as the minimum point and obstacles as areas with higher height levels. In such a way we can obtain the fastest path to get to the

goal. There are lots of possible definitions for the goal and obstacles [2, 5]. They enable us to define a safety range outside of obstacles, whose original positions are depicted as black areas in Figs. 1 and 2, so the border among obstacles and free area is not crisp but rather fuzzy. In other words, to get to the safety range it does not mean the robot will implicitly strike on an obstacle but it is not safe. This range can be simulated by continuously increasing functions in the direction to obstacles. In such a way we can obtain a compromise between speed and safety, which is the most important criterion in real applications.

Potential fields represent the description of the environment, which can be obtained completely a priory at the start of the motion process or sequentially (per parts) dependent by the radius of sensors, which can decrease time costs. Vector fields represent a map of actuator values, the orientation and magnitude. Therefore to fulfil the above criterion the relation between a potential and a vector field is defined as follows:

$$\vec{F} = -\nabla U. \tag{3}$$

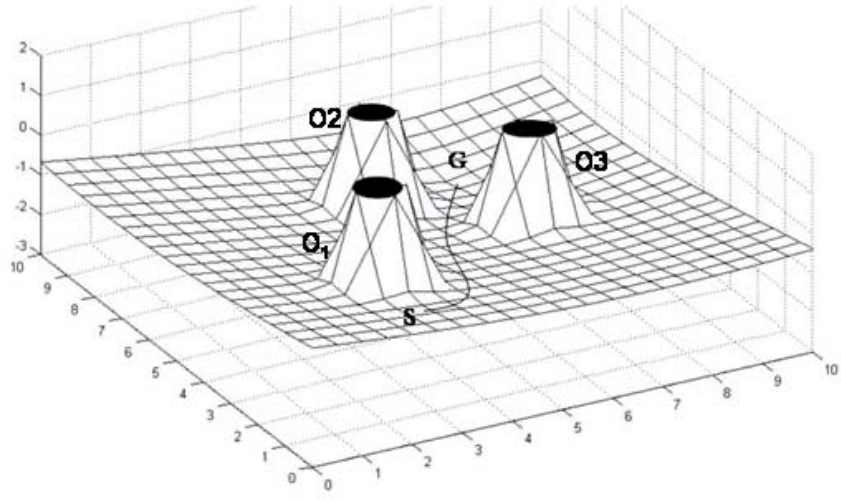


Figure 1
Example of a potential field with 3 obstacles O_1, O_2, O_3 and one goal G with computed path from the starting point S

Basically, the navigation consists of two stages. Firstly, from sensed data the potential field is calculated. Then using (3) the vector field (Fig. 2) is calculated. It represents all possible solutions from all possible starting positions. Both stages are evidently computationally very demanding. However, we usually need only a part of the vector field, which simplifies the computation.

In Fig. 2 we can see the actuator values decrease very quickly from obstacles to the goal. It means we must use data types with high precision and usually for more than 1 000 iterations it is necessary to define special extended data types and together with them special arithmetic, which makes the computation still more complex. Another possibility is finding suitable definitions for obstacles and goal but this way is often application dependent.

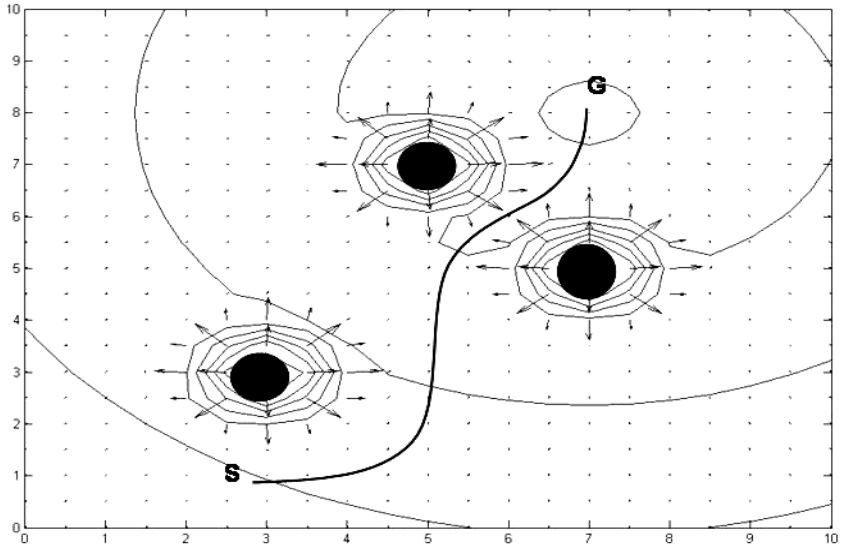


Figure 2
Vector field of actuator values computed from the potential field in Fig. 1. Black circles represent obstacles

The mentioned example is only a simple one. It has only one minimum, which is also global. However, there are also situations with several minima. The task is to navigate a robot to the global minimum not to local ones. Especially, the so-called 'robot traps' [4] (Fig. 3) can be created in the case of non-convex obstacles. If the robot enters such a trap then the algorithm will not be able to pull it from the trap and the robot 'freezes'.

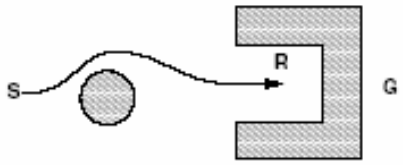


Figure 3
A robot trap. The robot jams in the point *R* instead of *G*

There are two basic ways how to solve this problem: either to design methods for recognizing such a danger, which is again application dependent or to use such potential fields, which have only one minimum. *Harmonic potential fields* posses just such a property, firstly used in [6].

1.1 Harmonic Potential Fields

By the definition, harmonic functions U have to fulfil for all dimensions x_i the condition of Laplace's equation [3]:

$$\nabla^2 U = \sum_{i=1}^{n} \frac{\partial^2 U}{\partial x_i^2} = 0. \tag{4}$$

For simplicity let us consider a one-dimensional function $U(x)$. Using Taylor series we will get values in adjacent points of $U(x, y)$, i.e. $U(x+1)$ and $U(x-1)$ as follows:

$$U(x+1) = U(x) + \frac{\partial U(x)}{\partial x} + \frac{1}{2}.\frac{\partial^2 U(x)}{\partial x^2} + \cdots \tag{5}$$

$$U(x-1) = U(x) - \frac{\partial U(x)}{\partial x} + \frac{1}{2}.\frac{\partial^2 U(x)}{\partial x^2} - \cdots \tag{6}$$

Neglecting further members in (5), (6) and their subsequent summation will result in:

$$U(x+1) + U(x-1) - 2.U(x) \approx \frac{\partial^2 U(x)}{\partial x^2} = 0. \tag{7}$$

The right part of (7) is equal to zero because of (4). Similar situation will be in a two-dimensional space (X x Y) where we will have 4 adjacent points to $U(x, y)$. We can see $U(x, y)$ is approximate to the arithmetic average of its neighbours. Further, from the simplicity reason, we will not more use the symbol of approximation but equivalence:

$$U(x,y) = \frac{\big(U(x+1,y) + U(x-1,y) + U(x,y+1) + U(x,y-1)\big)}{4}. \tag{8}$$

The equation (8) expresses energetic balance, too. If the point (x, y) is neither an obstacle nor the goal then the sum of all potentials must give zero, which also corresponds to the physical concept.

In such a way, we can describe all points (x, y) in the area and we will get a system of $||X||$ x $||Y||$ (let us say n) linear equations. Solving such a system is computationally a very arduous task. To speed up the computation the use of an iterative method is necessary. The Gauss-Seidel method seems to be the best

possibility. Having a system of linear equations $A.x=b$ the vector x of variables x_i (in our case $U(x_i, y_j)$) will be calculated as [1]:

$$x_i^k = \frac{b_i - \sum_{j<i} a_{ij}.x_j^k - \sum_{j>i} a_{ij}.x_j^{k-1}}{a_{ii}}, \tag{9}$$

where k is the iteration step and i, j are indexes of ordered equations. This method is computationally simpler than the Jacobi method (which is similar to it). We can see for calculating x_i^k the values from preceding equations (with order less than i) are already from the same iteration cycle.

Using (9) for (8) we will get:

$$U^k(x_i, y_j) = \frac{1}{4}.\left(U^{k-1}(x_{i+1}, y_j) + U^k(x_{i-1}, y_j) + U^{k-1}(x_i, y_{j+1}) + U^k(x_i, y_{j-1})\right). \tag{10}$$

As starting values for $k=0$ we will set the points (x, y) in obstacles to the maximum, (x_G, y_G) to the minimum and other points to zero. If changes of solutions in (10) fall in the tolerance, then the potential field will be constructed and further iterations stopped.

3 Hybrid Navigation Structure for Parking Problem

Potential fields can be principally used also for environments with dynamic obstacles, which is our case of a parking navigation system when we take into consideration moving cars (robots). However, this supposes recalculation of the potential field in each sampling step, which is impossible to do it in real-time. Therefore we proposed a hybrid structure with these elements and tasks:

1 *Harmonic potential field* – for calculation of the path in the initial step. All obstacles (parking boxes as well as cars) are considered as static. The path is described as series of orientation marks.

2 *Neural network* – as a controller (with back-propagation learning) trying to control the robot to pass through the orientation marks.

3 *Fuzzy controller* – as a Mamdani type controller solving a problem if a car starts to move (in other words, if it becomes to a dynamic obstacle). Then it will take over the control from the neural network and will do obstacle avoidance trying again to find orientation marks.

Using knowledge about the structure of a given car park and current occupation of individual parking boxes, see Fig. 4, (sensed by a type of appropriate sensors, e.g. cameras) a potential field is created, see Fig. 5. At this moment we suppose, the whole area is static (maybe it will also stay static). Then a vector field is computed. After determination of the starting point we will get the resulting path (the safest and shortest), see Fig. 6, and the robot will start its motion. In each sampling step it is controlled by the neural networks and also the situation in the car park is observed. There are controlled the turning angle and acceleration. If the environment stays static then control will continue by the neural network if no then the control will be switched to the fuzzy controller. It uses principles of sonar perception, detailed described in [7]. Its use is necessary if the obstacle intersects the computed path. The controller has two particular tasks, which are solved simultaneously: obstacle avoidance and searching for orientation marks. They issue to bringing the robot again to orientation marks. After that the control is again switched to the neural network. The whole process can be seen in the environment of a simulator, Figs. 7 and 11.

4 Experiments

Here two experiments will be described. The first one E1 can be seen in Figs. 4-7 and the simulation of the second one E2 in Fig. 11. Three basic characteristics were observed in dependence of sampling steps: control type, turning angle and acceleration.

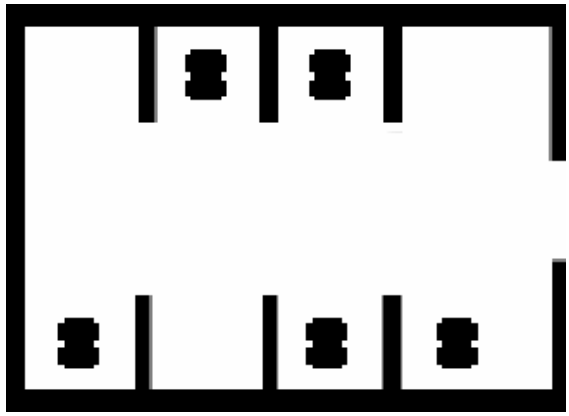


Figure 4
E1 - Sensed situation of the car park

At the first experiment the environment was all the time static and the goal was reached successfully. The characteristics are in Figs. 8-10.

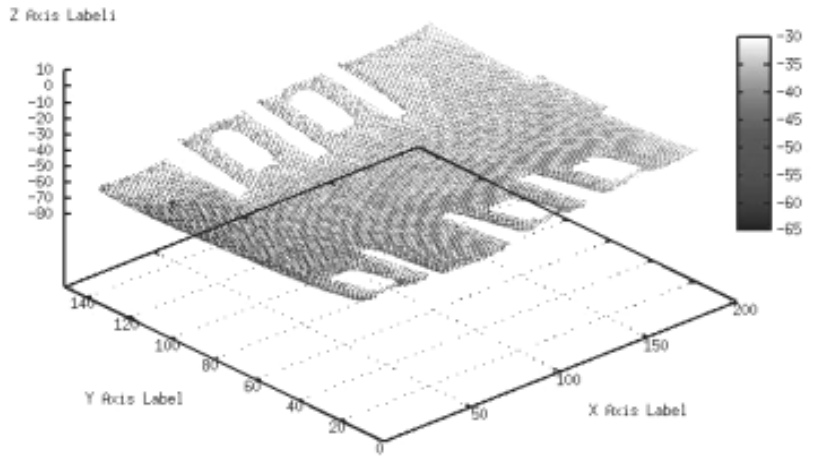


Figure 5
E1 - Created potential field

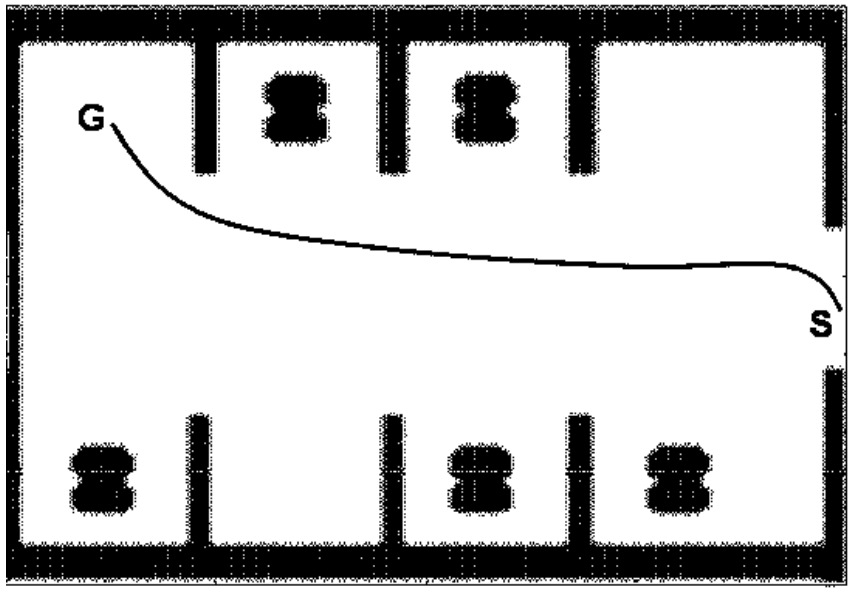


Figure 6
E1 - Computed path form the starting point *S* to the goal *G*

At the second experiment during the navigation another car intersected the chosen path so the fuzzy controller had to do obstacle avoidance. In spite of that the goal was reached successfully. The characteristics are in Figs. 12-14.

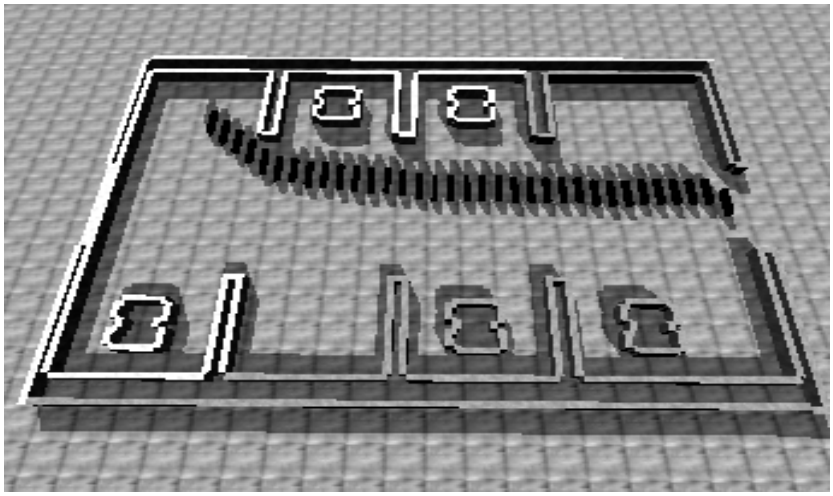


Figure 7
E1 - Simulated navigation in static environment

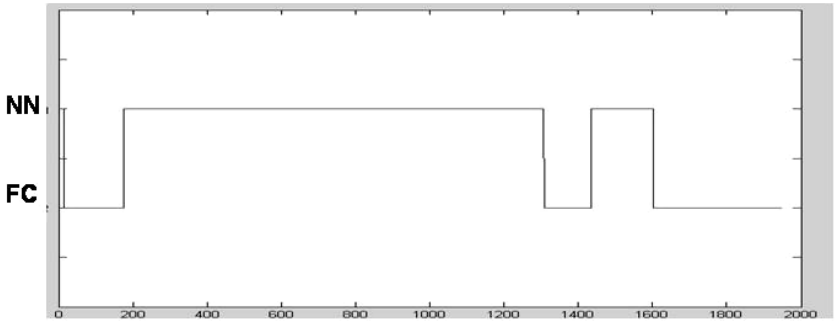


Figure 8
E1 - Control type, FC – fuzzy controller, NN – neural network

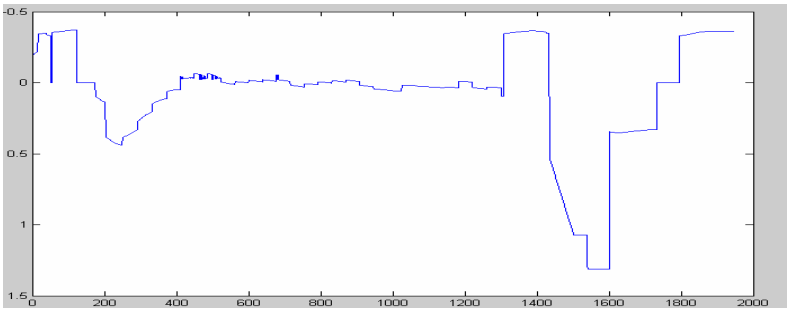


Figure 9
E1 - Turning angle in radians

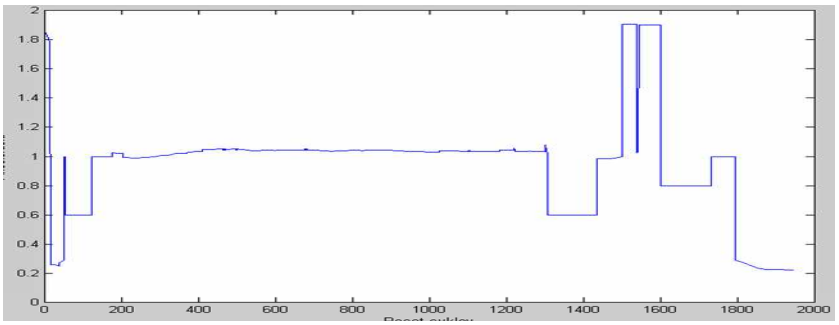


Figure 10
E1 – Acceleration in $m.s^{-2}$

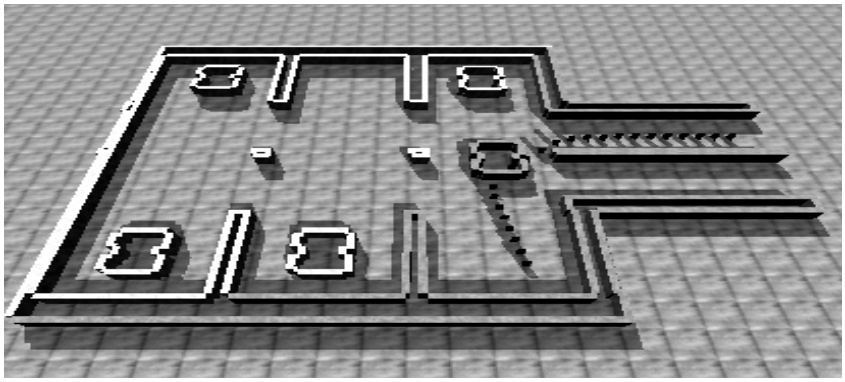


Figure 11
E2 - Simulated navigation in dynamic environment

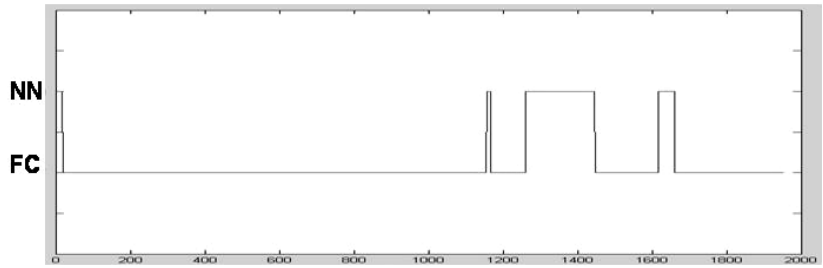


Figure 12
E2 - Control type, FC – fuzzy controller, NN – neural network

In both experiments, Figs. 8-10 and 12-14, we can observe certain chattering. The changes of turning angle and acceleration are in some moments significant. This is caused in times when the control is changed between the fuzzy controller and neural networks, which is a well known drawback of hybrid systems.

Changes of the control type occur also in the experiment E1, i.e. not only during avoidance of a dynamic obstacle as mentioned in above. This state arises always when the robot approaches to an obstacle too close or robot sensors lose the contact with orientation marks for a while then the situation will be evaluated as a dynamic obstacle and the fuzzy controller will take over the control.

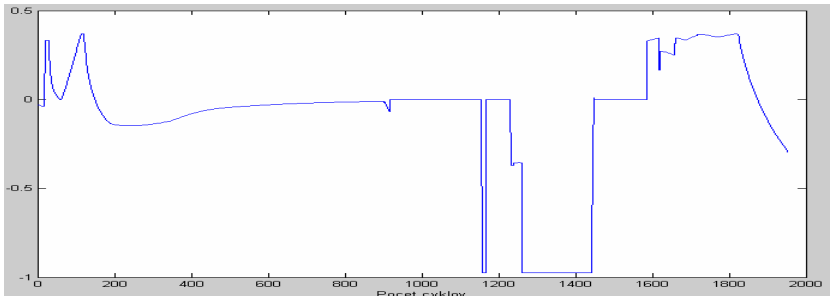


Figure 13
E2 - Turning angle in radians

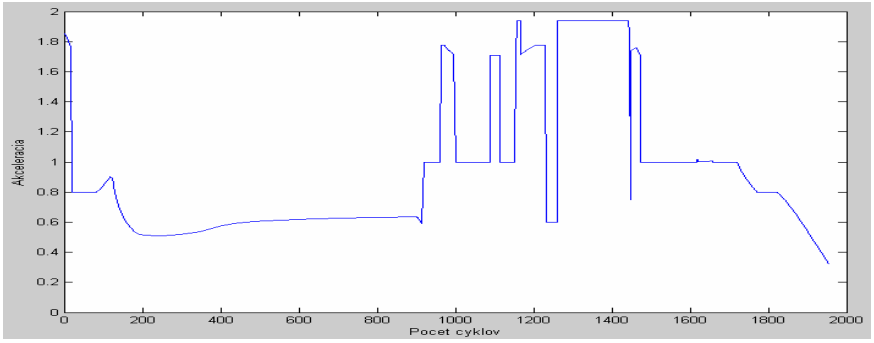


Figure 14
E2 - Acceleration in $m.s^{-2}$

Conclusions

The presented work had the aim to show the possibility of use harmonic potential fields together with systems of computational intelligence like fuzzy controllers and neural networks. All three approaches are based on numerical calculations and all three ones are derived from metaphors existing in the real world. The aim of this paper is also to contribute to discussions about using the so-called 'dirty' methods in artificial intelligence, whether yes or no. The answer in the opinion of the author is unambiguously 'yes'.

There are still a number of problems that were discovered during experiments (some experiments failed) but obtained results confirm applicability of the chosen approach. As further topics for research in this area might be regarded these:

- Design of methods for quicker constructing of potential fields.
- Modification of computation methods for handling only such area parts, which are necessary for trajectory design.
- Using previous two points in such a manner, which would reinforce the computational efficiency and thereby it would be possible directly to implement potential fields into environments with dynamic obstacles and to recalculate them in each time step [2].
- Design of functions for constructing potential fields, which could not cause rapid decrease of values in the vector field.
- More sophisticated switching between various controllers to eliminate the effect of chattering.
- Design of further approaches (also based on heuristics) being able to solve more complicated situations and various 'traps'.

References

[1] Barrett, R. at al.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd ed., Philadelphia, PA: SIAM, 1994, (http://www.netlib.org/linalg/html_templates/Templates.html), ISBN 0-89871-328-5

[2] Bell, G., Weir, M. K.: Forward Chaining for Robot and Agent Navigation using Potential Fields, In: Proceedings of 27th Australasian Computer Science Conference (ACSC2004), Australian Computer Science Communications, Vol. 26, N. 1, Dunedin, New Zealand, pp. 265-274, ISBN 1-920682-05-8, ISSN 1445-1336, 2004

[3] Connolly, C. I.: Harmonic Functions and Collision Probabilities, In: Proceedings of the IEEE International Conference on Robotics & Automation (ICRA), pp. 3015-3019, 1994

[4] Dudek, G., Jenkin M.: Computational Principles of Mobile Robotics, Cambridge University Press, Cambridge, ISBN 0-521-56021-7, 2000

[5] Khatib, O.: Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, In: Proceedings of IEEE International Conference on Robotics and Automation, Vol. 2, pp. 500-505, 1985

[6] Koditschek, D.: Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations, In: Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 4, pp. 1-6, 1987

[7] Vaščák, J.: Parameter Adaptation of a Fuzzy Controller by Genetic Algorithms (in Slovak); In: Kognice a umělý život V., Vol. 2, Smolenice, Slovakia, 2005, pp. 589-600, ISBN 80-7248-310-2

P-Graph-based Workflow Modelling

József Tick

Institute for Software Engineering, John von Neumann Faculty of Informatics,
Budapest Tech
Bécsi út 96/B, H-1034 Budapest, Hungary
tick@bmf.hu

Abstract: Workflow modelling has been successfully introduced and implemented in several application fields. Therefore, its significance has increased dramatically. Several work flow modelling techniques have been published so far, out of which quite a number are widespread applications. For instance the Petri-Net-based modelling has become popular partly due to its graphical design and partly due to its correct mathematical background. The workflow modelling based on Unified Modelling Language is important because of its practical usage. This paper introduces and examines the workflow modelling technique based on the Process-graph as a possible new solution next to the already existing modelling techniques.

Keywords: workflow management, workflow modelling, P-graphs

1 Introduction

Workflow management was originally a tool for the organisation, analysis and rearrangement of business processes. Information technology has gone through one of the most significant changes in the last decades while having proliferated and penetrated into business processes in each and every business field and has become a determining component. In parallel the application of workflow has improved as well. By nowadays the number of application fields has increased a lot and after successes in the production and administrative processes, workflow management gained ground in almost each professional field. Further application fields include a wide range of sectors ranging from technical design [4] via solving of various computing problems [3] to several application of the processing of information data [11].

The drastic development in workflow applications has meant the computerisation of the existing processes and later, with the appearance of the Business Process Management (BPM), the total restructuring of these business processes. BPM examines and models the processes from various aspects, which meant the information based restructuring and reorganisation of this field.

1.1 Key Definitions of Workflow and the Elements

The concept of workflow is widely interpreted, and it might mean simple activity steps but can be interpreted as a synonym of the total Business Process. The Workflow Management Coalition (WfMC) as the acknowledged professional association strives to reach standardisation and also conducts widespread marketing to spread the concepts and methodologies linked to Workflow technology. Consequently, this paper takes the WfMC [1] definitions and interpretations for granted when reviewing the definitions.

Workflow: 'The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.' [1]

Activity: 'A description of a piece of work that forms one logical step within a process. An activity may be a manual activity, which does not support computer automation, or a workflow (automated) activity. A workflow activity requires human and/or machine resources(s) to support process execution; where human resource is required an activity is allocated to a workflow participant.' [1]

Process: 'The representation of a business process in a form which supports automated manipulation, such as modelling, or enactment by a workflow management system. The process definition consists of a network of activities and their relationships, criteria to indicate the start and termination of the process, and information about the individual activities, such as participants, associated IT applications and data, etc.' [1]

Instance: 'The representation of a single enactment of a process, or activity within a process, including its associated data. Each instance represents a separate thread of execution of the process or activity, which may be controlled independently and will have its own internal state and externally visible identity, which may be used as a handle, for example, to record or retrieve audit data relating to the individual enactment.' [1] Instances are process instances, which are representations of a single enactment of a process, or activity instances, which are representations of an activity within a single enactment of a process (within a process instance). Process instances are often called cases.

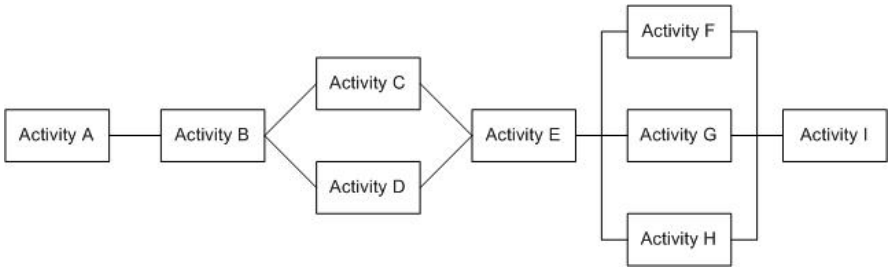


Figure 1
Example of a Workflow using WfMC's Notation

Figure 1 presents a simple example using the above definition and given the notation system provided by WfMC.

2 The Existing Workflow Modelling Methodologies

Taking workflow modelling into account five relevant workflow model perspectives can be differentiated based on the above [2], [5] and [8]. The **Control flow or process perspective** presents the workflow elements, their relationships, that is the static structure and organisation of the workflow. The control flow includes the time dependency between the elements, and the entire routing description valid for the workflow model. The **Resource or organization perspective** determines the types, the quantity and/or the availability and accessibility of the resources necessary for the execution of the tasks. It describes the roles from the perspective of functionality, and the groups from the aspect of the organisation as well as their labelled responsibility, authorisation and availability. The **Data or information perspective** includes the description of the control data needed for the operation of the workflow and the execution of the routine, and in the framework of production information the data, tables and documents containing the significant characteristics of production. The **Task or function perspective** defines the elementary operations carried out by the resources while performing a task. The **Operation or application perspective** specifies the elementary actions using specified applications. These applications could be general applications such as text editor, spreadsheet editor, or special applications developed for the given task. Traditional modelling focuses mainly on the first perspective.

This paper highlights two modelling techniques out of the several existing ones: the modelling technique based on the UML activity diagram and the modelling based on the Petri-Net.

Unified Modelling Language (UML) is a widespread modelling language in software engineering that is highly standardised and is rich in tool system. The control perspective of the Workflow model consists of a description process activities and its routing in cases [7]. This is the reason why UML's sequence diagram and activity diagram is useful to model the discussed aspects of workflow models. The research work done by W. van der Aalst et al. resulted in 21 several patterns which describe the behaviour of business processes. [7] S. A. White mapped this patterns from business process model to UML activity diagram. The usage of UML reflects a rather practice-oriented view, which is closely linked to the everyday work of software developers.

Petri-Net is one of the most important mathematical and graphical representation possibilities of distributed systems [12], networks and workflows. It has been a

popular workflow modelling tool for a long time [13]. Introducing Petri-Nets to the field of workflow modelling Aalst and Hee [5] specified a process using Petri-Net as the basic element of workflow. Van der Aalst with his more than three hundred publications worked out the whole theory and methodology of the Petri-Net based workflow management. He describes [14] how to map workflow management concepts onto Petri-Nets, defines the processes, the control flow possibilities, routing constructs, triggering, tasks, work items and activities. He focuses also on the analysis of workflow using Petri-nets.

3 Process Modelling Using P-Graphs

The P-graph based modelling as well as the process network synthesis generated by combinatorial methods are presented by using the works published by Friedler et al. The chapter discusses the mathematical description of the P-graphs mainly using the literature [15], [16], [17], [18], [19], [20].

3.1 Introduction of P-Graphs

The so called P-graph (Process graph), which is a directed bigraph, has been used for modelling network structures for some time. The vertices of the graph denote the operating units (***O*** – operating units) and the materials (***M*** – materials). The edges of the graph represent the material-flow between the materials and the operating units.

The P-graph is a bigraph, meaning that its vertices are in disjunctive sets and there are no edges between vertices in the same set. In case of P-graphs the assignment of operating units and materials are strictly determined by the tasks given, i.e. an edge can point to an ***M*** material type vertex from an ***O*** operating unit type vertex, only if ***M*** is element of the output set of ***O***, that is ***O*** produces ***M*** material namely, $\boldsymbol{M} \in \text{output}_{\boldsymbol{O}}$. An edge can point from an ***M*** material type vertex to an ***O*** operating unit type vertex, only if ***M*** is element of the input set of ***O***, that is ***O*** processes ***M*** material, namely $\boldsymbol{M} \in \text{input}_{\boldsymbol{O}}$. Thus, the P-graph can be presented by the pairs of operating unit and the assigned material vertices set like the (***M,O***) P-graph.

The material type vertices can be put into several subsets. There are various subsets like the raw-material type one, which contains the input elements of the whole process, the product-material type subset, which gathers the results of the entire process, the intermediate-material type one, the elements of which emerge or are used between the processing phases, and finally the by-product-material type set, which contains the non desired results of the process.

The applied operating unit and material element notations in the P-graph notation are presented in Figure 2. As an example let us consider a process network with ***7***

operating units, in which the operating units are ***1,2….7*** and the materials are ***A,B, …. L. A,B,C*** and ***D*** are the materials available for the production of ***L***. The possible structure is given in Figure 3.

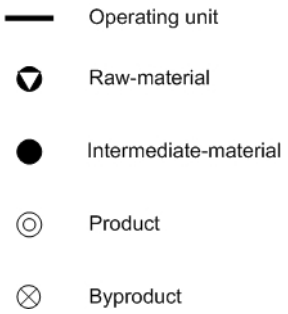


Figure 2
Notations used at P-graphs

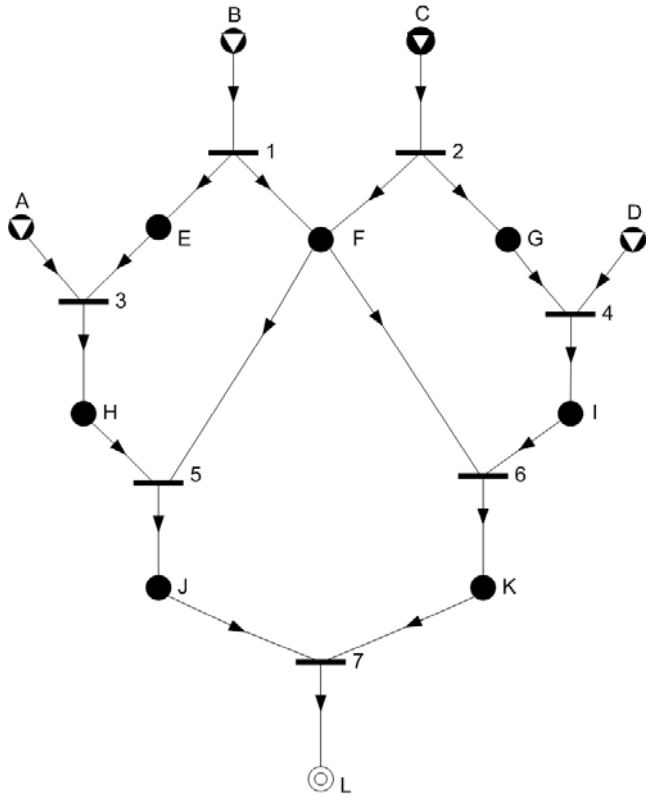


Figure 3
The notation system of a P-graph model

3.2 The Mathematical Definition of P-Graphs

There is a finite set of material $\boldsymbol{M}$ (which contains the sets of $\boldsymbol{P}$ products and $\boldsymbol{R}$ raw-materials) and the finite set of $\boldsymbol{O}$ operating units. Consequently, the set of $\boldsymbol{P}$ end-products and the set of $\boldsymbol{R}$ raw-materials must be subsets of M and the set of $\boldsymbol{M}$ materials and the set of $\boldsymbol{O}$ operating units are disjunctive. The basic relations between $\boldsymbol{M,P,R}$ and $\boldsymbol{O}$ are as follows:

$$\boldsymbol{P} \subseteq \boldsymbol{M}, \boldsymbol{R} \subseteq \boldsymbol{M}, \text{ and } \boldsymbol{M} \cap \boldsymbol{O} = \emptyset \tag{3.1}$$

As physical processes are defined, each operation unit produces output materials from input materials. Therefore two disjunctive sets can be assigned to each operating unit, i.e. the set of input and the set of output materials. Let an arbitrary operating unit (α, β), then α is the set of input materials which are processed by the (α, β) unit and β is the set of output materials, which are produced by the given unit. Considering the process-network the output materials of each operating unit are the inputs of different operating units. In general, it can be proved that

$$\boldsymbol{O} \subseteq \wp(\boldsymbol{M}) \times \wp(\boldsymbol{M}) \tag{3.2}$$

where $\boldsymbol{O}$ is the set of operating units, $\boldsymbol{M}$ is the set of materials and $\wp(\boldsymbol{M})$ is the power set, that is the set of subsets of $\boldsymbol{M}$, and $\wp(\boldsymbol{M}) \times \wp(\boldsymbol{M})$ represents the set of $\wp(\boldsymbol{M})$ and $\wp(\boldsymbol{M})$ pairs.

Supposing that there is a finite set $\boldsymbol{m}$, which is a subset of $\boldsymbol{M}$, i.e. it is true that $\boldsymbol{m} \subseteq \boldsymbol{M}$ and there is an $\boldsymbol{o}$ finite set, which is a subset of $\boldsymbol{O}$, i.e. it is true that $\boldsymbol{o} \subseteq \boldsymbol{O}$ and supposing that there is such a material which is an input for one or more operating units, and there is such material which is the output of one or more operating units, then

$$\boldsymbol{o} \subseteq \wp(\boldsymbol{m}) \times \wp(\boldsymbol{m}) \tag{3.3}$$

The P-graph is defined as a bigraph, where the set of $\boldsymbol{V}$ vertices is made of the elements of the union of $\boldsymbol{m}$ and $\boldsymbol{o}$ that is

$$\boldsymbol{V} = \boldsymbol{m} \cup \boldsymbol{o} \tag{3.4}$$

Obviously, the vertices that are elements of $\boldsymbol{m}$ are $\boldsymbol{M}$ (material) type vertices, while the vertices that are elements of $\boldsymbol{o}$ are $\boldsymbol{O}$ (operating units) type ones.

It is true for the set of $\boldsymbol{A}$ edges of P-graph that the elements of $\boldsymbol{A}$ are the elements of the union of $\boldsymbol{A_1}$ and $\boldsymbol{A_2}$, i.e.

$$\boldsymbol{A} = \boldsymbol{A_1} \cup \boldsymbol{A_2} \tag{3.5}$$

where

$$\boldsymbol{A_1} = \{(x, y) \mid y = (\alpha, \beta) \in o \text{ és } x \in \alpha\} \tag{3.6}$$

and

$$\boldsymbol{A_2} = \{(y, x) \mid y = (\alpha, \beta) \in o \text{ és } x \in \beta\} \tag{3.7}$$

On the basis of (3.5)…(3.7) the edges of the graph can be $\boldsymbol{A_1}$ and $\boldsymbol{A_2}$. The $\boldsymbol{A_1}$ type edge, which is determined by (x,y) points from an input material vertex belonging to the operating unit to the operating unit vertex, considering that x is element of the input set α, while y is the element of $\boldsymbol{o}$ operating unit set, which is determined by the α and β material set pairs. Opposing the $\boldsymbol{A_2}$ type edge, which is determined by (y,x), points from an operating unit vertex to an output material vertex belonging to the operating unit, considering that y is the element of the $\boldsymbol{o}$ operating unit set determined by α and β material set pairs, while **x** is the element of the β output material set.

3.3 The Combinatorial Solution of the Process-Network Synthesis (PSN)

The primary aim of the process-network is to produce $\boldsymbol{P}$ products from $\boldsymbol{R}$ raw-materials. As the first step of PNS, all the plausible operating units $\boldsymbol{O}$ and intermediate-materials must be determined. By determining $\boldsymbol{P,R}$ and $\boldsymbol{O}$, the set of all the material in the network, $\boldsymbol{M}$ is also defined.

The optimal solution structure generated by process-network synthesis must have several basic features that are taken for granted as axioms, and the introduction of which improves the efficiency of the combinatorial search during the process. These axioms are the following:

(S1) Each final product is represented in the graph.

(S2) An $\boldsymbol{M}$-type vertex does not have an input if and only if it represents a raw-material.

(S3) Each $\boldsymbol{O}$-type vertex representing an operating unit is defined in the network synthesis problem.

(S4) There must be at least one route from each $\boldsymbol{O}$-type (operating unit) vertex represented in the structure leading to an $\boldsymbol{M}$-type vertex representing a product.

(S5) If an $\boldsymbol{M}$-type vertex belongs to the graph, then there must be at least one route leading to an $\boldsymbol{O}$-type vertex or a route from an $\boldsymbol{O}$-type vertex to the given $\boldsymbol{M}$-type vertex.

The number of all the combinatorially possible networks increases exponentially by the increase in the number of operating units. For example, in case of a process-network synthesis made up of 35 operating units [20] the number of all the potentially possible structures is 2^{35}-1 that is 34,359,738,367. Supposing an average PC, continuous calculation, and assigning 10^{-2} sec processing time for each combination, the calculation and management of all the 34 billion combination would take more than 11 years.

The structures of the solutions must carry the features defined in the axioms. This is, however, only necessary but not satisfactory condition of the selection of the optimal structure. A practical reduction has been carried out in the number of the structures to be handled with the help of the axioms, thus leaving out and eliminating the redundant and not valid structures. The remaining structures, which fulfil the axioms, are called the combinatorially possible structures. With the help of this practical restriction the search field has been drastically reduced. (see Figure 4) Taking the previous example the number of the structures drops from 34 billion to 3465. Therefore the actual processing time drops from 11 years to 11 and half minutes.

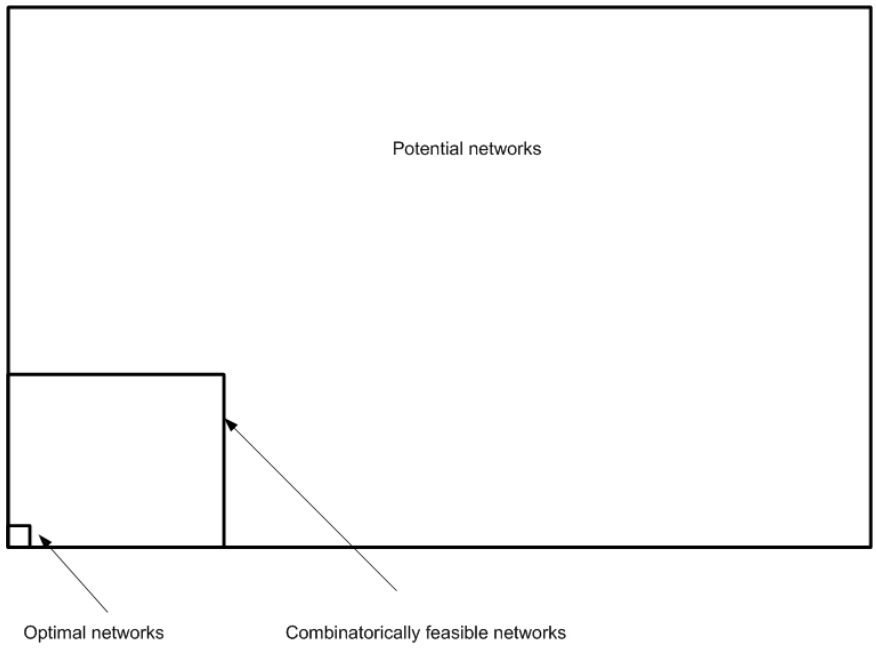


Figure 4
The decrease in search field with the help of the axioms

Such a reduction in the size of the search field, i.e. the exclusion of those structures which are at first sight cannot be taken as possible optimal structures that are the determination of the combinatorially possible structures cannot be

carried out in a mathematically precise way with the conventional methods of the process synthesis. The identification of all the possible structures and the selection of the optimal structure based on the conventional super-structure methods like MILP or MINLP cannot be used due to the features of the applied exponential algorithms. The Maximal Structure Generation (MSG) polynomial algorithm elaborated by Fiedler et al., which uses the 5 axioms, generates that maximal structure, all the combinatorially possible structures of which are its subsets.

3.4 The Generation of the Maximal Structure, the MSG Algorithm

The maximal structure of the synthesis problem (***P***, ***R***, ***O***) contains all the combinatorially possible structures, which make the production of defined products possible from given raw-materials. Therefore, it certainly contains the optimal structure as well.

The algorithm can be divided into four main phases:

The first phase is the input phase, in which the synthesis problem is defined (***P***, ***R***, ***O***) such a way, that the set of ***M*** all the plausible materials, the set of P end-products, the set of ***R*** raw-materials and the set of ***O*** operating units are given. M contains not only the intermediate-materials assigned to operating units defined in the set of ***O***, but the raw-materials given in ***R*** and the end-products given in ***P*** as well.

The second phase is the elaboration of the input structure of the network, which is carried out by the linking of all the similar (same type) material type vertices.

The third phase is the elimination phase, where those materials and operating units are eliminated, which, taking the 5 axioms into account, are not and cannot be linked to the maximal structure for sure. The elimination is carried out step by step, starting from the deepest level of the input structure, from the raw-materials, and going from level to level by examining the vertices at the material level and the operating unit level in turns to see whether that fulfil all the 5 axioms. Certainly, the elimination of a vertex often leads to the elimination of further vertices linked to it.

During the fourth phase the vertices are linked again from level to level, starting from the highest, the end-product level.

The maximal structure generated this way contains all the combinatorially possible structures and all of its elements fulfil the 5 axioms.

3.5 The Generation of the Solution Structure, the SSG Algorithm

The maximal structure generated by the MSG algorithm contains all such combinatorically possible network structures that are able to produce the end-product from the given raw-materials. Consequently, it contains the optimal network as well. In most cases the optimalisation means to find the most cost effective solution.

The application of the SSG (Solution Structure Generation) algorithm enables the production of all the solution structures. The SSG is a new mathematical tool based on the application of the Decision-mapping (DM) which has been developed by Friedler et al. [20].

On (***M,O***) P-graph let Δ denote a mapping between the ***M*** set of materials various groups of the (α , β) operating units in $\wp$ (***O***) power set including the empty set as well. The Δ mapping can be defined, which determines for $X \in M$ material the set of operation units producing ***X***. The above defined mapping is Δ (***X***) which is a set itself:

$$\Delta(X) = \{(\alpha,\beta) \mid (\alpha,\beta) \in O \text{ és } X \in \beta\} \tag{3.8}$$

Let us suppose that ***m*** is a subset of ***M*** and ***X*** is an element of ***m***, as well as δ (***X***) is one of the subsets of Δ (***X***) then a δ [***m***] decision mapping can be defined for ***m*** with the help of ***m*** and δ (***X***):

$$\delta[m] = \{(X, \delta(X)) \mid X \in m\} \tag{3.9}$$

The δ [***m***] decision mapping given for ***m*** determines the relation between ***m*** set and the subsets of ***O*** set, since the element of the set ***m*** i.e. the materials are generated as the various combinations of the operating units. Each element of the subsets of δ [***m***] that is the δ_i[***m***]s pair the given ***X*** materials of the set ***m*** with the groups of operating units producing the given material, that is with δ (***X***).

The complement of the decision-mapping $\bar{\delta}$ [***m***] is defined that in case of any $X \in m$, $\bar{\delta}$ (***X***)= Δ (***X***)\ δ (***X***), i.e. if δ (***X***) is the set of operating units producing ***X*** material then $\bar{\delta}$ (***X***) is the set of operating units, which produce ***X*** material but are not elements of δ (***X***), that is $\delta(X) \cup \bar{\delta}(X) = \Delta(X)$. The definition of the complement of the decision-mapping in mathematical terms is the following:

$$\bar{\delta}[m] = \{(X, Y) \mid X \in m \text{ és } Y = \Delta(X) \setminus \delta(X)\} \tag{3.10}$$

As the first step in the process representation made with decision-mapping an active set must defined. Let us suppose that ***m'*** is an active set, and a subset of the

m material set of the (***m***, ***o***) P-graph. This ***m'*** subset of the (***m***, ***o***) P-graph is an active set if and only if at least one element of β output materials of each (α, β) operating unit is element of ***m'***, i.e. $m' \cap \beta \neq \emptyset$. On the basis of this, the ***m'*** active set will definitely contain with certain restrictions one of the optimal solutions including all the operating units.

With the help and application of the algorithm generating the maximal structure and the decision-mapping given for P-graph, the steps of the realisation of the SSG algorithm can be defined. This procedure generates all the solution structures, i.e. produces the combinatorially possible solution structures.

In the input phase of the SSG algorithm the sets needed for the generation, the products in the ***P*** set, the raw-materials in the ***R*** set, and all the other materials (intermediate-materials and by-products) not defined in ***M***, ***P*** and ***R*** are defined. The restriction given earlier must be true for the input sets, that is $R \subset M$, $P \subset M$ and $P \cap R \neq \emptyset$. A $\Delta[M]$ set must also be defined in the input phase, which contains the ***X*** materials and the $\Delta[X]$ mappings.

After the input phase the computer algorithm selects systematically and combinatorially the m active sets in a recursive way and executes the [***m***] decision mapping on them. The algorithm works until the selection and execution of all the ***m*** active sets is done.

4 P-Graph Extension for Workflow Modelling

A workflow model can be considered as a network structure. As it was presented in the previous chapter, the so far described solutions have been based on graph modelling. A simple graph, however, does not contain enough information so as to give the model precisely. The Petri-Net is also a directed bigraph which, with a token supplement, enables modelling of demand of resources crucial for workflow management and/or of eventual information or of raw material.

The Process-graph or P-Graph has been implemented in the fields dealing with the network-oriented synthesis of process networks (PNS) [15, 16, 17, 18, 19]. The P-graph is also a directed bigraph, the vertices of which are of two types. One vertex type is operation unit type vertex, while the other one is the material type vertex.

The ***O*** operating units in the P-graph can be interpreted in the same way, thus they can be assigned regarding their functions and handling to the activities in the workflow. In the P-graphs the ***M*** materials are basically modelled as documents or/and document processes in the workflow. During the process a document behaves similarly to the materials, however, there are slight differences which can be assigned to materials but not to documents. This means smaller restrictions which, however, do not reduce the representation significance of the modelling.

Therefore, similarly to the previously written about P-graph based workflow model the following documents can be introduced deducted from the materials:

Raw-materials	⟶	input documents
Intermediate-materials	⟶	intermediate documents
By-product	⟶	----- (cannot be interpreted)
End-product	⟶	output documents

By-products cannot be interpreted in the workflow, since the processing of the documents is targeted, and it is not influenced technologically, namely no redundant documents are prepared on purpose.

The separation of the intermediate documents from the output documents is not unambiguous, since documents which seem to be intermediate documents in one phase will be an output documents in others. Therefore, such databases can be regarded as real intermediate documents, which are prepared during the workflow process and will become information sources in later phases.

In addition to the condition above, the axioms, (S1 ... S5) used during the traditional application of the P-graphs and presented in the third chapter, can also apply and be interpreted in the case of workflow modelling. The relations (3.1 ... 3.10) presented in the algorithms (MSG, SSG) used for generating PNS can also be interpreted. Consequently, the biggest advantage of the P-graph-based workflow modelling is that the PNS method introduced above can be applied for the determination of the optimal network.

A lot of problems are already solved with Petri-Net modelling. Model controlling problems have not been mentioned at all, and routing has not been dealt with at all. The advantage of the P-graph is that there is a sound mathematical background, the P-graph can be presented by set theory tools; a well elaborated methodology for the optimal synthesis of the network structure can be found in the [15], [16], [17] literature, which apparently have similar advantages in case of workflow modelling as in several other applications [18], [19].

Conclusions

Next to the already existing workflow modelling solutions, it is justified to search new solution methods. In parallel to the Petri-Net-based workflow modelling that has an extensive literature and is widely used in the profession, P-graph-based workflow modelling can take ground and prove to be a useful way of workflow modelling. As a future step, the task is to investigate further application fields of the P-graph based workflow modelling.

References

[1] The Workflow Management Coalition Specification, Workflow Management Coalition Terminology & Glossary; Document Number WFMC-TC-1011; Document Status - Issue 3.0; February 1999

[2] S. Jablonski, C. Bussler: Workflow Management: Modelling Concepts, Architecture, and Implementation. International Thomson Computer Press, 1996

[3] Gy. Hermann: Distributed Computer System for Gauge Calibration, in Proceedings of 4th Serbian-Hungarian Joint Symposium on Intelligent Systems, SISY 2006, Subotica, Serbia, September 29-30, 2006, pp. 349-358

[4] Gy. Hermann: The Design of a Submicron Precision Coordinate Measuring Machine, in Proceedings of 3rd Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence, SAMI 2005, Herlany, Slovakia, January 21-22, 2005, pp. 397-408

[5] W. M. P. van der Aalst, K. M van Hee: Workflow Management – Models, Methods, and Systems, The MIT Press, Cambridge, Massachusetts, London, England, 2002

[6] S. A. P. White: Process Modelling Notations and Workflow Patterns, [Online] http://www.omg.org /bp-corner/bp-files/Process_Modelling _Notations.pdf

[7] R. Eshuis, R. Wieringa: Verification Support for Workflow Design with UML Activity Graphs, in the Proceedings of the 24th International Conference on Software Engineering Orlando, Florida , 19-25, May, 2002, pp. 166-176

[8] J. Tick: Workflow Model Representation Concepts, in Proceedings of 7th International Symposium of Hungarian Researchers on Computational Intelligence, HUCI 2006, Budapest, Hungary, November 24-25, 2006, pp. 329-337, ISBN 963 7154 54X

[9] P. Hruby: Specification of Workflow Management Systems with UML. In the Proceedings of the 1998 OOPSLA Workshop on Implementation and Application of Object-oriented Workflow Management Systems, Vancouver, BC 1998

[10] T. Kövér, D. Vígh, Z. Vámossy: Improved Face Recognition in the MYRA System, in Proceedings 4th Serbian-Hungarian Joint Symposium on Intelligent Systems, Subotica, Serbia, September 29-30, 2006, pp. 187-195, ISBN 963 7154 50 7

[11] Sz. Sergyán, L. Csink: Consistency Check of Image Databases, in Proceedings 2nd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence, Timisoara, Romania, May 12-14, 2005, pp. 201-206

[12] L. Horváth, I. J. Rudas: Evaluation of Petri Net Process Model Representation as a Tool of Virtual Manufacturing, in Proceedings SMC'98 Conference Proceedings, 1998 IEEE International Conference on Systems,

Man, and Cybernetics, October 11-14, 1998, ISBN: 0-7803-4778-1, pp. 178-183

[13] L. Horváth, I. J. Rudas: Modelling of Manufacturing Processes Using Object-oriented Extended Petri Nets and Advanced Knowledge Representations, in Proceedings IEEE International Conference on Systems, Man, and Cybernetics, Vancouver, BC, Canada, October 22-25, 1995, ISBN 0-7803-4778-1, Vol. 3, pp. 2576-2581

[14] W. M. P. van der Aalst: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers, Vol. 8(1), pp. 21-66, 1998, ISSN: 0218-1266

[15] F. Friedler, K. Tarjan, Y. W. Huang, L. T. Fan: Combinatorial Algorithms for Process Synthesis, Computers Chem. Engng, Vol. 16, pp. 313-320 (1992)

[16] F. Friedler, K. Tarjan, Y. W. Huang, L. T. Fan: Graph-Theoretic Approach to Process Synthesis: Axioms and Theorems, Chem. Engng Sci., Vol. 47, pp. 1973-1988 (1992)

[17] F. Friedler, L. T. Fan, B. Imreh, Process Network Synthesis: Problem Definition, Networks, Vol. 28, pp. 119-124 (1998)

[18] J. Varga: A folyamat-hálózatszintézis feladat kiterjesztései, PhD értekezés, Veszprémi Egyetem, Mérnöki Kar, Veszprém, 2000.

[19] B. Bertók: Folyamathálózatok struktúráinak algoritmikus szintézise, PhD értekezés, Veszprémi Egyetem, Műszaki Informatikai Kar, Veszprém, 2003

[20] F. Friedler, J. B. Varga, L. T. Fan: Decision Mapping: A Tool for Consistent and Complete Decisions in Process Synthesis, Chemical Eng. Sci. Vol. 50, pp. 1755-1768, 1995

Formalizing the Evaluation of OCL Constraints

Gergely Mezei, Tihamér Levendovszky, Hassan Charaf

Department of Automation and Applied Informatics, Budapest University of Technology and Economics
Goldmann György tér 3, H-1111 Budapest, Hungary
{gmezei, tihamer, hassan}@aut.bme.hu

Abstract: Domain-specific modeling has growing importance in many fields of software enginering, such as modeling control flows of data processing, or in man-machine systems. Customizable language dictionary and customizable notations of the model elements offered by domain-specific technologies make software systems easier to create and maintain. However, visual model definitions have a tendency to be incomplete, or imprecise; the definitions can be extended by textual constraints attached to the model items. Textual constraints can eliminate the incompleteness stemming from the limitations of the structural definition as well. The Object Constraint Language (OCL) is one of the most popular constraint languages in the field of UML and Domain Specific Modeling Languages. OCL is a flexible, yet formal language with a mathematical background. Existing formalisms of OCL does not describe dynamic behavior of constraints. Our research aims at creating an OCL optimization solution and prove its correctness formally. However, the shortcomings of the existing formalism has led us to create a new formalism. The paper presents OCLASM, a new formalism for OCL, which can describe both the semantics and the dynamical behavior of the language constructs, thus, it is capable of describing proofs of optimization algortihms. OCLASM is based on the Abstract State Machines technique.

Keywords: OCL, optimization, constraints, Abstract State Machines

1 Introduction and Motivation

Visual languages can accelerate the develepement and maintenance of software systems. Moreover, the ability to illustrate the structure of a system graphically means that even non-programmer users can understand the underlying logic. One family of visual languages are the Domain-Specific Modeling Languages (DSMLs), which allow creating visual models using a high level of abstraction, the customization of model rules and notation. The customization abilities of DSMLs makes easier to understand and handle the problems that made DSMLs very popular in almost all fields of software engineering including, but not limited

to general software modeling, feature modeling [1], resource editing [2] or control flow modeling.

Besides the advantages of visual languages, they have weaknesses as well. Visual model definitions have the tendency to be imprecise, incomplete, and sometimes even inconsistent. For example, assume a domain describing the cooperation between computer networks and humans in a man-machine system. A computer can have input and output connections in the network, but these connections use the same cable with maximum n channels. Thus, the number of the maximum available output connections equals the total number of channels minus the current number of input channels. It is hard, or even impossible to express this relation in a visual way. The solution to the problem is to extend the visual definitions by textual constraints. There exist several textual constraint languages, the Object Constraint Language (OCL) is possible the most popular among them. OCL was originally developed to create precise UML diagrams [3] only, but the flexibility of the language made possible to reuse OCL in language engineering, such as in metamodeling [4]. Nowadays, OCL is one of the most wide-spread approaches in the field of metamodeling and model transformations. The textual constraint definitions of OCL are unambiguous and still easy to use.

There exists several commerical and non-commerical domain-specific modeling tools, which have support for OCL either by interpreters, or by compilers. Interpreters are easier to implement, but they are not as flexible and efficient as compilers. The key of efficient constraint handling is to use optimizing OCL compilers. To our knowledge, currently none of the existing tools support optimization.

Our research focuses on creating a complete, system-independent optimizing constraint compiler. We have created three optimizing algorithms (presented in [5] and in [6]) that can accelerate the validation process by relocating, decomposing the constraint expressions and by caching the model queries. We have implemented these algorithms in our tool Visual Modeling and Transformaion Tool [7]. Besides the pseudo code of the algorithms, preliminary proofs of correctness were also presented in the mentioned papers. However, when implementing the algorithms, we have found that formal proofs are required to ensure the correctnes of the compiler.

OCL has a mathematical definition based on set theory with a notion of object model and system states. Although this formalism defines the syntax and semantics of OCL constraints, it does not cover dynamic behavior of the constraints. This paper presents OCLASM, a new formalism of OCL based on Abstract State Machines [8]. Our aim is to use this formalism to describe the OCL language and to prove the correctness of our optimization algorithms. The paper contains the formalism of one of the optimization algorithms, the *RelocateConstraint* algorithm to show how the formalism can be used in practice.

The paper is organized as follows: Section 2 explains why we have decided to create a new formalism for OCL instead of extending the existing. The section elaborates the most important projects in the field of OCL formalism and Abstract State Machines as well. Section 3 presents the basics of Abstract State Machines. Section 4 introduces the new formalism technique including the construction of the formalism and several basic examples. Section 5 shows how OCLASM can be used in case of Relocateonstraint algorithm. Finally, we summarize the presented work in section *Conclusions*.

2 Related Work

2.1 Set Theory and ASMs

The first question to answer is why we have decided to create a new formalism for OCL instead of using the existing one. Existing formalism does not define the dynamic behavior of constraints, however set theory is a highly flexible formalism technique, thus the formalism could be extended to support dynamic behavior. To show the differences between such an extension and OCLASM, we have to introduce the properties of ASMs.

Abstract State Machines (ASMs) are formerly known as *evolving algebras*. ASM is working on abstract data structures, which are provided with a simple mathematical foundation. The notation of ASM is based on the mathematically precise notion of a virtual machine execution, states and state transitions. This notation is familiar from programming practice. ASM provides a concise way to define system semantics and dynamic behavior. ASMs are very popular in the domain of formal specification.

The ASM formalism has several advantages in contrast with the extension of the original formalism in this field. Firstly, the notation of ASM is easier to use for proving the correctness of algorithms given by pseudo code. Secondly, modularization and stepwise refinement is easier to accomplish in ASM. This also means that the formalism specification of the dynamic behavior can be hierarchically decomposed. Set theory is a flexible technique, but it uses a low-level description of the problem space, thus, the description the dynamic behavior would produce a considerably huge rule set. In case of ASM this problem does not occur, because ASM allows to choose the level of abstraction used in the formalism. Therefore, the formalism in ASM can be more concise for our purposes with respect to OCL optimization.

2.2 OCL and ASM

Abstract State Machines were used in many projects as a mathematical formalism. This section introduces only a few of these projects, more precisely, the projects in connection with OCL or modeling, and projects from where our method has borrowed some basic ideas. Some of the mathematical model formalisms not based on ASM are also presented.

The book [9] presents a precise approach, which facilitates the analysis and validation of UML models and OCL constraints. It defines a formal syntax and semantics of OCL types, operations, expressions, invariants, and pre-postconditions, and it discusses some of the main problems with the original OCL specification. Although the book does not use ASM for formalism, it gives a precise overview about the topic.

The OCL formalism available in set theory is examined in [10]. The paper collects the elements appearing in OCL standard, but not in the formalism. It presents an extension of the original formalism to solve these problems.

In [11], an ASM definition for dynamic OCL semantics is presented. This formalism focuses on the states of the modeling environment and handles the invariants as atomic units implemented in outer functions. This means that the formalism handles the effects and the result of the validation, but it does not give ASM definition for the OCL statements, such as `forall`, thus, it is not capable of describing algorithms operating with statements.

ASM definition for Java and Java Virtual Machine (JVM) is elaborated in [12]. This ASM formalism offers an implementation-independent description of the language and the execution environment. Using this abstract description, several properties of Java and JVM have been proved. The book contains several straightforward solutions. Our approach has borrowed the basic idea of formalization, namely handling the code as an annotated syntax tree from here.

3 Backgrounds

3.1 ASM Basics

In [8], ASMs are introduced as follows. ASMs are finite sets of *transition rules* of the form

```
if (condition) then Updates
```

which transform abstract states. Where *Condition* (referred to as guard) under which a rule is applied is an arbitrary predicate logic formula without free variables. The formula of *Condition* evaluates to *true* or *false*. *Updates* denotes an infinite set of assignments in the form of $f(t_1..t_n) := t$ whose execution is understood as changing (or defining, if it has been not defined before) the value of the occurring function *f* at the given arguments.

The notion of *ASM states* is the classical notion of mathematical structures where data is provided as abstract objects, i.e., as elements of sets (domains, universes, one for each category of data) which are equipped with basic operations (partial functions) and predicates (attributes or relations). The notion of *ASM run* is the classical notion of computation in transition systems. An ASM computation step in a given state consists of executing simultaneously all updates of all transition rules whose guard is *true* in the state if these updates are *consistent*. A set of updates is called *consistent* if it contains no pair of updates with the same location.

Simultaneous execution provides of an ASM rule *R* for each *x* satisfying a given condition φ:

forall x with φ R,

where φ is a Boolean-valued expression and *R* is a rule. We freely use abbreviations, such as *where*, *let*, *if then else*, *case* and similar standard notations which are easily reducible to the above basic definitions.

A priori no restriction is imposed either on the abstraction level or on the complexity or on the means of the function definitions used to compute the arguments and the new value denoted by t_i, *t* in function updates. The major distinction made in this connection for a given ASM *M* is that between *static* functions which never change during any run of *M* and *dynamic* ones which typically do change as a consequence of updates by *M* or by the environment. The dynamic functions are further divided into four subclasses. *Controlled* functions are dynamic functions which can directly be updated by and only by the rules of *M*. *Monitored* functions are dynamic functions which can directly be updated by the environment only. *Interaction* or *shared* functions are dynamic functions which can directly updated by rules of *M* and by the environment. *Derived* functions are dynamic functions which cannot be directly updated either by *M* or by the environment, but are nevertheless dynamic, because they are defined in terms of static and dynamic functions.

3.2 The Mathematical Definition of ASM

In an ASM state, data is available as abstract elements of domains which are equipped with basic operations represented by functions. Relations are treated as Boolean-valued functions and view domains as characteristic functions, defined

over the superuniverse which represents the union of all domains. Thus, the states of ASMs are algebraic structures, also called algebras.

Definition 1 *A vocabulary (also called signature) Σ is a finite collection of function names. Each function name has an* arity*, which is a non-negative integer representing the number of arguments the function takes. Function names can be static or dynamic. Nullary function names are often called constants; but the interpretation of dynamic nullary functions can change from one state to the next. Every ASM vocabulary is assumed to contain the static constants undef*, True *and* False.

Definition 2 *A state $\mathfrak{A}$ of the vocabulary Σ is a non-empty set X, together with the interpretation of the function names of Σ, where X means the superuniverse of $\mathfrak{A}$. If* f *is an n-ary function name of Σ, then its interpretation $f^{\mathfrak{A}}$ is a function from X^n into X; if c is a constant of Σ, then its interpretation $c^{\mathfrak{A}}$ is an element of X. The superuniverse X of the state $\mathfrak{A}$ is denoted by $|\mathfrak{A}|$.*

The *elements* of the state are the elements of the superuniverse of the state and, according to the definition, the parameters of the functions are also elements of the superuniverse. The new elements come from *reserve*, which is a set whose role is to provide new elements whenever needed.

Definition 3 *An abstract state machine M consists of a vocabulary Σ, an initial state $\mathfrak{A}$ for Σ, a rule definition for each rule name, and a distinguished rule name called the main rule name of the machine.*

4 ASM for OCL

4.1 Overview

OCLASM has been created to formalize the evaluation of constraints, OCLASM acts as an *interpreter* for OCL constraints. The same functions and the same rule set are used regardless of the constraint or the underlying model. This property of OCLASM is essential, since the OCLASM has been created as a generic formalism for OCL.

States of OCLASM represent the state of execution at a certain point of time. A *state* can be considered as an internal state of the evaluating environment, which is evolved by the rule set of OCLASM. *States* describe for example which expression is under evaluation or which local variables are available. The rules of OCLASM are used to navigate between the *states* when running the validation.

It is important that the *state*s do not contain any particular information about the underlying model (the model to validate), since (i) OCL cannot change the underlying model by the definition of OCL [3], and (ii) the validation must be platform independent. Similarly, *states* do not describe the constraints directly, but the expressions of the constraints are obtained by a monitored function.

The presented approach is similar to the method published in [12] in several aspects, where the constraint expression is handled as the sequence of programming statements and expressions. The execution of the constraint is a step-by-step execution of these programming units. At each position, the corresponding expression or statement is evaluated or executed, and then the evaluation proceeds to the next programming unit. The method is also similar to traversing the annotated abstract syntax tree of the constraint.

4.2 Monitored and Shared Functions

Obtaining the model items and the phrases of constraint expressions is handled by monitored functions. This solution ensures that the modeling environment and the evaluation environment are independent from the dynamic behavior of the constraint formalized by the OCLASM.

The underlying model extends the original model structure definition used in OCL: it consists of model nodes, attributes and relationships (not restricted to UML types). Attributes are either of primitive types or of complex attributes containing several sub-attributes. Model nodes can contain attributes (both primitive and complex attributes). The attribute and modeling structure described in [13] is used, which can extend the UML-based modeling structure to an n-level metamodeling hierarchy. This generic structure allows extending OCL to domain-specific languages as well [4]. The extension of the underlying modeling structure means that OCLASM can be used not only to describe the dynamics of constraint evaluation, but to formalize constraints for domain-specific models as well.

Both model nodes and attributes are identified by a unique ID (a literal expression), the universe of IDs is shared between the two different constructs. This uniformity helps reducing the number of monitored functions (for example, node-node and node-attribute navigations can be handled uniformly). To differentiate attributes and normal model nodes, there is a *IsModelNode(ID)* function defined.

To simplify handling of attributes the function *AttrValue(ID)* is used. The function returns *undef* for complex attributes, but returns a primitive value for primitive attributes. Primitive value in this context means that the type of the value is Boolean, number or string.

To express the meta level- instance level relationship the *Meta(ID)* function is used, which returns the meta item of the instance level item identified by ID. This

function is essential to capture different instantiation techniques (for example the instantiation of UML [14] and VMTS [15]).

Navigation between model items is handled by two functions. The function *To(ID, Dest)* works on the model level, the ID identifies the source model item, while *Dest* selects target items. The function returns all nodes or attributes which can be reached from the model item using a relation where the destination name is *Dest*. The result of the function is a list of possible destinations. Note that the list contains either model item IDs, or attribute IDs, but not both of them, because OCL does not allow this kind of polymorphism (attribute queries and navigations could not be distinguished).

The *Mul(ID, Dest)* ('Mul' stands for Multiplicity) function is another function to handle relations between model items. It works on the metamodel level, thus, the model definition is checked instead of the concrete models. The function *Mul* checks the minimum and the maximum multiplicity of the given relation according to the metamodel. It returns four integers as a list, the minimum/maximum multiplicity of the source/destination side.

OCLASM uses a shared function *GetPhrase(Position)* as well. *Phrase*s are basic syntactic constructs (programming statements and expressions) available in OCL. A *Phrase* has a string attribute *PhraseType* (e.g. 'NavigationCall'). *Phrase*s can contain other *Phrase*s as children. For example, an iteration *Phrase* can have an iterator variable declaration *Phrase*, an iteration condition *Phrase* and an iteration core block *Phrase*. The function *GetPhrase* returns a *Phrase* of the constraint identified by the parameter Position. Position is a value from the universe of all possible positions of *Phrase*s of the constraints (the universe is referred to as *Pos*). If the constraint is handled as a syntax tree then *Phrase*s are the nodes of this tree and the universe of *Pos* contains the pointers to the nodes. Note that the function is marked as shared, which means that it can be updated by the environment, or by the rules of OCLASM. This duality is required, because in general, constraints are defined outside the scope of OCLASM, but certain algorithms can modify the original constraints. For example an optimization algorithm can restructure the expressions of constraints in order to improve the performance.

Child(Position, I) is another shared function. It obtains the *I*th children *Phrase* of a complex *Phrase* (a *Phrase* which has children). The first parameter of the function is the (valid) position of the complex *Phrase*. If the *Phrase* does not have a child at the selected index, then the function returns *undef*. The function *Parent(Position)* implements the reverse direction: it obtains the position of a *Phrase* and returns the position of the container *Phrase*. These functions are shared functions for the same reason as *GetPhrase*. Note that *Child* and *Parent* functions are always synchronized automatically by the framework.

4.3 Dynamic Functions

Constructs of OCL, for example iterate or navigate, are mainly defined as rules in OCLASM. Dynamic functions help to store the current state of the evaluation environment when the rules are applied. For example, navigate operation selects a new model item, which is used as the origin of all further operations. Dynamic functions are just like helper variables in the environment framework.

To obtain the current position of evaluation, the position of the *Phrase* currently under execution, the function *CurrentPos()* is used. The return value of the function is a position from the universe *Pos*.

The dynamic function *Type(Pos)* is used to handle the type of the different (OCL) expressions uniformly. Its parameter is the position of the target expression. The return value of the function can be one of the basic types defined in OCL, such as real, integer, or tuple.

The value of the expressions are handled similarly to the type function: the unary function *Value(Pos)* retrieves the position of the expression and returns its value. Using the notation of common programming languages, such as C, the difference between *GetPhrase*(pos) and *Value(pos)* is the following: *GetPhrase*(pos) is similar to a pointer. In contrast, *Value(pos)* is the value in the pointed memory block.

OCL allows the user defining local variables. In OCLASM, these variables are handled by the function *Local(Name)*. The function has one input parameter: the name of the variable. The function *Local* returns the position of a variable declaration expression. When defining a new local variable, then a new position is created using the *reserve*. Variable declarations contain the name, type and value of the variable. If a local variable is requested by its name and there is no local variable defined with the given name, then the function returns *undef*. The name of the local variables are handled by the unary function *Name*, which has one input parameter, the position of the variable expression. *Name* returns the name of the local variable as a string, or undefined if the position is not a valid variable definition.

OCLASM handles all four types of collections *(Set, OrderedSet, Bag* and *Sequence)* by arrays indexed by integer numbers. Indexing is denoted by brackets. The items in the arrays are the items in the collections, for example, *Value(Position)[3]* means the third item in the collection: expression at the position *Position*. The arrays can be traversed by the *forall* expression of ASM, obtaining every element. According to the default notation of ASM [8] the length of collection is denoted as *l(Value(Position))*, where *Position* is the position of the expression.

Tuple types are also handled as arrays indexed by integer values, the definition of tuple items are stored in lists with two elements (name – value pairs), these lists

are list items of the array representing the tuple. For example, the expression `Tuple(x: Integer = 5, y: String = 'Ok')` results an array with two items: `TupleArray[1] = ['x', Integer = 5]`, while `TupleArray[2] = ['y', String = 'Ok']`. When a tuple item is queried by its name, then OCLASM tries to find an item in the associated array with the name and updates the *Type – Value* functions. In the previous example if the tuple item with name 'y' is requested, then OCLASM checks `TupleArray[1]`, but its name ('x') does not match, thus, it advances to `TupleArray[2]`. Since the name is found, OCLASM sets the *Type* of the current position to 'String' and the *Value* to 'Ok'.

4.4 Vocabulary and Universes

Using the previously defined functions, the syntax of OCLASM can be defined:

Definition 4 *The vocabulary Σ_{OCLASM} of the OCLASM formalism is assumed to contain the following characteristic functions (arities are denoted by dashes):*

- *Monitored functions: IsModelItem/1, AttrValue/1, Meta/1, To/2, Mul/2*
- *Shared functions: GetPhrase/1, Child/2, Parent/1*
- *Dynamic functions: CurrentPos/0, Type/1, Value/1, Name/1, Local/1*

Definition 5 *The superuniverse $|\mathfrak{A}|$ of a state $\mathfrak{A}$ of Σ_{OCLASM} is the union of six universes:*

- *The universe of Phrases (basic syntactic constructs of OCL)*
- *The universe of possible positions of Phrases in the constraints*
- *The universe Boolean (true/false/undef)*
- *The universe of finite lists of numbers*
- *The universe of finite lists of finite strings*
- *The universe of finite lists of possible identifiers (IDs) for model items and attributes*

4.5 Transition Rules

Transition rules describe how the states of OCLASM change over time by evaluating expressions and executing statements of the input program. The main idea is to create a rule for each type of language constructs available in OCL. For example OCLASM has rules for iterate, navigation, or variable declaration actions. These rules describe the semantics of the expression and manage dynamic functions.

OCLASM a central rule called *eval*. The rule *eval* is a rather complex rule, it has a switch-case block with many branches, more precisely for each type of language constructs, *eval* has a branch (see sketch of the rule below). It checks the type of the parameter *Phrase* and executes the appropriate branch by calling the rule associated with the *PhraseType*. Therefore, *eval* acts as a mapping function between *Phrase*s and rules of OCLASM. Moreover, *eval* helps calling the rules with the appropriate parameters (it adds sub-expressions as parameters, using the *Child* function). Updating the value of *CurrentPos* is also handled by *eval*.

```
1.  rule eval(Phrase)
2.  {
3.    CurrentPos():=Phrase;
4.    switch(PhraseType(Phrase))
5.    {
6.      case 'VariableDeclaration':
7.        VariableDeclaration(Child(Phrase,1), Child(Phrase,2),
8.                            Child(Phrase,3));
9.      case 'NavigationCall':
10.       Navigate(Child(Phrase,1), Child(Phrase,2);
11.     …
12.   }
13. }
```

The initial position of OCLASM sets the *CurrentPos* to the start position of the outermost constraint expression and sets the value of all other dynamic functions to *undef* for all possible parameters. A run of OCLASM is started by calling *eval* for the start position. When the run of the state machine of OCLASM is finished then the outermost expression holds a single value. If the evaluated constraint was an invariant, then this value shows whether the model was valid.

4.6 Invariants

OCLASM, as presented until this point is useful to simulate the execution of a simple or complex OCL expression, but not a whole constraint, such as an invariant. This definition is the core of OCLASM, but the presented approach can be extended in order to support OCL invariants or pre and post conditions. The current description shows how invariants can be formalized.

```
1.      rule CheckModelInvariants(Invariants)
2.      {
3.        forall(Invariant in Invariant)
4.        {
5.          forall (ID in {∀ID: ModelItem(ID)= true})
6.          {
7.            if (Invariant.Context) = Meta(ID))
8.            {
9.              CurrentPos():=Invariant.StartPosition;
10.             Local():=undef;
11.             if (not eval(CurrentPos())
12.             {
13.               print "The model is not valid.";
```

```
14.             exit;
15.           }
16.         }
17.       }
18.     }
19.   }
```

To have this rule as part of OCLASM, the universe of *Invariants* must be added to the superuniverse in Defintion 5. *Invariants* have a *Context* property and a pointer to the outermost expression in the invariant. The extended OCLASM presented in this section is referred to as $OCLASM_{Inv}$ to differentiate from the original OCLASM definition. Pre and post conditions can be handled similarly resulting a family of OCLASM formalisms.

4.7 Examples

Although the OCLASM formalism presented in this paper is capable of describing all OCL operations, we present only a few rules showing the method in practice. Other operations can be formalized similarly.

Firstly, a very simple rule, the *VariableDeclaration* is shown. The function *eval* obtains the position of the children expressions (variable name, type and init value) and executes them before this rule is executed.

```
1.    rule VariableDeclaration(VarName, VarType, VarInit)
2.    {
3.     Name(CurrentPos)  = Value(VarName)
4.     Type(CurrentPos)  = eval(VarType)
5.     if (VarInit!= undef)
6.        Value(CurrentPos) = eval(VarInit)
7.     else
8.        Value(CurrentPos) = undef
9.     endif
10.   }
```

Secondly, the rule for *iterate* operations is presented. It is essential to formalize this operation, because every other collection operation can be accomplished by using *iterate [1]*. For example, the collection operation *count()* can be simulated by an iterate expression

```
iterate(i : Integer, r Integer = 0 | r+1).
```

The node *iterate* has exactly four children in the abstract syntax tree: (i) the collection where the operation is applied; (ii) the declaration of the iterate variable, (iii) the declaration of the result variable, and (iv) the iteration body.

```
1.  rule iterate(CollectionDef, IteratorDecl, ResultDecl, Iteration)
2.  {
3.     eval(CollectionDef);
4.     eval(IteratorDecl);
5.     eval(ResultDecl);
```

```
6.      Local(Name(ResultDecl)):= new(Pos);
7.      Value(Local(Name(ResultDecl))):= Value(ResultDecl);
8.      Type(Local(Name(ResultDecl))):= Type(ResultDecl);

9.      forall collectionElement in Value(CollectionDef)
10.        Local(Name(IteratorDecl)):= new(Pos);
11.        Value(Local(Name(IteratorDecl)))= collectionElement
12.        Type(Local(Name(IteratorDecl))) = Type(IteratorDecl)
13.        eval(Iteration);
14.     endfor

15.     Local(Name(IteratorDecl))= undef;
16.     Value(CurrentPos) = Value(Local(Name(ResultDecl)));
17.     Type(CurrentPos)  = Type(Local(Name(ResultDecl)));

18.     Local(Name(ResultDecl))= undef;
19.  }
```

The third example shows the rule constructed for navigation expressions. Here the model-based, external functions are also used. The rule evaluates the origin, namely, it obtains the model item which is the starting point of the navigation. As next, the rule checks the multiplicity of the rule, if it allows exactly one connection, then the result is a *ModelItem*, in any other case the result is a collection of *ModelItems*. Since the function *To* always returns a list (with the IDs of the destination nodes), in the first case the first element of the result array is used (`To(Value(Origin), DestName)[0]`). In this case the type of the result is the ID of the meta node of the destinations node. If the multiplicity is not 1, then a new collection is created and returned.

```
1.   rule Navigate(Origin, DestName)
2.   {
3.     eval(Origin);
4.     if ( Mul(Value(Origin),DestName)[2]==1 and
5.          Mul(Value(Origin),DestName)[3]==1 and
6.          IsModelNode(To(Value(Origin),DestName)[0]))
7.     {
8.        Value(CurrentPos)= To(Value(Origin),DestName)[0]
9.        Type(CurrentPos) = 'ModelItem'
10.    }
11.    else
12.    {
13.       Type(CurrentPos) = 'Set'
14.       Value(CurrentPos) = Set()
15.       forall ModelId in To(Value(Origin),DestName)
16.         Append ModelId in Value(CurrentPos)
17.       endfor
18.    }
19.  }
```

5 Application of the Formalism

5.1 The Relocation Algorithm

This section introduces how OCLASM can be used to prove the correctness of a dynamic algorithm working on OCL constraints. The *RelocateConstraint* algorithm is an optimization algorithm for OCL, it has been presented in [5] and in [6]. The main idea behind the algorithm is to reduce the time-consuming model queries, if it is possible. Therefore, the algorithm tries to relocate OCL invariants defined by the user to another context, where the evaluation requires the least model queries. Using OCLASM, it is possible to give a formal proof of correctness for the algorithm.

The algorithm applies the relocation always between adjacent nodes. In other words, the original and the new context of the constraint are always connected in the metamodel. This limitation does not restrict the optimization capabilities of the algorithm (as shown in [6]). Note that this does not mean that the nodes of the original context and the new context are connected *on the instance level* as well. This is because checking invariants is based on meta level (see algorithm X in section Y) and metamodels can allow zero multiplicity between the elements. Our research has shown [6] that the correctness of relocation is heavily affected by the multiplicities on the source and on the destination side.

The algorithm is defined as a rule, which runs before *eval* is called with the outermost position of the invariant. The algorithm modifies the constraints by updating the *GetPhrase* and *Child* functions. To simplify the rule it is worth defining formulas and helper rules:

```
φA.Dest(Phrase, IDA, Dest)  =  Value(eval(Child(Phrase,0))) = IDA and
                               Value(Child(Phrase,1)) = Dest

φA.x (Phrase, IDA, Dest)  =  Value(eval(Child(Phrase,0))) = IDA and
                             Value(Child(Phrase,1)) <> Dest
```

If the original context is A and the new context is B then $\varphi_{A.Dest}$ expresses that the *Phrase* is a navigation from A to B, while $\varphi_{A.X}$ expresses that the *Phrase* is a navigation from A to anywhere, but not to B. The formulas use the information that the first (0^{th}) child of navigations (to model nodes and attributes) is the origin, while the second (1^{st}) child is the name of the destination.

```
ForallCheck():=  ∃ModelPhrase:  GetPhrase(ModelPhrase)<>  undef  and
                  (φA.x(ModelPhrase,0,D)  or  (φA.Dest(ModelPhrase,0,D)
                 and Parent(ModelPhrase).PhraseType<>'ForAll')  )))
```

This derived helper function returns true, if (i) there is navigation/attribute call from A to x (x<>B), or (ii) there is a navigation from A to B and the outermost phrase in the constraint has a specific type ('ForAll').

```
GetSrc(ID, Dest) := { Src: if ∃ID2: ID2 ∈ To(ID,Dest) and
                                ID ∈ To(ID2, Src)
                      undef, otherwise
```

This derived helper function obtains the name of the source side of navigations. For example there is a navigation between A and B, we can navigate from A to B using the destination name "B". This function obtains the reverse direction, namely to source name, which is used in navigation from B to A.

```
Top(Phrase)          := Phrase, if Parent(Phrase)=undef
                        PhraseTOP, if ∃PhraseTOP :
                                   PhraseTOP= Top(Parent(Phrase))
```

This derived function tries to find the outermost *Phrase* of the constraint by using a recursive method.

```
1.  rule AddChild (Type,Parent,Idx)
2.  {
3.    Child(Parent, Idx):= new(Pos);
4.    GetPhrase(Child(Parent, Idx)):= new (Phrase);
5.    GetPhrase(Child(Parent, Idx)).PhraseType:= Type;
6.  }

1.  rule AddBackNavigation(Phrase, WithForallCheck)
2.  {
3.    if (WithForallCheck and (ForallCheck() and Mul(O,D)[1]>1))
4.    {
5.         AddChild('VariableCall',Phrase,0);
6.         AddChild('OrigSelf', Child(Phrase,0), 0);
7.    }
8.    else
9.    {
10.        AddChild ('NavigationCall', Phrase,0);
11.        AddChild ('SelfReference',Child(Phrase,0),0);
12.        AddChild ( GetSrc(O,D),Child(Phrase,0),1);
13.   }
14. }
```

These helper rules are used to automate the repeatable parts of the main relocation rule. The first rule creates a new *Phrase* with the given type and adds it as a child of the parameter *Phrase*. This rule is used when the algorithm inserts new expressions into the constraint. The rule *AddBackNavigation* inserts a complete set of *Phrase*s as a child of the parameter *Phrase*. The rule has two different modes according to the second parameter, the result of *ForallCheck* function and the multiplicity on the source side. The first mode creates phrases that call a variable with the name 'OrigSelf', while the second mode inserts phrases implementing a navigation call from the new context to the original context. The reason while the modes are differentiated is explained later.

```
1.  rule RelocateConstraint(O, D)

2.  if (Mul(O,D)[2]=0)
3.  {
4.    exit ("Relocation error.")
5.  }

6.  if (Mul(O,D)[0]=0)
7.  {
8.    AddChild ('IfExpr', undef ,0);
9.    AddChild ('IsEmpty', Child(undef,0) ,0);
10.   AddBackNavigation(Child(Child(undef,0), 0), false);
11.   Child(Child(undef,0), 1):= Top(ModelPhrase);
12.   Child(Child(undef,0), 2):= True;
13. }

14. if (Mul(O,D)[1]>1 and ForAllCheck())
15. {
16.   AddChild ('ForAll', undef ,0);
17.   AddBackNavigation(Child(undef,0), false);
18.   AddChild ('IteratorVariable', undef ,1);
19.   Value(Child(undef,1)):='OrigSelf';
20.   Child(Child(undef,0), 2):= Top(ModelPhrase);
21. }

22. forall (ModelPhrase in {∀Phrase: GetPhrase(Phrase)<> undef and
                        (Phrase.PhraseType='AttributeCall' or
                        Phrase.PhraseType = 'NavigationCall')})
23. {
24.     if (φA.Dest(ModelPhrase,0,D))
25.     {
26.        if (Mul(O,D)[3]>1)
27.        {
28.           if (Parent(ModelPhrase).PhraseType='ForAll')
29.           {
30.              GetPhrase(Parent(ModelPhrase)):=
31.              GetPhrase(Child(Parent(ModelPhrase),2));
32.           }
33.           else
34.              AddBackNavigation(ModelPhrase, true);
35.        }
36.        else
37.           ModelPhrase:= Child(ModelPhrase, 0);
38.     }
39.     else
40.        AddBackNavigation(ModelPhrase, true);
41. }
42. endrule
```

The rule is relatively complex; different cases are summarized in Table 1. The table shows the different cases based on the multiplicities on the source side (rows of the table) and on the destination side (columns of the table). The table summarizes the special constructs used in different cases, each cell has the corresponding line numbers for the constructs in the rule. The basic relocation case can be found at line 37 and 40 in the rule. Recall that the explanation why we distinguish these cases can be found in [6].

	<1	**1**	**>1**
<1	Not allowed (#2-5)	Add Filter (#6-13)	Add Filter, ForAll check (#6-13; #28-34)
1	Not allowed (#2-5)	No spec.	ForAll check (#28-34)
>1	Not allowed (#2-5)	Add ForAll (#14-21)	Add ForAll, ForAll check (#14-21; #28-34)

Table 1
Relocation overview

5.2 Proof of Correctness

The algorithm applied by the *RelocateConstraint* rule is correct, if none of the constructs (modification of the constraint) modifies the result of validation, i.e. if the set of valid models remains the same. Therefore, the presented proof shows that the steps of the rule are correct by mapping expressions of the original constraint to expressions in the new constraint and show that they are equivalent. The proof uses the *CheckModel* rule from section 4.6. For sake of simplicity, the condition expression at line #11 in the algorithm is referred to as 'ValidModelCheck'. We refer to sections of the rules as Rule(#From-To), for example RelocateConstraint(#2-5). Note that the aim of the proving method is to show the equivalency of the original and the relocated constraint, therefore, it is not stressed whether the rule RelocateConstraint is optimal.

To distinguish the original and the relocated constraints we use the labels 'old' and 'new', for example C_{old} (that is the original constraint), or $self_{old}$ (self reference of the original constraint) Furthermore, for sake of simplicity we suppose that the original context type was A, while the new context type is B. The instantiations of type A are a_1, $a_{2 \ldots} a_n$ (collected in a set $Inst_A$), the instantiation of type *B* are b_1, $b_{2\ldots} b_n$ (collected in the set $Inst_B$). *B* can be reached from *A* using the name *Dest* (**To(a_i,Dest)=b_j**) while the reverse direction uses *Src* (**To(b_i,Dest)=a_j**).
If the model allows zero multiplicity, then one of the following formulas are true:

$\varphi_{ZeroOnSrc}$: **Mul(A,Dest)[0] = 0**
$\varphi_{ZeroOnDest}$: **Mul(A,Dest)[2] = 0**

We prove that the algorithm is always correct in these cases, then we formalize other possible cases as follows:

$\varphi_{ExactlyOne}$: **Mul(A,Dest)[1]=1 ∧ Mul(A,Dest)[3]=1**
$\varphi_{ManyOnSrc}$: **Mul(A,Dest)[1]>1 ∧ Mul(A,Dest)[3]=1**
$\varphi_{ManyOnDest}$: **Mul(A,Dest)[1]=1 ∧ Mul(A,Dest)[3]>1**
$\varphi_{ManyBoth}$: **Mul(A,Dest)[1]>1 ∧ Mul(A,Dest)[3]>1**

If the constraint does not contain any model query, then the constraints results in a simple Boolean value true, or false. The result does not depend on the underlying model, thus it is *always* true, or false.

If this constant result is true, then it means that all possible models are valid. This means that ValidModelCondition is always satisfied, thus, the inner *forall* expression at CheckModel(#5-17) can be replaced by a constant true value. Therefore C_{new} always evaluates to true as well.

If this constant result of the constraint is false (no model is valid, which contains a model item of the selected type), then $\text{ValidModelCondition}_{Old}$ is not satisfied for instances of *A*, while $\text{ValidModelCondition}_{New}$ is not satisfied for instances of *B*. Therefore, the relocation of the constraint is not correct if Inst_A, or Inst_B is empty.

Inst_B can be empty only if formula $\varphi_{\textbf{ZeroOnDest}}$ holds (zero multiplicity is allowed on the destination side). This case is handled at RelocateConstraint(#2-5): the condition checks whether for each a_i there exists at least one b_i, more generally that there is a b_i in the model, which can be used as a host for the constraint (host nodes of a constraint are nodes, in which the constraint is defined). If not, then the rule throws an error message.

Similarly, Inst_A can be empty if formula $\varphi_{\textbf{ZeroOnSrc}}$ holds. The problem is solved by the condition at RelocateConstraint(#6-13), which ensures that the constraint is evaluated only to those b_js which are connected to at least one a_i (otherwise it returns with a constant true). Thus, if Inst_A is empty, then the constraint is not evaluated.

As result of the conditions at RelocateConstraint(#2-13), the following formulas *always* hold:

$\varphi_{\textbf{AToB}}$: $\forall$i: Meta(i)=A $\rightarrow$ ($\exists$j, $\exists$Dest: Meta(j)=B $\wedge$ j$\in$To(i, Dest))

$\varphi_{\textbf{BToA}}$: $\forall$i: Meta(i)=B $\rightarrow$ ($\exists$j, $\exists$Src: Meta(j)=A $\wedge$ j$\in$To(i, Src))

Note that these formulas ensure also that the relocation of constraints without model queries is always correct.

If the constraint contains model queries, then the result of eval at CheckModel(#11) can be affected by the navigations and attributes of a_i. The relocation is correct if the result of model queries in the original context is the same as the result in the new context. Different cases are indexed by the formulas defined above:

$\varphi_{\textbf{ExactlyOne}}$: RelocateConstraint(#37) and RelocateConstraint(#40) are used. RelocateConstraint replaces navigations from A to B with a self reference. Instead of self_{Old}.Dest the new constraint has a self_{New} expression (RelocateConstraint(#40)). According to the rules of OCLASM, the value of self_{Old}.Dest is retrieved by the monitored function call To(a_i,"Dest"), which results b_j, an instance of B. Since formula φ_{AtoB} holds, thus this b_j always exists, therefore

the expression To(a_i,"Dest") can be replaced by the value b_j for this certain navigation expression for this certain a_i. However, the constraint is cheked against all a_is (CheckModel(#7)), thus To(a_i,"Dest") is always replacable by an appropriate b_j. In the relocated constraint this replacement is done by using the new self reference. Note that relocation does not create false host nodes (nodes which could not affect the resul of the original constraint, but could affect the result of the relocated version) for the constraint (due to $\varphi_{BtoA.}$).

RelocateConstraint inserts a navigation expression from B to A (RelocateConstraint(#37)) before any navigations/attribute queries from A to anywhere, but not B. The expression $self_{Old}$.x (where x is not "Dest") is replaced by $self_{New}$.Src.x. Proving is similar to the previous case, but on the reverse direction. Since the formula φ_{BtoA} holds, thus $self_{New}$.Src which results in a To(b_i, "Src") can be replaced by the value a_j for this certain navigation/attribute expression for this certain b_i. Moreover, the new constraint is cheked against all occurences of b_i (CheckModel(#7)), thus To(b_i,"Src") is always replacable by an appropriate a_j. Therefore, the value of $self_{Old}$ always equals with the value of $self_{New}$.Src. Note that relocation does not delete host nodes because of φ_{AtoB}.

$\varphi_{ManyOnDest}$: The section RelocateConstraint(#28-34) and RelocateConstraint(#40) are applied. Correctness of navigations of the form $self_{Old}$.x, where x <> "Dest" is ensured by RelocateConstraint(#40) according to constructs used in $\varphi_{ExactlyOne}$. However, the result of navigations from A to B results in a set of b_is. The condition at RuleConstraint(#28) checks whether the result of the navigation expression is used in a *forall* construct, if so, then the original expression is replaced by the inner expression of *forall*. Thus, $self_{Old}$.Dest $\rightarrow$ *forall*(Exp_{Old}) is transformed into Exp_{New}. In OCL, *forall*(Exp) is a set operation, which is true only if Exp is satisfied for each item in the set. Therefore, the original expression $self_{Old}$.Dest $\rightarrow$ *forall*(Exp) is true for a certain a_i, if Exp is true for each $b_j \in$ To(a_i, Dest). This means that the original expression in a certain a_i can be replaced by Exp_{New} in $b_j \in$To(a_i, Dest). The original constraint is evaluated for each a_i, thus the replacement is correct in each b_j connected to one of the a_is. Moreover, since φ_{BToA} holds, thus, b_js are always connected to an a_i. This means that the relocation is always correct in this case.

If the condition at RelocateConstraint(#28) is not satisfied, then replacement inserts a navigation back to the original constraint and the expression is evaluated there. The expression $self_{Old}$.Dest is replaced by $self_{New}$.Src.Dest. Because of φ_{BToA} this replacement is always correct as shown constructs used in $\varphi_{ExactlyOne}$.

$\varphi_{ManyOnSrc}$: The section RelocateConstraint(#14-21) and RelocateConstraint(#38) are applied. Correctness of navigations of the form $self_{Old}$.Dest is granted by RelocateConstraint(#38) according to constructs used in $\varphi_{ExactlyOne}$. However, navigations from B to A results in a set of a_is. The condition at RelocateConstraint(#14) checks whether such navigation is required (using the rule ForAllCheck). Backward navigation to the original constraint is required in

the case of $self_{Old}$.x expressions (where x <> "Dest"), or if Mul(A,Dest)[1]>1 according to constructs used in $\varphi_{ManyOnDest}$. Here only the first case is possible, which is handled by adding a *forall* expression to the constraint during relocation (RelocateConstraint(#15-21)). The new *forall* expression encapsulates the whole constraint and it simulates the backward navigation using the iteration variable 'OrigSelf', where navigation expressions are replaced by variable calls in AddBackNavigation(#5-6). Therefore, the constraint … `self`$_{Old}$`.x` … is transformed into `self`$_{New}$`.Src→`*`forall`*`(OrigSelf | … OrigSelf.x …)`.

For a certain expression and b_j, the replacement is correct for a_is connected to b_j. Since φ_{AtoB} holds, thus, each a_i is connected with at least one b_j and evaluation checkes each b_j of the model, thus, the replacement is correct for all a_is of the model. Note that the outermost *forall* expression ensures that different navigation/attribute calls of the constraint are using the same a_i (since the value of OrigSelf does not change), thus the attributes/navigation of a_is are not mixed up.

$\varphi_{ManyOnBoth}$: RelocateConstraint(#14- 21) and RelocateConstraint(#28-34) are used. Firstly an encapsulating *forall* expression is added if there is a navigation/attribute call of form $self_{Old}$.x (where x<>"Dest") or a $self_{Old}$.Dest expression without *forall*. The construction from $\varphi_{ManyOnSrc}$. Secondly, the *forall* expressions of the original constraint are replaced according to constructs used in $\varphi_{ManyOnDest}$.

The replacement of the expression $self_{Old}$.x (x<>"Dest") in a certain a_i is correct for each $b_j \in To(a_i, Dest)$ according to constructs used in $\varphi_{ManyOnSrc}$ Moreover, the replacement of these type of expressions is also correct in general (because of φ_{AtoB}). However, it is possible that the expression is evaluated in a certain a_i more then once (more precisely once for each element of $To(a_i, Dest)$).

The replacement of the expression $self_{Old}$.Dest `→`*`forall`* is correct according to constructs used in $\varphi_{ManyOnDest}$. Multiplicity 'MoreThanOne' on the source side does not affect this correctness, because the updated expressions use navigations only from context B, thus, it is not important how many a_is are connected with the current b_j.

This is not the case with expressions of form $self_{Old}$.Dest->Exp, where Exp is not *forall*. Here navigation back to the original context is mandatory, thus the original expressions is transformed into $self_{New}$.Src.Dest->Exp. Relocation is correct in this case if the value of $self_{Old}$ can be replaced by $self_{New}$.Src. However, the function *ForAllCheck* returns true at the condition at RuleConstraint(#14), which means that the constraint is encapsulated by a new *forall* as in $\varphi_{ManyOnSrc}$. This *forall* expression ensures that for a certain b_j, the constraint is evaluated for all items of the set $self_{New}$.Src seperately (for all a_is connected with b_j). Moreover φ_{BtoA} holds, which means that all a_is are checked by the relocated constraint. This means that the relocation is correct in this case as well.

The possible multiplicity combinations were tested, we have proved that the RelocateConstraint rule is correct in all cases.

Conclusions

Textual constraints are useful in order to extend visual model definitions and create precise models. OCL is one of the most popular textual constraint language, it is used to provide precise, unambiguous definitions in several modeling techniques such as UML, or metamodeling techniques in general. One of the key features of OCL is the mathematical formalism based on set theory with a notion of an object model and system states. This formalism describes the syntax and semantics of OCL and it can prove the completeness of the models using OCL, but it does not contain the definition of constraint evaluation, dynamic behavior of the constraint expressions. Due to this limitation of the OCL formalism, it cannot be used to prove the correctness of dynamic, OCL manipulating algorithms, for example our optimization algorithms.

This paper has presented OCLASM, a new formalism for OCL. The paper has presented the main reasons, why a new formalism was created instead of extending the original formalism, or one of its extensions. The new formalism is based on the popular ASM technology, it can be used to study the dynamic behavior in a compact, yet rigorous way. The basic idea of the formalism is to create rules for all language expressions, such as `iterate`, and use these rules to simulate the validation. OCLASM handles the constraints as a sequence of statements and expressions and it navigates through these programming units using a function pointing to the current expression. Model-based operations and constraint expression retrievals use monitored (external) functions showing that constraint validation must be independent from the current model and constraint representation. The mechanism of the formalism method has been shown including how to handle language construct, such as tuple types, or collections. The formal definition of OCLASM has also been presented and the paper also includes several rules for the most important language constructs. Using the new formalism of OCL, it is possible to create and validate algorithms based on OCL. The paper has shown how to use OCLASM in order to define RelocateConstraint algorithm (which is used in constraint optimization) and how to prove its correctness formarly. Future work mainly consists of continuing this work and prove the correctness of other optimization algorithms as well.

Acknowledgement

The paper is established by the support of the National Office for Research and Technology (Hungary).

References

[1] L. Lengyel, T. Levendovszky, H. Charaf: Constraint Handling in Feature Models, 5th International Symposium of Hungarian Researchers on Computational Intelligence, 2004

[2] L. Lengyel, T. Levendovszky, G. Mezei, B. Forstner, I. Kelényi, H. Charaf: Model-based System Development for Embedded Mobile Platforms, ECBS, pp. 43-52, 2006

[3] J. Warmer, A. Kleppe: Object Constraint Language, The: Getting Your Models Ready for MDA, Second Edition, Addison Wesley, 2003

[4] G. Mezei, L. Lengyel, T. Levendovszky, H. Charaf: Extending an OCL Compiler for Metamodeling and Model Transformation Systems: Unifying the Twofold Functionality, INES 2006

[5] G. Mezei, T. Levendovszky, H. Charaf: An Optimizing OCL Compiler for Metamodeling and Model Transformation Environments, Working Conference of Software Engineering, 2006

[6] G. Mezei, T. Levendovszky, H. Charaf: Restrictions for OCL Constraint Optimization Algorithms, OCL for (Meta-) Models in Multiple Application Domains (OCLApps) Workshop, 2006

[7] VMTS Web Site, http://vmts.aut.bme.hu

[8] E. Börger, R. Stärk: Abstract State Machines: A Method for High-Level System Design and Analysis, Springer-Verlag, 2003

[9] M. Richters: A Precise Approach to Validating UML Models and OCL Constraints. PhD thesis, University at Bremen, Bremen, Germany, 2001

[10] S. Flake: Towards the Completion of the Formal Semantics of OCL 2.0 ACSC, pp. 73-82, 2004

[11] S. Flake, W. Müller: An ASM Definition of the Dynamic OCL 2.0 Semantics, UML 226-240, 2004

[12] R. F. Stärk, J. Schmid, E. Börger: Java and the Java Virtual Machine: Definition, Verification, Validation, Springer Verlag, 2001

[13] G. Mezei, T. Levendovszky, H. Charaf: An Attribute Transformation Technique For N-Layer Metamodeling, Microcad, 2007

[14] UML 2.0 Specification http://www.omg.org/uml/

[15] G. Mezei, T. Levendovszky, L. Lengyel, H. Charaf: Multilevel Metamodeling - A Case Study, MicroCAD, March 10-11, 2005, Miskolc, pp. 321-326

Perspective Methods and Tools for the Design of Distributed Software Systems Based on Services

Marek Paralič

Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics (FEI), Technical University in Košice
Letná 9, 042 00 Košice, Slovakia
Marek.Paralic@tuke.sk

Abstract: In this paper the current research activities of the Distributed team at the Department of Computer and Informatics, FEI, TU in Košice are described. Our focus is to propose and verify new methods and tools, which could contribute to the design and implementation of integrated, pervasive and collaborative services. Pervasive computing is an important research area whose challenges require a thorough rethinking and revision of conventional software design ideas. In reality, there is little consensus and very little basic understanding of the underlying issues and their interactions to produce useful solutions. Our research activities aim to perform the research necessary to contribute to this understanding.

Keywords: service-oriented computing, agent technology, pervasive computing, semantic web services

1 Introduction

Research at the Department of Computers and Informatics in the area of distributed software systems design is supported by the Slovak National Grant Agency and in the past covered also topics such as: methods and tools for design of the distributed programming systems [1] or design and implementation of a technological platform for creating multi-agent applications with hierarchical structure of mobile and static agents [2][3]. Our research in last decade has been focused on methods and methodologies in the area of distributed software systems with strengthened emphasis on agent technologies, whereby the major results achieved include the following:

- architecture that combines mobile agent paradigm with formal foundation of concurrent constraint programming, logic variables and true distributed systems with shared distributed data structure [1],

- definition of simple form for formal expression, control and analysis of dynamic properties (communication and mobility) of mobile software applications [3],
- tool for modelling of multi-agent system that respects security requirements of agents and places, which utilizes modified calculus of mobile ambients [3],
- agent based scheme for generic service for static and mobile users [2],
- multi-agent platform for support of design, implementation and maintenance of flexible distributed services for mobile users, which was verified on the set of simple services designated to the university communities [2][13].

One of the current high interesting areas in the distributed systems is the problem of building and running pervasive computing services. Pervasive computing is an important research area whose challenges require a thorough rethinking and revision of conventional software design ideas. In reality, there is little consensus and very little basic understanding of the underlying issues and their interactions to produce useful solutions. Our research activities aim to perform the research necessary to contribute to this understanding.

Pervasive computing can be seen as a method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user. Pervasive computing system should faced up to the following features [4]: (i) pervasive – it must be everywhere, (ii) embedded – it must live in our world, sensing and affecting it, (iii) nomadic – it must allow users and computations to move freely, according to their needs, (iv) adaptable – it must provide flexibility and spontaneity in response to changes in user requirements and operating conditions, (v) powerful, yet efficient: it must free itself from constraints imposed by bounded hardware resources, addressing instead system constraints imposed by user demands and available power or communication bandwidth, (vi) intentional – it must enable people to name services and software objects by intent, and (vii) eternal – it must be available all the time.

In our current research activities we focus on the methods and tools, which could support the design and implementation of pervasive services, whereby we utilise our experiences with mobile agents and agent technologies in general, service-oriented architecture and technologies of the semantic web. The following sections bring the basic characteristics of the applied technologies – agent technology, service-oriented design based on web services and semantic web technologies relevant for our purposes. In the conclusion our future plans are briefly sketched.

2 Agent Technologies

Agents can be defined to be autonomous, problem-solving computational entities capable of effective operation in dynamic and open environments. Agents are often deployed in environments in which they interact, and sometimes cooperate with other agents (including both, people and software) that have possibly conflicting aims. Such environments are known as multi-agent systems [5]. The term multi-agent system is now used for all types of systems composed of multiple autonomous components showing the following characteristics [6]: each agent has incomplete capabilities to solve a problem, there is no global control system, data is decentralized and computation is asynchronous. Agent technology enables the realization of complex software systems characterized by situation awareness and intelligent behaviour, a high degree of distribution, as well as mobility support. Agent technology has the potential to play a key role in enabling intelligent applications and services by improving automation of routine processes, and supporting the nomadic users with pro-active and intelligent assistance based on principles of adaptation and self-organization [7].

On the field of standardization provides The Foundation for Intelligent Physical Agents (FIPA)[1] a comprehensive set of specifications from messaging and directories, through system management, agent mobility, up to well-formed semantic communication languages for agents. One of the most popular FIPA-compliant agent frameworks is JADE (Java Agent DEvelopment Framework)[2] with the LEAP (Lightweight Extensible Agent Platform) enhancement that enables to run agent platform on small devices like PDAs and mobile phones [9]. Further activities represent OMG[3], conception for communication and cooperation, design languages for agent-based systems with supported tools like AUML[4]. The goal of such activities is to improve the analysis and specification of the multi-agent systems.

Some of the key industrial application areas for agent-based systems are [10]: monitoring and control of physically highly distributed systems; real-time control of high-volume, high-variety, discrete manufacturing operations and transportation and material-handling systems. Another application area for agent technology is building of smart environment based on pervasive computing – e.g. EasyMeating system [11] that explores the use of FIPA agent technologies, Semantic Web ontologies, logic reasoning, and security and privacy policies. Agent technology can open the way to new application domains while supporting the integration of existing and new software, and make the development process for such applications easier and more flexible [7][12][12].

1 The Foundation for Intelligent Physical Agent (FIPA), http://www.fipa.org/

2 Java Agent DEvelopment Framework (JADE), http://jade.cselt.it/

3 Object Management Group, http://www.omg.org

4 Agent UML, http://www.auml.org

Despite many of relevant results, multi-agent systems have not become widespread as industrial and commercial applications. In order to bridge the gap between agent technology and methodologies or technologies accepted for real world applications some efforts have been performed. A survey of agent-oriented methodologies can be found in [14]. Some of the most interesting results are Gaia [15] (methodology for agent-oriented analysis and design supporting macro/societal level as well as micro/agent level aspects) and MaSE [16] (object-oriented approach for support of the complete software lifecycle from problem description to realization), which we also applied when developing our own agent-based systems (e.g. [12]).

3 Service-oriented Computing and Web Services

Services are means for building distributed applications and are used to build service-based applications. In this context the service composition is of great importance, whereby it should be supported by every aspect of services – they can be described, selected, engaged, collaborated with and evaluated. Service-oriented computing is a general topic, more specifically in the context of WWW technologies we are speaking about Web Services.

As defined by the World Wide Web Consortium [17], a Web Service is a software system identified by a URI (RFC 2396), whose public interface and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web Service in a manner prescribed by its definition, using XML-based messages conveyed by Internet protocols. Web Services rely on the functionalities of publish, find, bind [18] and the components of a Web Service Model include Service Providers, Service Broker and Service Requester. Web Services are defined by their interfaces in particular about how they describe their functionality, how they register their presence, and how they communicate with other Web Services. People who want to use Web Services could connect to the Universal Description Discovery and Integration (UDDI) centre to search for required services. The information about the Web Services described by Web Service Description Language (WSDL) can be acquired. And the user could use the Simple Object Access Protocol (SOAP) to transfer the requirement information and receive the real service.

As described above, core Web Service technology (WSDL, UDDI) defines formal interface contracts, describing the message syntax, but does not address the semantics of those interfaces. This means, that the meaning of the exchanged data is not formally described. Since the emerging Semantic Web and Web Services have a similar target audience, namely application clients, therefore they are intended for automated processing and share a common base technology (XML),

it is obvious to apply semantic web techniques to web services. The Resource Description Framework (RDF) is particularly intended for representation of metadata about web resources in general and web services in particular and represents a notation to express structured metadata which has also a XML-based format representation (RDF/XML).

On higher level of abstraction there are several web ontology languages which are based on RDF notation and are used for knowledge modelling, i.e. define vocabularies and express classes of information by means of the vocabulary and their inherent relationships. The most applied ontology language is OWL (based on DAML, etc.). For the specific application field of web services, languages like OWL-S are evolving. These languages are derived from OWL and enable to define richer semantic for service specifications. This can enable richer, more flexible automation of service provision and use, and support the construction of more powerful tools and methodologies [19]. Another example of research activities in the same area is the Web Service Modelling Ontology (WSMO) and its reference implementation The Web Service Execution Environment (WSMX) [20].

4 Service Compositions in Open Environment

Desirable abstractions for service composition in open environments should exploit opportunities offered by those services:

(i) because services are autonomous, we should not require them to be subservient to other services,

(ii) because services are heterogeneous, we should develop expressive, standardizable representations (presently this is done for data, but not for processes and policies),

(iii) because services are long-lived, evolving and operate in environments that produce exceptions, representations should handle such situations,

(iv) because services can be cooperative, abstractions would represent how they behave in the awareness of the behaviours of other services.

Current approaches are connected to low-level invocation of services – they are not specially geared for enabling composition. Services are integrated through method invocation without regard to any higher-level constraints. Semantic web technologies make use of prior work done in workflow management systems, artificial intelligence approaches for planning, formal process models, multi-agent planning and description logic. The objectives for the development of Semantic Web services are to enable reasoning about Web services, planning compositions of Web services and automating the use of services by software agents. The goal

is to make Web services unambiguously interpretable by a computer. Examples of ontologies used for description of Web services are already mentioned OWL-S or WSML. Using these ontologies, a Web service can advertise its functionality to potential users. A request for a service would then be matched against the Web service's advertisement via a matchmaking process, because the objectives of such a request are expressed as goals, which are high level descriptions of concrete tasks. Every requester expresses its goal in terms of its own ontology, which provides the means not only for human to understand the goal, but also for a machine to interpret is as a part of requester's ontology. Similar to the goal description, all semantic web services have their descriptions in their own ontology. In case the ontologies differ, the mediation process should be applied. Mediation is also utilized in case of heterogeneous communication protocols of involved services.

There are many formalisms for describing a business process, such as BPMI or BPML and OWL-S was designed to be neutral with respect to a particular formalism. It instead just provides the necessary vocabulary and properties for a process model. Both the OWL-S service model and WS-BPEL provide a mechanism for description of a business process model. However, they can be contrasted in terms of their expressiveness, representations, semantics, discovery support, execution support and fault handling [21].

Conclusions

In the centre of our attention are methods and tools for building of integrated services in a distributed environment, whereby as basic constructing elements the agent-oriented design and service-oriented design are used. To achieve the goal – design of pervasive integrated services – the agent technology is applied, whereby the concept of the higher-level agent – ambient – is proposed. Ambients support the process of building dynamic workflows of services, whereby semantic description of services is used. Utilisation of semantic information offers improved capabilities in terms of their expressiveness, representations, discovery support, execution support and fault handling. During our future research we strive to give precise semantic of identified workflow patterns, which will be integrated into tools aimed to support design and execution of integrated pervasive services.

Acknowledgement

Research described in this paper is supported by the project '***Integrated distributed applications based on ambients – higher-level agents***' Nr. 1/3135/06 of the Slovak Grant Agency.

References

[1] Paralič, M.: Mobile Agents Based on Concurrent Constraint Programming, Joint Modular Languages Conference (JMLC 2000), September 6-8, 2000, Zurich, Switzerland, Lecture Notes in Computer Science, Vol. 1897, pp. 62-75

[2] Paralič, M., Krokavec, M.: Multi-Agent Applications Design Support. Journal of Information and Organizational Sciences, Vol. 27, No. 2, 2003, ISSN 0351-1804, pp. 101-108

[3] Tomášek, M.: Interactions of Mobile Agents in the Distributed System, in Journal of Information, Control and Management Systems, Vol. 3, Nr. 2, 2005, ISSN 1336-1716, pp. 173-182

[4] Liu, R., Chen, F., Yang, H., Chu, W. C., Yu-Bin Lai: Agent-based Web Services Evolution for Pervasive Computing, in Proceedings of 11th Asia-Pacific Software Engineering Conference (APSEC 2004), November 30-December 3, 2004, Busan, Korea, pp. 726-731

[5] Luck, M., McBurney, P., Preist, Ch.: Agent Technology: Enabling Next Generation Computing, A Roadmap for Agent Based Computing. AgentLink, 2003, http://www.agentlink.org/roadmap

[6] Jennings, N. R., Sycara, K., Wooldridge, M.: A Roadmap of Agent Research and Development, in Journal of Autonomous Agents and Multi-Agent Systems, Vol. 1(1), 1998, pp. 7-38

[7] Bauer, B., Odell, J.: UML 2.0 and Agents: How to Build Agent-based Systems with the new UML Standard, Journal of Engineering Applications of Artificial Intelligence, Volume 18, Issue 2, March 2005, pp. 141-157

[8] Fallah-Saghrouchni, A. E., Suna, A.: Programming Mobile Intelligent Agents: an Operational Semantics, in Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'04), 2004, pp. 65-71

[9] Berger, M., Bouzid, M., Buckland, M., Lee, H., Lhuillier, N., Olpp, D., Picault, J., Sheperdson, J.: An Approach to Agent-based Service Composition and Its Application to Mobile Business Processes, IEEE Transactions on mobile computing, Vol. 2, No. 3, July-September 2003, pp. 197-205

[10] Mařík, V., McFarlane, D.: Industrial Adoption of Agent-based Technologies. IEEE Intelligent Systems, January/February 2005, pp. 27-35

[11] Chen, H., Perich, F., Chakraborty, D., Finin, T., Joshi, A.: Intelligent Agents Meet Semantic Web in a Smart Meeting Room, in Proceedings of 3rd International Joint Conference on Autonomous Agents and Multiagent Systems, July 2004, pp. 854-861

[12] Cooney, D., Roe, P.: Mobile Agents Make for Flexible Web Services, in Proceedings of the 9th Australian World Wide Web Conference AUSWEB 2003, July 5-9, 2003, pp. 385-391

[13] Paralič M., Paralič J.: Data Analysis for Agent-based Mobile Services, Journal of Information and Organizational Sciences, Vol. 29, No. 1, 2005, ISSN 0351-1804, pp. 53-61

[14] Shehory, O., Sturm, A.: Evaluation of Modeling Techniques for Agent-based Systems. In Proceedings of the 5th International Conference on Autonomous Agents, Montreal, Canada, June 2001, pp. 624-631

[15] Zambonelli, F., Jennings, N. R., Wooldridge, M.: Developing Multiagent Systems: The Gaia Methodology, ACM Transactions on Software Engineering and Methodology, Vol. 12, No. 3, July 2003, pp. 317-370

[16] Deloach, S. A., Wood, M. F., Sparkman, C. H.: Multiagent Systems Engineering. International Journal of Software Engineering and Knowledge Engineering. Vol. 11, No. 3, 2001, pp. 231-258

[17] Web Service Architecture Requirements, W3C Working Group, February 2004, http://www.w3.org/TR/2004/NOTE-wsa-reqs-20040211/

[18] Huhns, M. N.: Agents as Web Services. IEEE Internet Computing, 6(4), 2002, pp. 93-95

[19] Martin, D., et al: Bringing Semantics to Web Services: The OWL-S Approach, SWSWPC 2004, Lecture Notes in Computer Science 3387, 2005, pp. 26-42

[20] Haller, A., Cimpian, E., Mocan, A., Oren, E., Bussler, Ch.: WSMX – A Semantic Service-oriented Architecture, in Proceedings of the International Conference on Web Services (ICWS 2005), IEEE Computer Society Orlando, Florida, USA, 2005, pp. 321-328

[21] Ankolekar, A., Martin, D., McGuinness, D., McIlraith, S., Paolucci, M., Parsia, B.: OWL-S' Relationship to Selected Other Technologies, November 2004, http://www.w3.org/Submission/2004/SUBM-OWL-S-related-20041122/

Deterioration-based Maintenance Management Algorithm

Kornélia Ambrus-Somogyi

Institute of Media Technology, Budapest Tech
Doberdó út 6, H-1034 Budapest, Hungary
a_somogyi.kornelia@rkk.bmf.hu

Abstract: The Road Management Systems (and the PMS) usually do not take into consideration the future traffic change. The maintenance and rehabilitation actions and the development of the road network structure and the changing traffic structure modify the amount of the traffic on the road section. The deterioration process depends on mostly the volume of the traffic. That is why it is important to take into consideration the change of the traffic volume during the planning time horizon. In the lecture some techniques are shown which handle this problem: in multiperiod, long time model at each planning period the traffic volume change is take into consideration. In ranking models the problem could be handled and solved. In the case of one period Markov stabile model there is nothing to do. In the multiperiod model the problem could be solved also.

1 PMS Models

1.1 The Main Questions of Decision Makers of the PMS

The main questions of decision makers of the Road Management System and the PMS:

- How much budget is required each year for maintenance and rehabilitation the elements for their whole lifetime to hold them in a certain condition level?
- Which AM elements condition distribution would be expected if the sum mentions above were to be available?
- What consequences would be expected if this sum were not available?
- What consequences would be expected if the maintenance expenditures were significantly increased?
- What are the optimum times for maintenance actions?

– What consequences would be expected from the delay of necessary actions?

– Who benefits and who loses, to what extent, etc?

1.2 The Models

A decision model for pavement management has been developed herein based on linear programming formulation. The engineering and deterioration model was created by Gáspár [5], the mathematical one by Bakó [2], Markov transition probability matrices are introduce to model the deterioration process of the road sections is determined by this matrices. To every type of road surfaces and class of traffic amount belongs a certain Markov matrix. The presented model and methodology is used to determine the optimal rehabilitation and maintenance policy in network level. Depending on the objective function two types of problems could be solved by the model: the necessary funds calculation and the optimal budget allocation for the entire network.

Several types of solution algorithms can be used depending on the given task, the available data, the budget constraints, etc. Two main types are the heuristic and the optimisation algorithms. The heuristic technique is usually used in project level, but it could be used in network level too. The optimisation models are solved by the traditional optimisation algorithms. Depending on the problem to be solved, we use integer, a linear or a dynamic programming algorithm.

There are several constraints to be fulfilled. We will denote the unknown variable by X_{ijk} which belongs to the pavement type i, to the traffic volume j and to the maintenance politics. The solution have to be Markov stabile. The Markovian stability constrain is

$$\sum_{i=1}^{s}\sum_{j=1}^{f}\sum_{k=1}^{t}\left(Q_{ijk} - E\ \right)X_{ijk} = 0 \tag{1}$$

where E is a unite matrix.

Because the equality is usually not fulfilled or not desirable, we use ≤ or ≥ relation instead of equality in (1). There are several further constraints, which are connected with other suppositions. We suppose that the traffic volume will not change during the planning period:

$$\sum_{k=1}^{t} X_{ijk} = b_{ij}, i = 1,2,....,s \qquad j = 1,2,\ldots f \tag{2}$$

where b_{ij} belongs to the pavement type i and the traffic volume j.

The total area of the road surface type i will remain the same at the end of the planning period

$$\sum_{j=1}^{f}\sum_{k=1}^{t} X_{ijk} = d_i, i = 1,2,\ldots s \tag{3}$$

where d_i belongs to the pavement type i and $\sum_{i=1}^{s} d_i = \underline{1}$.

We have to apply one of the maintenance politic on every road section

$$\sum_{i=1}^{s}\sum_{j=1}^{f}\sum_{k=1}^{t} X_{ijk} = \underline{1} \tag{4}$$

We divide the segments into 3 groups: acceptable (good), unacceptable (bad) and the rest. Let us denote the tree set by J (good) by R (bad) and by E (rest of the segments) and by H the whole set of segments. The relations for these sets are given by

$$J \cap R = \varnothing, \qquad J \cap E = 0$$
$$R \cap E = \varnothing, \qquad J \cup R \cup E = H$$

The following conditions are related to these sets

$$\sum_{i,j,k \in J} X_{ijk} \geq v_J$$

$$\sum_{i,j,k \in R} X_{ijk} \leq v_R \tag{5}$$

$$\underline{v}_E \leq \sum X_{ijk} \leq \bar{v}_E$$

where J, R, E are given above, and

v_j the total length of the good road after the planning period

v_R the total length of the bad road after the planning period

v_E the lower bound of the other road

$\overline{v_E}$ the upper bound of the other road.

The meaning of the first condition is, that the amount of good segment has to be greater than or equal to a given value. The second relation does not allow more bad roads than it is fixed in advanced. The third relation gives an upper and lover limit to the amount of the rest road.

Let us denote by c_{ijk} the unit cost of the maintenance politic k on the pavement type i and traffic volume j. Our objective is to choose such an X which fulfils the conditions given above with minimal rehabilitation cost.

The objective is

$$\sum_{i=1}^{s} \sum_{j=1}^{f} \sum_{k=1}^{t} X_{ijk} C_{ijk} \rightarrow \min! \tag{6}$$

Let us denote this value by C. The budget C* which is available for the maintenance purpose usually less than C, so C*<C. In this case we modify our model: the above mentioned rehabilitation cost function becomes constrained:

$$\sum X_{ijk} C_{ijk} \leq C^* \tag{7}$$

and we use an other objective. Let us denote the benefit by h_{ijk} where this is the benefit of the societies when we apply on the pavement type i and with the traffic volume j the maintenance politic k.

Our aim is to determine such a solution X which fulfils the constraints (1)-(5) and (7) and maximizes the total benefit of the society.

The objective in this case is

$$\sum X_{ijk} h_{ijk} \rightarrow \max! \tag{8}$$

Gáspár [6, 7] summarizes the detailed of the using algorithms.

The Road Management Systems (and the PMS) usually do not take into consideration the future traffic change. The deterioration process depends on mostly the volume of the traffic.That is why it is important to take into consideration the change of the traffic volume during the planning time horizon.

2 Traffic Planning and Forecasting

A lot of problems are known to be connected with traffic planning, forecasting, distribution and assignment:

- determination of the path with minimal travelling time between every pair of points (SP-problem);
- forecast of the future traffic starting from the source point (Forecasting);
- determination of the traffic on the road segments (Assignment problem).

2.1 SP Problem

At the SP problem let N be a finite set of points, E the set of edges, and $\mathrm{d}(\mathrm{x},\mathrm{y})>0$ the travelling time on the edge (x,y). The travelling time on path $\mathrm{P}=(\mathrm{x}_1,\mathrm{x}_2,\ldots,\mathrm{x}_\mathrm{r})$ is

$$L(P)=\sum_{i=1}^{r-1} d(x_i, x_{i+1}). \tag{9}$$

The multiterminal shortest path problem can be formulated as follows: determine the path P with minimal travelling time for every pair of points.

There are several algorithms for solving this problem. The best method was given by Warshall [9]. This algorithm consists of the following steps:

$$\begin{aligned} d_{ij}^{(0)} &= d(x_i, x_j), \\ d_{ij}^{(k)} &= \min\left(d_{ij}^{(r-1)}, d_{ir}^{(r-1)} + d_{rj}^{(r-1)}\right) \end{aligned} \tag{10}$$

where n is the number of points. This algorithm needs the least number of computational steps; only n^3 additions and comparsions are needed. The matrix $D^{(n)}=\left(d_{ij}^{(n)}\right)$ gives the length of the minimal paths. In order to obtain the paths themselves we compute another so called labelling matrix $S^{(k)}$:

$$\begin{aligned} s_{ij}^{(0)} &= j \\ s_{ij}^{(k)} &= \begin{cases} s_{ij}^{(r-1)}, if\ d_{ij}^{(r-1)} \le d_{ir}^{(r-1)} + d_{rj}^{(r-1)}, \\ s_{ir}^{(r-1)} \ldots otherwise \end{cases} \end{aligned} \tag{11}$$

2.2 Traffic Forecasting and Assignment

Fratar problem: Given a matrix $A=(a_{ij})$. The element a_{ij} means the amount of traffic from point *i* to point *j*. Let us denote by k_i respectively by l_i the amount of traffic leaving respectively entering to point *i*. The Fratar problem (F-problem) is to determine the future traffic matrix from the matrix *A* using the given k_i and l_i values. In the case of Gravity problem Instead of matrix *A* matrix *D* is given where the element d_{ij} means the minimal travelling time from point *i* to point *j*. These two problems seem to be different but both problems can be solved by the forecasting of the input-output (I/O) table.

Let $K_1, K_2, \ldots K_m$ be sources and let $L_1, L_2, \ldots L_m$ be sinks. The element $a_{ij} \geq 0$ of matrix A means the traffic from K_i to L_j. Let us denote by k_i respectively by l_j the total traffic leaving K_i respectively L_j:

$$k_i = \sum_{j=1}^{n} a_{ij}, (i = 1, 2 \ldots, n),$$
$$l_j = \sum_{i=1}^{n} a_{ij}, (i = 1, 2 \ldots, n). \tag{12}$$

The matrix A is called I/O table, and elements k_i, l_j are called the marginal values of I/O table *A*.

Let $k_1, k_2, \ldots, k_n$ and $\lambda_1, \lambda_2, \ldots, \lambda_n$ be new marginal values $(k_i \geq 0,\ \lambda_i \geq 0,\quad i = 1, 2, \ldots, n)$. The forecasting of the I/O table is to determine the new I/O table X from A and given marginal values k_i, λ_j so that the tables A and X are 'similar'. The two tables are similar, when x_{ij} can be determined in the following form

$$x_{ij} = \xi_i a_{ij} \eta_j \tag{13}$$

where ξ_i, η_j are unknown. Klafszki [8] Bakó [1] has shown by the help of geometric programming that this problem is equivalent to the minimalization of information divergence. The table X has to satisfy the equations (12).

$$\sum_{j=1}^{n} x_{ij} = k_i, \quad (i = 1, 2, \ldots, n)$$
$$\sum_{i=1}^{n} x_{ij} = \lambda_j, \quad (i = 1, 2, \ldots, n) \tag{14}$$

Using equations (13) and (14) we obtain

$$\xi_i \sum \eta_i a_{ij} = k_i, \quad \eta_j \sum \xi_i a_{ij} = \lambda_j \tag{15}$$

D'Esopo [4] suggested the following method for solving the equations (16):

Step 0 $\qquad \eta_j^{(0)} = \lambda_j, \quad (j = 1, 2, \ldots, n)$

Step r

$$\xi_i^{(r)} = \frac{k_i}{\sum_j n_j^{(r)} a_{ij}}, \quad (i = 1, 2, \ldots, n),$$
$$\eta_j^{(r)} = \frac{\lambda_j}{\sum_i \xi_i^{(r)} a_{ij}}, \quad (j = 1, 2, \ldots, n) \tag{16}$$

Bergmann [3] proved the convergence of the algorithms. The algorithm is terminated, when the error

$$\max_{1 \le i \le \pi} \left| \frac{\sum \xi_i a_{ij}^{n_j - k_i}}{k_i} \right| \tag{17}$$

is sufficiently small.

Traffic assignment is the process of allocating all the trips in one or more O/D matrices to their route in the network, resulting the flows on its links. The traffic assignment methods employ three basic steps: SP problem, assignment a part of the traffic on the segments and check for convergence.

3 Combined Algorithms

The maintenance and rehabilitation actions and the development of the road network structure and the changing traffic structure modify the amount of the traffic on the road section. The deterioration process depends on mostly the volume of the traffic.

That is why it is important to take into consideration the change of the traffic volume during the planning time horizon.In the first part the PMS models are summarised. In the second part we illustrate the traffic forecast, distribution any assignment algorithm (TFDA).

We proposed a new model with:new segments each time period,

– traffic forecasting and assignment each planning period.Now the basic idea of combined algorithm is demonstrated. The algorithms are a long range multitime period ones, we compute the maintenance and rehabilitation actions for each year. After determination the actions in a year, a TFDA (*traffic forecast, distribution any assignment algorithm*) step is taken, where the new traffic on the segments are determined.

After that new traffic categories is formed and the next PMS step (for the next time period) use this new traffic categories. The algorithms demonstrate is both cases the steps of the algorithm.

The frst algorithm is shown in the case of heuristic (ranking) method:

Step 1 This is a PMS ranking for the first time period.

Step 2 A TFDA step is given.

Step 3 This computation continues until the last period is finished. The number of the time period is n.

The combined algorithm is shown in the case of Markov deterioration model is the following:

Step 1 The first step is to compute the Markov stable solution. Here traffic change is not taken into consideration.

Step 2 The yearly deterioration and maintenance process is determined.

Step 3 After in each year a TFDA algorithm is taken and the result of that is used to form the new traffic categories for the next year.

Step 4 The algorithm is finished when we compute the deterioration and maintenance actions for the last time period.

The procedures given above serve better results because the traffic changes are taken into consideration in each year.

References

[1] Bakó, A.. Mathematical and Computational Evaluation and Development of Traffic Planning and Forecasting, Candidate dissertation, HAS, 1980, p. 220

[2] Bakó, A.: Decision Supporting Model for Highway Maintenance, Acta Politechnica Hungarica, 2004, pp. 96-108

[3] Bergman, L.: Dokazatyelsztvo sztodimosztyi G.B., Selejkkoszkov dlja zadacsi sz transzportnümi organicsamijami, Zsurnal Vücsiszl. Mat. I. Mat. Fiziki 1, 1967, pp. 147-156

[4] D'Esopo, D. A.: Lekkowitz, An Algorithm for Computing Intersonal Transfers Using the Gravity Model, OPNS, Res. 11, 1963, pp. 901-907

[5] Gáspár L.: Development of the First Hungarian PMS, Transportation Research, 1992, pp. 132-141 (in Hungarian)

[6] Gáspár, L.: Ein netzbezogenes Managementsystem für die Shassenerhaltung in Ungarn, Strasse und Autobahn 1992/8, pp. 123-127.

[7] Gáspár, L.: Road Management, Hungarian Academy of Sciences, 2003, p. 361 (in Hungarian)

[8] Klafszky, E.: On the Prediction of the Input-Output Table, Report of Computig and Aut. Inst. 10, 1973, pp. 1-13, (in Hungarian)

[9] Warshall, S.: A Theorem of Boolean Matrices, J. of ACM 9, 1962. pp. 11-12

Case-based Reasoning in Agent-based Decision Support System

Jolana Sebestyénová

Institute of Informatics, Slovak Academy of Sciences
Dúbravská cesta 9, 845 07 Bratislava, Slovakia
sebestyenova@savba.sk

Abstract: The paper describes decision support system for modeling, control, and simulation of continuous, as well as discrete event systems. Models, control methods, and tools are in database specified by their attributes. Each attribute's weight is initially estimated according to importance and classification power of a given feature. Automatic learning of attributes weights uses the answers of the users after simulation provided by the system to increase the quality of case-based reasoning. The proposed learning algorithm guarantees convergence of the attributes weights to relatively steady values yielding to Case-based reasoning of best quality. If simulation of a new case was successful from the point of view of the user, the new case is added to case base.

Keywords: decision support system, case-based reasoning, case-based learning, learning of weights

1 Introduction

Decision support system is an interactive computer-based system intended to help decision makers use communications technologies, data, documents, knowledge and/or models to identify and solve problems, complete decision process tasks, and make decisions.

The concept of decision automation is deceptively simple and intriguingly complex. From a narrow perspective, a decision is a choice among defined alternative courses of action. From a broader perspective, a decision involves the complete process of gathering and evaluating information about a situation, identifying a need for a decision, identifying or in other ways defining relevant alternative courses of action, choosing the 'best', the 'most appropriate' or the 'optimum' action, and then applying the solution and choice in the situation. Automation refers to using technologies including computer processing to make decisions and implement programmed decision processes. Typically decision

automation is considered most appropriate for well-structured, clearly defined, routine or programmed decision situations.

A decision support system [7] can be approached from two major disciplinary perspectives, those of information systems science and artificial intelligence. We present in this paper an extended ontology for a decision support system in control theory domain. The ontology explicates relevant constructs and presents a vocabulary for a decision support system, and emphasizes the need to cover environmental and contextual variables as an integral part of decision support system development and evaluation methodologies. These results help the system developers to take the system's context into account through the set of defined variables that are linked to the application domain. With these extensions the focus in decision support systems development shifts from task ontology towards domain ontology.

Most AI systems operate on a first-principles basis, using rules or axioms plus logical inference to do their work [5]. Those few reasoning systems that include analogy tend to treat it as a method of last resort, something to use only when other forms of inference have failed. The exceptions are case-based reasoning systems, which started out to provide computational mechanisms similar to those that people seem to use to solve everyday problems. Unfortunately, case-based reasoning systems generally have the opposite problem, tending to use only minimal first-principles reasoning.

Decision support system for modeling and control of continuous as well as discrete event systems developed in MARABU project [3] is used to illustrate reasoning (see Figure 1). The database of the support system contains methods and tools for modeling of the systems, control synthesis, and simulation. Further, the database contains complete models of some systems.

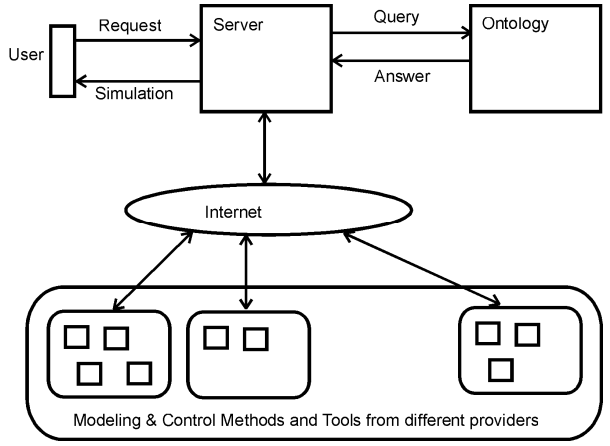


Figure 1
Grid design of MARABU

2 Agent-based Decision Support System

In knowledge engineering, agents offer the flexibility to integrate many different categories of processing within a single system. Agent definitions range from descriptions based on a functional analysis of how agents are used in technology to far more ranging expositions based on different interpretations of the role and objectives of artificial intelligence and cognitive science. Artificial intelligence is a very diverse field and agents are used as metaphors for work in many areas.

Multi-agent systems are appropriate for domains that are naturally distributed and require automated reasoning [2]. Agents should perform the following capabilities to some degree:

- Planning or reacting to achieve goals,
- Modeling the environment to properly react to situations,
- Sensing and acting,
- Inter-agent coordination,
- Conflict resolution (coordination is a continuous process, conflict resolution is event-driven, triggered by conflict detection).

To design a multi-agent system for a given problem, the designer has to understand how should agent and AI techniques be applied to the domain, what competencies agents need, and which techniques implement those competencies. Thus, multi-agent system design consists of (1) dividing resources and domain responsibilities among agents, (2) determining which core competencies satisfy which domain responsibilities, and (3) selecting techniques to satisfy each core competency. According to distributed domain-specific responsibilities agent-based systems may be heterogeneous, with each agent responsible for a different set of goals or homogeneous, where agents share the same goals. Agents in the proposed system work according to simple workflow that is specified by user in terms of required support.

Decision support systems are used by people who are skilled in their jobs and who need to be supported rather than replaced by a computer system. The broadest definition states that decision support system is an interactive computer-based system or subsystem intended to help decision makers use communications technologies, data, documents, knowledge and/or models to identify and solve problems, complete decision process tasks, and make decisions. Five specific decision support system types include [7]:

- Communications-driven DSS,
- Data-driven DSS,
- Document-driven DSS,
- Knowledge-driven DSS,
- Model-driven DSS.

2.1 Knowledge Representation

There are two kinds of conceptual knowledge: concept and set. A concept is defined by the essence of the objects it subsumes and not by their state. Such a definition allows us to focus on the essence of the concepts and not on their state. An essence is invariant, which is not the case of state. On the other hand, a set makes it possible to put together objects whose state shares some common properties. For instance, if 'Human Being' refers to a concept, 'Teenagers' refers to a set composed of human beings whose age is in given constraints.

Differences are elementary units from which the meaning of terms is built. This means they have no meaning in themselves. A difference belongs to the essence of objects. Unlike an attribute it cannot be removed from the definition of an object without changing its nature; nor can it be valued. For example, for human beings 'mortal' is a difference whereas 'age' is an attribute. A difference is a unit that builds meanings and divides concepts. Classification of concepts used in such a large area is not a trivial task [8].

In database design, it is important to properly arrange and index the attributes to achieve effective reasoning. The proposed database consists of three parts: DB of methods and tools that are available, case-base of concrete examples, and knowledge base of control theory domain. They are arranged in 37 tables of relational database. For any model, we need to know which system is the model of, what modeling method is used, which tool was used to create the model, and what modeling requirements were given. Similar specifications are used for control, too.

A straight comparison between a DB and ontology needs to take the nature of the data into account. The advent of object orientated databases, improved logics and faster inference is making the distinction between DBs and ontologies more fuzzy [4]. For a domain of control theory, following terms have to be defined in the ontology: system, model, control, system description method, control method, and tool. Each term is characterized by attributes. Concept hierarchy plus attributes gives ontology. Relations that are used in the used ontology are: part of, attribute of, value of, is a (subclass of), instance of.

Knowledge base of described agent-based decision support system contains ontology-based representation of data relevant to control theory domain and further data needed for system functionality [8]:

- Persistent data stored in database,
- Temporary information stored in system variables, e.g. type of required support is stored invariables: *model, control, simulation, similar_cases*.

2.2 Web Portal of Decision Support System

The web-based portals prove to be very suitable for knowledge management. Knowledge portals are flexible and easy to use and may provide almost any kind of content or functionality. To structure the architecture of a knowledge portal, the following three-layer model is being used: (1) user interface and navigation, (2) functions (personalization, active process support, coordination of agents, document management), and (3) knowledge base. Knowledge portals provide a flexible knowledge environment to a potentially large number of users. The mission of a knowledge portal is not only to provide a library-like pool of information, but to actively support the user in his or her decision processes.

Agent-based decision support system MARABU [9] consists of: (1) database of modeling and control methods and tools, (2) case base of models created and simulated in past, (3) knowledge base of the control theory domain, (4) web-portal enabling to specify user requirements, to display results of reasoning, and to connect a provider of a selected tool. Web portal of proposed support system provides basic information about the system, besides obvious functions, such as registration and login of authorized users. Web portal further enables authorized users to:

Specification of a required support: User can choose any combination from four given options: creation of a model of a specified system, control synthesis, simulation, or list of accomplished similar cases.

Basic system characteristics: User has to choose one of the three basic system types: continuous systems, discrete event systems, distributed systems. Basic system characteristic determines content and sequence of questionnaires for specification of attributes and requirements, as well as context of reasoning.

Questionnaires: Actual workflow of the system is determined by requirements fulfilled in the questionnaires. User needs to specify only required values of those attributes that are important from his point of view. All possible values of not specified attributes are taken by reasoning algorithm as applicable. Usage of modeling and control methods and tools provided by the decision support system is described in help files.

The proposed support system enables devices to be virtually shared, managed, and accessed across a consortium or workgroup. Although the physical resources may reside in multiple locations, users have seamless and uninterrupted access to these resources.

3 Reasoning

There are two main directions in DB reasoning: forward chaining and backward chaining algorithms. Forward chaining is an example of the general concept of data-driven reasoning - that is, reasoning in which the focus of attention starts with the known data. It can be used within an agent to derive conclusions from incoming percepts, often without a specific query in mind. New facts can be added to the agenda to initiate new inferences.

The backward-chaining algorithm, as its name suggests, works backwards from the query. If the query q is known to be true, then no work is needed. Otherwise, the algorithm finds those implications in the knowledge base that conclude q. Backward chaining is a form of goal-directed reasoning. It is useful for answering specific questions such as 'What shall I do now?' Often, the cost of backward chaining is much less than linear in the size of the knowledge base, because the process touches only relevant facts. In MAS, an agent should share the work between forward and backward reasoning, limiting forward reasoning to the generation of facts that are likely to be relevant to queries that will be solved by backward chaining.

One of the bottlenecks in the creation of AI systems is the difficulty of creating large knowledge bases. There have been a number of systems that capture some aspects of reasoning by analogy. No previous analogy systems have been successfully used with multiple, large general-purpose knowledge bases created by other research groups. While the majority of today's CBR systems have moved to feature-vector representations, there are a number of systems that still use relational information.

3.1 Case-based Reasoning

There is mounting psychological evidence that human cognition centrally involves similarity computations over structured representations, in tasks ranging from high-level visual perception to problem solving, learning, and conceptual change. Understanding how to integrate analogical processing into AI systems seems crucial to creating more human-like reasoning systems [6]. Yet similarity plays at best a minor role in many AI systems. Most AI systems operate on a first-principles basis, using rules or axioms plus logical inference to do their work. Those few reasoning systems that include analogy tend to treat it as a method of last resort, something to use only when other forms of inference have failed.

The exceptions are case-based reasoning systems, which started out to provide computational mechanisms similar to those that people seem to use to solve everyday problems. Unfortunately, CBR systems generally have the opposite problem, tending to use only minimal first-principles reasoning. Moreover, most of today's CBR systems also tend to rely on feature-based descriptions that cannot

match the expressive power of predicate calculus. Those relatively few CBR systems that rely on more expressive representations tend to use domain-specific and task-specific similarity metrics. This can be fine for a specific application, but being able to exploit similarity computations that are more like what people do could make such systems even more useful, since they will be more understandable to their human partners. While many useful application systems can be built with purely first-principles reasoning [5] and with today's CBR technologies, integrating analogical processing with first-principles reasoning will bring us closer to the flexibility and power of human reasoning.

CBR enables to use in computer program such kind of problem solving that is usually used by people, i.e. a new task is solved by adapting previously accomplished solution.

In most case-based reasoning systems, cases are stored as named collections of facts in a memory. They are designed for a specific range of problems. Each case is a set of features, or attribute-value pairs, that encode the context in which the ambiguity was encountered. The case retrieval algorithm is mostly a simple k-nearest neighbors algorithm. The basic case-based learning algorithm performs poorly when cases contain many irrelevant attributes. Unfortunately, deciding which features are important for a particular learning task is difficult.

At the highest level of generality, a general CBR cycle may be described [1] by the following four processes:

- **Retrieve** the most similar case or cases,
- **Reuse** the information and knowledge in that case to solve the problem,
- **Revise** the proposed solution,
- **Retain** the parts of this experience to be useful for future problem solving.

3.2 Reasoning in Decision Support System

A described agent-based decision support system MARABU [9] consists of: (1) database of modeling and control methods and tools, (2) case base of models created and simulated in past, (3) knowledge base of the control theory domain, (4) web-portal enabling to specify user requirements, to display results of reasoning, and to connect a provider of a selected tool. In the system, there are three possibilities of reasoning:

- Classical database querying: Models, control methods and tools are searched in database according to user specified requirements and given context.
- Case-based reasoning: Whenever no model or control method matches exactly the user requirements, the model forms and control methods are reasoned from similar cases.

- List of similar cases to the user specifications and requirements: If the user requires support in a form of accomplished similar case, similar cases with references to a tool where simulation can be done are provided (useful for e-learning purposes).

Attributes used in questionnaires to systems specification are weighted by real number 0 - 1. Each attribute's weight is initially estimated according to importance and classification power of a given feature.

Case-based reasoning algorithm proposed for system MARABU works in following steps:

1) An accomplished (history) case from case-base is searched that am best fulfils user specifications and requirements. Similarity rate of the history cases to the actual user specified task is counted using attributes weights.

2) Model and/or control method used in the history case with maximal similarity rate are offered to user to solution of actually specified task.

3) After reasoning, the user agent sends information to the selected tool provider's agent. Using the tool, simulation of a created model can be provided to the user. After simulation, the user can answer to the decision support system by filling in a form, whether his/her requirements were fulfilled.

4a) Automatic learning of weights uses this answer to increase the quality of case-based reasoning. The proposed learning algorithm guarantees convergence of the attributes weights to relatively steady values yielding to CBR of best quality.

4b) Some of new cases after a validation step are added to the case-base.

4 Learning Algorithm

Important feature of case-based reasoning is its coupling to learning [10]. The notion of case-based reasoning does not only denote a particular reasoning method, it also denotes a machine learning paradigm that enables sustained learning by updating the case base after a problem has been solved. Learning in CBR occurs as a natural by-product of problem solving. When a problem is successfully solved, the experience is retained in order to solve similar problems in the future.

Case retainment is the process of incorporating what is useful to retain from the new problem-solving episode into the existing knowledge. The learning from success or failure of the proposed solution is triggered by the outcome of the evaluation and possible repair. It involves selecting which information from the

case to retain, in what form to retain it, how to index the case for later retrieval from similar problems, and how to integrate the new case in the memory structure. The tuning of existing indexes is an important part of CBR learning. Index strengths or importance for a particular case or solution are adjusted due to the success or failure of using the case to solve the input problem. For features that have been judged relevant for retrieving a successful case, the association with the case is strengthened, while it is weakened for features that lead to unsuccessful cases being retrieved. The weights are updated, based on feedback of success or failure.

One issue in similarity assessment [6] is how to determine the right features to compare. Decisions about which features are important are often based on explanations of feature relevance, but those explanations may be imperfect, leading to a need for robust similarity metrics that take the difficulties in specifying important features into account. Defining adequate similarity measures is one of the most difficult tasks when developing CBR applications. Unfortunately, only a limited number of techniques for supporting this task by using machine learning techniques have been developed up to now.

In described decision support system, attributes used to systems specification are weighted by real number 0 - 1 (i.e. 0% - 100%). Each attribute's weight is initially estimated according to importance and classification power of a given feature.

After reasoning, the user agent sends information to the selected tool provider's agent. Using the tool, simulation of a created model can be provided to the user. After simulation, the user can answer to the decision support system by filling in a form, whether his/her requirements were fulfilled or not (i.e. reasoning was successful or it failed).

Automatic learning of weights uses the answers of the users to increase the quality of case-based reasoning. The proposed learning algorithm guarantees convergence of the attributes weights to relatively steady values yielding to CBR of best quality. The attributes weights are modified in following way:

$$w_i = \begin{cases} w_i + K_{1,j} \text{ or } w_{i,max} & \text{if success and } u_i == a_i \\ w_i - K_{2,j} \text{ or } w_{i,min} & \text{if fail and } u_i == a_i \\ w_i - K_{1,j} \text{ or } w_{i,min} & \text{if success and } u_i \neq a_i \\ w_i + K_{2,j} \text{ or } w_{i,max} & \text{if fail and } u_i \neq a_i \end{cases}$$

where $i = 1, \ldots, n$

n … number of attributes used to specification of the system and user requirements

w_i … weight of i-th attribute, minimal and maximal values $w_{i,min} = 0.$, $w_{i,max} = 1.$

u_i … valuation of i-th attribute in user's specification of the system and user requirements

a_i … valuation of corresponding i-th attribute in a given case

learning cycles j = 0, … , ∞

initial and minimal values of coefficients $K_{1,0} = 0.2$, $K_{2,0} = 0.1$, $K_{1,\,min} = 0.01$, $K_{2,\,min} = 0.01$

$$K_{1,j+1} = \begin{cases} K_{1,j} * c_1 \\ K_{1,\,min} \end{cases} \qquad K_{2,j+1} = \begin{cases} K_{2,j} * c_2 \\ K_{2,\,min} \end{cases}$$

In described decision support system following values were used: $c_1 = c_2 = 0.9$

Figure 2 explaines the first part of learning algorithm (setting of initial weights of the attributes) and Figure 3 explaines the second part of learning of the attributes weights as a part of case-based reasoning. If simulation of a new case was successful from the point of view of the user, the new case with the attributes stored in database (session information) is added to case base of the system before deletion of given session information.

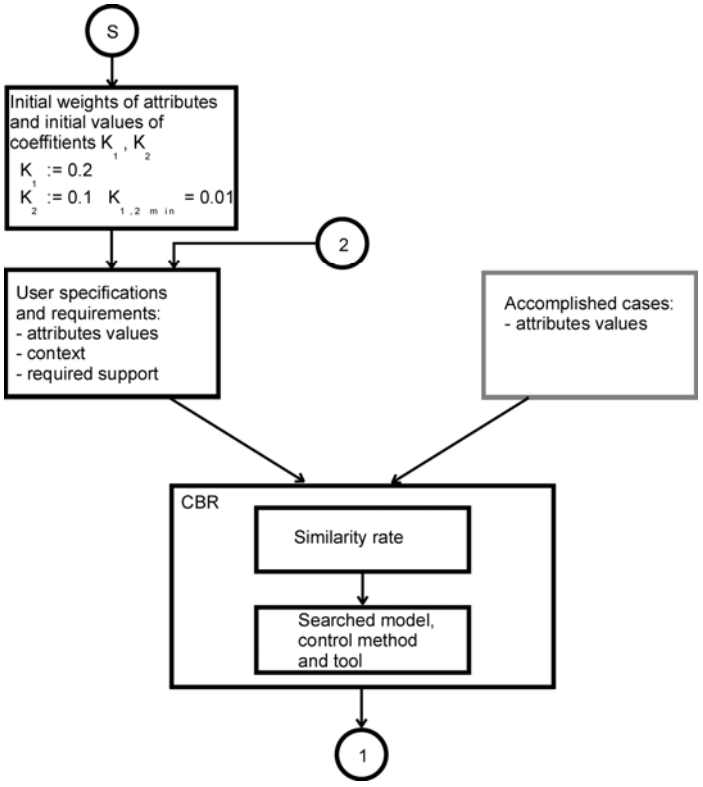


Figure 2
Learning of attributes weights (first part: initial settings)

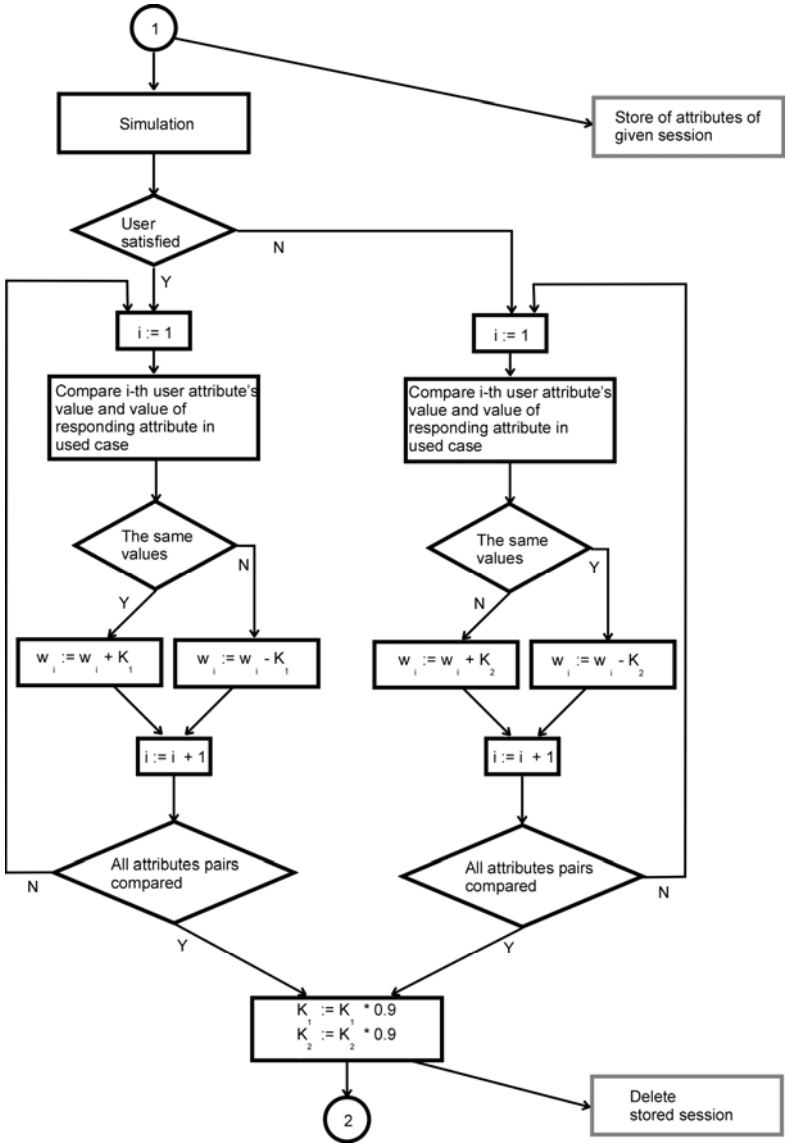


Figure 3
Learning of attributes weights (second part)

Conclusions

Decision support system for control theory domain has been described. The database of the system contains methods and tools for modeling and control synthesis as well as a set of complete models of systems specified by their attributes. Each attribute's weight is initially estimated according to importance

and classification power of a given feature. Automatic learning of attributes weights uses the answers of the users to increase the quality of case-based reasoning. The proposed learning algorithm guarantees convergence of the attributes weights to relatively steady values yielding to CBR of best quality.

Acknowledgement

The author is grateful to the Slovak Grant Agency for Science for partial support of this work - APVV Project No. LPP-0231-06, and VEGA Project No. 2/7101/27.

References

[1] Aamodt A., E. Plaza: Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, in Artificial Intelligence Communications, 1994, IOS Press, Vol. 7: 1, pp. 39-59, http://www.iiia.csic.es/

[2] Davis D. N.: Synthetic Agents: Synthetic Minds? Frontiers of Cognitive Agents, in IEEE Symposium on Systems, Man and Cybernetics, San Diego, 1998, IEEE Press

[3] Frankovič B., I. Budinská, J. Sebestyénová, Dang T. Tung, V. Oravec: MARABU - Multiagentový podporný systém pre modelovanie, riadenie a simuláciu dynamických systémov, in AT&P Journal 4, 2005, ISSN 1335-2237, pp. 57-59, in Slovak

[4] Heijst G.: The Role of Ontologies in Knowledge Engineering, PhD. Thesis, University of Amsterdam, Netherlands, 1995

[5] Kenneth D. Forbus, T. Mostek, R. Ferguson: An Analogy Ontology for Integrating Analogical Processing and First-Principles Reasoning, American Association for Artificial Intelligence, 2002, www.aaai.org

[6] Leake D. B.: Case-based Reasoning: Experiences, Lessons, and Future Directions, AAAI Press/MIT Press, 1996, www.cs.indiana.edu/~leake/papers/a-96-book.html

[7] Power D. J.: Decision Support Systems Hyperbook. Cedar Falls, IA: DSSResources.COM, HTML version, Fall 2000, http://dssresources.com/dssbook/

[8] Sebestyénová J.: Ontology-based Databases for Decision Support Systems, in WSEAS Trans. on Information Science and Applications, Issue 12, Vol. 2, December 2005, ISSN: 1790-0832, pp. 2184-2191

[9] Sebestyénová J.: Decision Support System for Modelling of Systems and Control Systems Design, in Proc. IEEE Int. Conf. on Computational Intelligence for Modelling Control and Automation CIMCA'2005, Ed. M. Mohammadian, Vol. 1, 28-30. Nov. 2005, Vienna, ISBN-13: 978-0-7695-2504-4, ISBN-10: 0-7695-2504-0, pp. 70-75

[10] Wettschereck D., D. W. Aha: Weighting Features. Proc. of the First Int. Conference on Case-based Reasoning, 1995, pp. 347-358, http://citeseer.ifi.unizh.ch/article/wettschereck95weighting.html