A Multiagent System for Monitoring Health

Leo van Moergestel, Brian van der Bijl,
Erik Puik, Daniël Telgen
Department of Computer science
HU Utrecht University of Applied Sciences
Utrecht, the Netherlands
Email: leo.vanmoergestel@hu.nl

John-Jules Meyer
Intelligent systems group
Utrecht University
Utrecht, the Netherlands
Alan Turing Institute Almere, The Netherlands
Email: J.J.C.Meyer@uu.nl

***Abstract*—By using agent technology, a versatile and modular monitoring system can be built. In this paper, such a multiagent-based monitoring system will be described. The system can be trained to detect several conditions in combination and react accordingly. Because of the distributed nature of the system, the concept can be used in many situations, especially when combinations of different sensor inputs are used. Another advantage of the approach presented in this paper is the fact that every monitoring system can be adapted to specific situations. As a case-study, a health monitoring system will be presented.**

Keywords-Multiagent-based health monitoring; learning agent.

I. INTRODUCTION

Monitoring systems are widely used in many situations. Simple systems collect information that can be inspected by humans or other systems. More advanced systems have the capability to react on the data monitored. Smoke detecting systems with an alarm are examples of these systems. Often a situation arises where more than one monitored condition should be taken into account before an action should be performed. Industrial production systems are examples of complicated situations where many sensors are used to control the process [1]. Another example of a complicated situation is the health condition of the human body [2]. Here, alarm conditions may also depend on individual factors, necessitating for the monitoring system to be trained for the specific individual person.

This paper describes a modular agent-based system [3] that can be trained by a medical expert and can monitor the status of a person and react adequately on the conditions encountered. This system has been built using agent technology, resulting in a robust and versatile multiagent-based monitoring system. The concepts presented here can be used in other situations as well [4].

The rest of this paper is organised as follows: Section II will describe the concepts of our approach, the reason for choosing agent technology as well as the architecture of the multiagent system. The section is followed by Section III where the training system will be explained. This training aspect is an important aspect of the system and is treated in detail. The implementation and results are presented in Section IV. Related work will be discussed in Section V and a conclusion will end the paper.

II. AGENT-BASED MONITORING

The first part of this section will show the requirements and explain the use of agent technology, while the second part will focus on the multiagent architecture.

A. *Requirements and technology*

A monitoring system should be built to be capable to handle several input values in combination. Depending on the combined values of the inputs a specific action should be executed. The system should be trained to build a knowledge base and utilise known information to decide on a strategy to react to the current situation. This resulted in the following a list of requirements:

- the system should monitor different inputs simultaneously;
- it should be easy to add extra monitoring inputs;
- the system should be trained in an effective manner;
- the system should have a set of possibilities to react on certain conditions;
- different types of reaction should be possible depending on the input values.

As a case study, a system in the medical domain has been adopted, but the concepts presented here can easily be used in other domains as well.

For the realisation, agent technology has been used [3]. The reasons for choosing agent technology are:

- error resistant. By using separate agents, the failure of an agent responsible for sensor input will not bring down the whole monitoring system. There is now a possibility to fall back on a different solution based on the availablity of sensor inputs.
- clear separation of responsibilities and goals. In our design, the sensors will be tied to separate agents that have a clearly defined goal. This is also true for the

 ISBN: 978-1-61208-518-0

other agents involved, as will be discussed in the next subsection.

- modularity. A multiagent system (MAS) is modular by nature and can be easily expanded with new features and possibilities.

B. MAS design

The agents involved have roles and responsibilities. When the different responsibilities are taken into account this will result in the architecture of a multiagent system as depicted in Fig. 1.

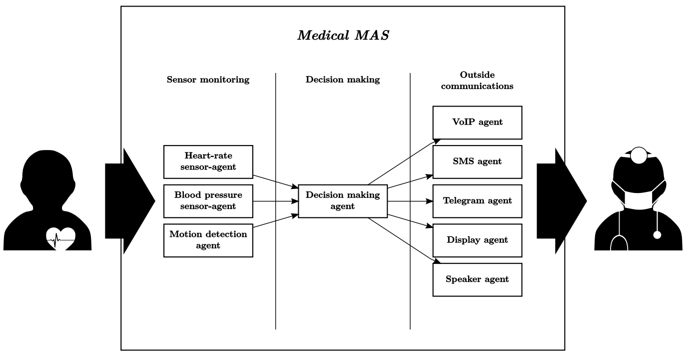

Figure 1. Medical MAS architecture

Three different roles are incorporated in the design resulting in three types of agents.

1) monitoring agents: Monitoring agents are responsible for delivering data to the decision agent. The data is coming from sensors. The agents themselves have a rather simple design. It could be possible to tell the agent at what intervals the data should be presented as well as the format expected by the decision agent.

2) decision making agent: A central role in the system is played by the decision agent. This agent decides what action should be performed under what conditions given by the monitoring agents. To do so, it has to be trained to build a knowledge base on how to react on certain conditions. This training has to be supervised by an expert, in our case a medical expert. A data acquisition system has been developed to help the expert to efficiently add data to the knowledge base. That system will be explained in the next section.

3) communication agent: The system has a set of communication agents that are responsible to communicate with the outside world. These agents are used by the decision agent to send emails, messages for several communication systems, like SMS, and also putting information on a display or generating an audible alarm.

III. DATA AQUISITION FOR AGENT TRAINING

In order to interpret the measurements acquired from the sensors and predict whether the current patient situation constitutes a cause of alarm, the decision agent needs a way to classify potentially high-dimensional data. Each biological factor considered in the model represents an additional dimension for data points. As this information is not guaranteed to be available for various combinations of biological features, it makes sense to explore a way for medical personnel to easily enter such data into the system. Not only does this guarantee the required data can be generated, if not available, it also allows for far greater personalisation, providing the agent with a data-set tailored to its patient. Manual entry, or at least confirmation, also allows an expert intimate knowledge of the agents decision-making process, potentially increasing trust by removing the “black box” aspect of machine learning.

Teaching the system to recognise alarming measurements and differentiate between various levels of threat requires large amounts of information provided by medical personnel, preferably tailored to the patient as thresholds might not be the same for every person. Entering this data can be challenging: as potentially multiple factors need to be taken into account together, it becomes progressively harder for humans to visualise and communicate relevant thresholds. A better way might be to input a set of data-points, together with appropriate assessments of the situation associated with each data-point. These data-points could be used, alone or in conjunction with more general datasets, to train a classification algorithm.

In order to train an agent to make accurate predictions, training data will need to be entered into the system by a medical expert. This should be as easy as possible: the focus should be to quickly train an agent without expending significant time accommodating the system. Unfortunately, entering possibly poly-dimensional data graphically is a difficult task. For one or two dimensional data, clicking points in a scatter plot, as pictured in Fig. 2, can be a quick way to enter points; for three dimensional data this becomes harder: a scatter-plot is still possible for data-visualisation, but entry becomes impossible as a mouse or trackpad and a computer screen are both essentially two-dimensional. For even more simultaneous features, only a subset of the features can be plotted at the same time.

An alternative approach would be to require the expert to manually enter all features, as well as the results that the system should predict. Not only is this rather work-intensive,

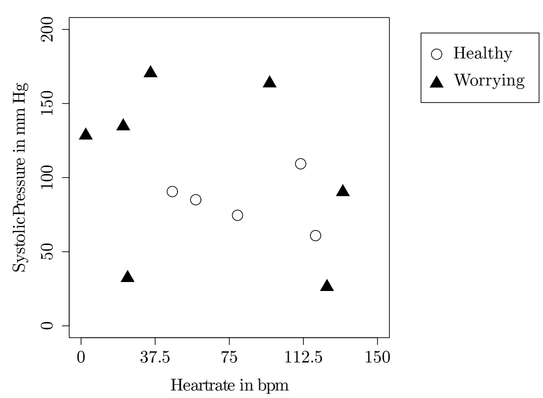

Figure 2. Scatter plot in two dimensions of a small random dataset.

 ISBN: 978-1-61208-518-0

but also prone to omissions: as it is hard for the human mind to visualise all features simultaneously and large gaps are a significant risk.

A better solution would be for the system to dynamically suggest data-points based on the largest knowledge gaps. An expert would then be provided with the parameters for a new datapoint by the algorithm. For this datapoint an assessment of the situation can then be entered. The algorithm continuously updates its collection of datapoints, as well as the model derived from the combination of datapoints and expert assessments, and proceeds to suggest the largest empty areas in its knowledge-continuum as possible locations for new datapoints. This continues until the expert considers the fit of the model to be satisfactory, after which the model is accepted. The expert remains in control of the process of entering datapoints, and can at any time ignore a suggestion or opt to enter the parameters for a new datapoint themself.

This section considers an approach to accomplish this. Each problem will be examined in two dimensions first, as this makes it easier to visualise and demonstrate the applied methods. After the solution has been sufficiently exposed a generalisation can be made in n-dimensions.

To represent gaps in the knowledge-continuum, we create a triangulation of the known datapoints. Each datapoint is considered a vertex in an n-dimensional space, and by triangulating over this set of vertices we can detect sparsely populated areas by the emergence of larger triangles. In contrast, a large amount of datapoints in close proximity will yield a large number of smaller triangles.

Triangles and their higher-dimensional analogues (the tetrahedron in three dimensions, the 5-cell in four, etc.) are collectively referred to as n-simplex or just simplices (singular: simplex). As a triangle (2-simplex) is defined by three vertices of the form (x, y) and and a tetrahedron (3-simplex) is defined by four vertices of the form (x, y, z), an n-simplex is the most basic n-dimensional object defined by $n + 1$ vertices in n-dimensional space.

A. *Finding the most valuable points for data-querying*

When entering data-points to train an agent, some points are more valuable than others. For example, potential locations completely surrounded by existing data-points all belonging to the same class are unlikely to add any new information to the system. Similarly, points in sparse areas are potentially more valuable, as are points closer to the centre of the point cloud. Fig. 3 shows the same scatter plot as Fig. 2, but adds a decision boundary and three possible locations for new data points marked by numbers. Location 1 does not appear to be a good addition, as it is very close to existing points and is therefore unlikely to add a great deal of information. Location 2 is not a good suggestion either, as it is very far from the decision boundary — it will likely have the same category as the points surrounding it, especially if a large amount of data has been entered. Location 3 is a better spot for a new data point: it is not a near duplicate of another point, and it lies close to the decision boundary. Depending on the category this point will be assigned to it may significantly change the decision boundary in either direction.

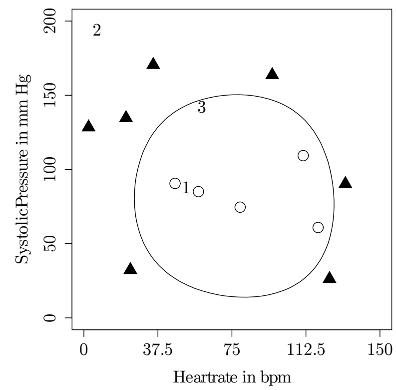

Figure 3. Three possible locations for a new data-point.

B. *Data-point-distribution*

To find sparsely populated areas to add new data-points, we first create a triangulation containing all data points. For each of these triangles, the circumcentre is calculated, and the collection is ordered based on the area of the triangles. These points can now be evaluated in order to find points close to the current decision-boundary.

C. *Triangulating n-dimensional space in simplices*

To triangulate a set of points we utilise the Delaunay Triangulation [5]. Most mathematical libraries include a function to quickly get the Delaunay Triangulation of a set of points in n dimensions. Triangulating the example data from Fig. 2 yields the triangulation as shown in Fig. 4.

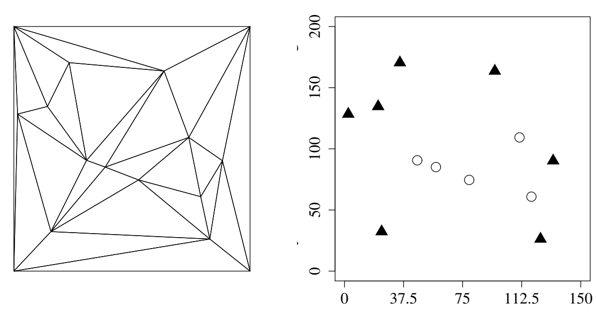

Figure 4. Triangulation and scatter plot in two dimensions

D. *Calculating the size of each n-simplex*

To find the largest simplex we use the determinant of the matrix constructed by adding each vector representing a vertex as a single column, and adding a final row of ones [6]. For a triangle, the absolute value of the result is equal to two factorial times the triangle's area. For a tetrahedron, the absolute value equals three factorial times the volume. For higher-dimensional shapes, this method continues to yield a scalar multiple of the n-hypervolume of the simplex. As the simplex size is only used for sorting, the scalar multiplication does not influence the ordering and can safely be ignored. As an example, the size of a triangle described by $a = (0, 0)$, $b = (0, 4)$ and $c = (3, 0)$ is given by

$$\text{abs}\left(\begin{vmatrix} 0 & 0 & 3 \\ 0 & 4 & 0 \\ 1 & 1 & 1 \end{vmatrix}\right) = 12 \tag{1}$$

which is twice the area of the triangle.

E. Calculating the circumcentre of each n-simplex

Once the largest data-gap has been found, we want to find its centre to suggest as a new data point. A simplex has multiple definitions of its centre; for this purpose the circumcentre, the point equidistant from all its vertices [7], seems a logical choice. Given a n-simplex defined by vertex $\boldsymbol{v^{(1)}}, \boldsymbol{v^{(2)}}, \dots \boldsymbol{v^{(n+1)}}$ with a circumcentre c, we know that the distance between any vertex and c must, by definition, be equal. For any two vertices $\boldsymbol{v^{(a)}}$ and $\boldsymbol{v^{(b)}}$, this means:

$$\begin{gathered} \|\boldsymbol{v^{(a)}} - \boldsymbol{c}\| = \|\boldsymbol{v^{(b)}} - \boldsymbol{c}\| \\ \|\boldsymbol{v^{(a)}} - \boldsymbol{c}\|^2 = \|\boldsymbol{v^{(b)}} - \boldsymbol{c}\|^2 \\ (\boldsymbol{v^{(a)}} - \boldsymbol{c}) \cdot (\boldsymbol{v^{(a)}} - \boldsymbol{c}) = (\boldsymbol{v^{(b)}} - \boldsymbol{c}) \cdot (\boldsymbol{v^{(b)}} - \boldsymbol{c}) \end{gathered} \tag{2}$$

We translate each vector by $-\boldsymbol{v^{(1)}}$ so that $\boldsymbol{v^{(1)}}$ becomes the origin (denoted $\boldsymbol{o}$) and equate the distance to $\boldsymbol{c}$ of each remaining vector with the distance of $\boldsymbol{c}$ to $\boldsymbol{o}$, yielding the locus for each translated vertex $\boldsymbol{v}$ and the origin $\boldsymbol{o}$:

$$\begin{gathered} (\boldsymbol{o} - \boldsymbol{c}) \cdot (\boldsymbol{o} - \boldsymbol{c}) = (\boldsymbol{v} - \boldsymbol{c}) \cdot (\boldsymbol{v} - \boldsymbol{c}) \\ \boldsymbol{c}^2 = \boldsymbol{v}^2 - 2\boldsymbol{v} \cdot \boldsymbol{c} + \boldsymbol{c}^2 \\ 2\boldsymbol{v} \cdot \boldsymbol{c} = \boldsymbol{v}^2 \\ \boldsymbol{v} \cdot \boldsymbol{c} = 0.5\boldsymbol{v}^2 \\ \boldsymbol{v}_1\boldsymbol{c}_1 + \boldsymbol{v}_2\boldsymbol{c}_2 + \dots + \boldsymbol{v}_n\boldsymbol{c}n = 0.5\|v\|^2 \end{gathered} \tag{3}$$

Doing this for every vertex $\boldsymbol{v^{(2)}}$ to $\boldsymbol{v^{(n+1)}}$ gives us n equations, allowing us to find the n-dimensional vector $\boldsymbol{c}$. We can write these equations in matrix form and solve all equations simultaneously:

Writing

$$S = \begin{pmatrix} v_1^{(2)} - v_1^{(1)} & v_1^{(2)} - v_1^{(1)} & \dots & v_1^{(2)} - v_1^{(1)} \\ v_2^{(3)} - v_2^{(1)} & v_2^{(3)} - v_2^{(1)} & \dots & v_2^{(3)} - v_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ v_n^{(n+1)} - v_n^{(1)} & v_n^{(n+1)} - v_n^{(1)} & \dots & v_n^{(n+1)} - v_n^{(1)} \end{pmatrix}$$

$$\boldsymbol{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \quad \boldsymbol{r} = 0.5 \begin{pmatrix} \|v^{(2)} - v^{(1)}\|^2 \\ \|v^{(3)} - v^{(1)}\|^2 \\ \vdots \\ \|v^{(n+1)} - v^{(1)}\|^2 \end{pmatrix} , \tag{4}$$

we have

$$S\boldsymbol{c} = \boldsymbol{r} \ . \tag{5}$$

Given this, we can multiply both sides by S^{-1} to get

$$\boldsymbol{c} = S^{-1}\boldsymbol{r} \ . \tag{6}$$

As c was translated by $-v^{(1)}$, all that remains is adding $v^{(1)}$ to find the triangle's circumcentre.

F. Avoiding suggesting out-of-bounds points

As shown in Fig. 4, Delaunay triangulations are prone to yielding obtuse simplices, in particular around the edges. This can be a problem because an obtuse simplex has a circumcentre outside itself. On the edges, this will result in the algorithm suggesting points outside the sensor's bounds. As these points are meaningless and only serve to distract the user, we would like to avoid generating obtuse simplices.

We solve this problem by introducing a border of false data-points around the edge. These data-points are only used to determine the Delaunay triangulation, and are not present in the actual training-data being generated. The number of data-points is determined by a variable $\beta \in \mathbb{N}_1$: For $\beta = 1$, only the corners of the graph are added. For larger values of β, each axis is subdivided into β parts. As β becomes larger, out-of-bounds points become increasingly unlikely, and suggestions start to gravitate towards existing data-points.

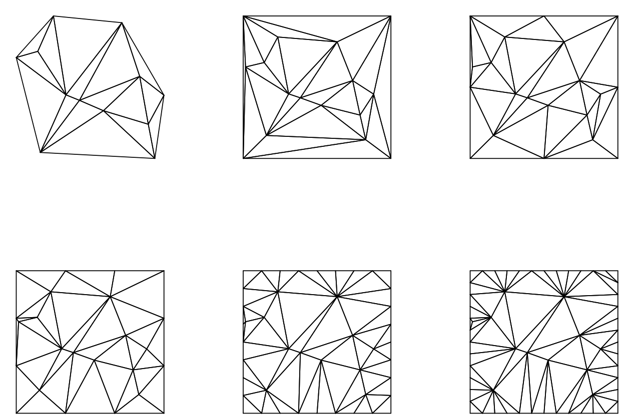

Figure 5. Triangulation for $\beta \in \{1, 2, 3, 8, 12\}$ alongside original triangulation.

As Fig. 5 and Fig. 6 show, too large a value for β makes the algorithm increasingly unlikely to suggest points around the edges. Though more central points are preferred, limiting data-points to a central cluster might not be the way to go. A solution for this could be to gradually decrease β over time.

G. Generating the borders

The set of points to be used as a border constitutes of the following:

- a point for each vertex of the n-cube describing the range of data-points

 ISBN: 978-1-61208-518-0

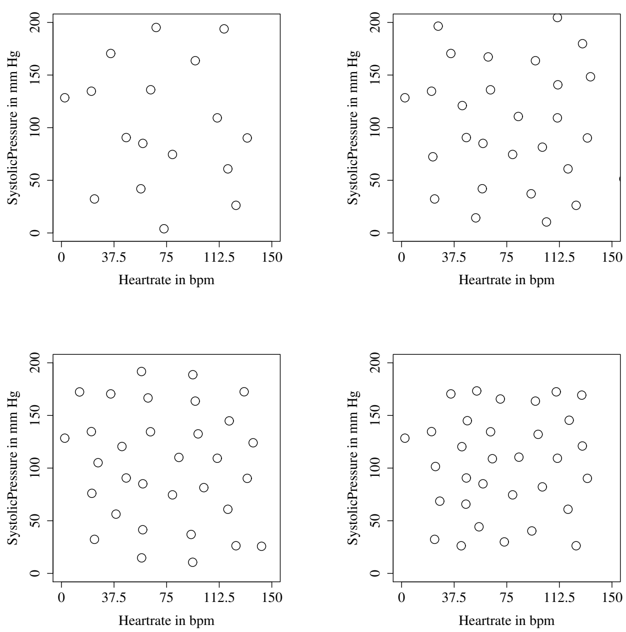

Figure 6. Scatter plot of the first twenty suggestions for $\beta \in \{1, 2, 4, 8\}$. Note that out-of-bounds points are not plotted.

- $(\beta - 1)$ points on each edge (1-face)
- $(\beta - 1)^2$ points on each face (2-face)
- $(\beta - 1)^3$ points on each cell (3-face)
- ...
- $(\beta - 1)^{n-1}$ points on each $(n - 1)$-face

The number of points denoted by $\#P$ needed given a dimensionality n and a border-saturation β can therefore be calculated by

$$\#P(n, \beta) = \sum_{i=0}^{n-1} F(n, i)(\beta - 1)^i \tag{7}$$

where $F(n, i)$ is the number of i-faces on a n-cube [8]:

$$F(n, i) = 2^{n-i}\binom{n}{i} \tag{8}$$

The actual value of $P(n, \beta)$ can intuitively be seen as the Cartesian product of n instances of interval(β), also known as its Cartesian Power, of which only those points for which at least one of its members is equal to -1 or 1 are kept. In other words, for which the infinity norm $\|x\|_\infty$ equals 1.

$$P(n, \beta) = \{x \mid x \in \text{interval}(\beta)^n \wedge \|x\|_\infty = 1\} \tag{9}$$

$$\|x\|_\infty = \max_i |x_i| \tag{10}$$

H. Feature Scaling

The interval-function creates an interval between -1 and 1 in β steps. This is because all features are scaled to lie between -1 and 1, even though the actual measurements might range from 0 to some arbitrary maximum. This fscale is applied to make sure that all features are of the same importance when applying logit later on.

I. Avoiding symmetry

The algorithm presented above tends to favour generating a symmetrical data-set: As the range of values is a perfect n-cube, the first point suggested will be the centre, followed by a group of points equidistant from the first. This is undesirable, as symmetrical data points feature will introduce redundant features when multiplied during the fmap process. It will not help in generating a better hypothesis but will slow down the learning algorithm.

To prevent generating such a duplicate set of data, we will move each suggestion by a small random amount, controlled by a variable δ, that represents the maximal displacement for each point in each dimension. In order to ensure that this displacement will not place points outside the feature boundaries, this displacement will be opposite to the sign of the original location. This results in the data point being moved slightly towards the centre, which generally is the most interesting area to collect data on. We achieve this by replacing each vector element c_i by the weighted mean of $r \cdot 0$ and $(1 - r)c_i$, where $r \sim U([0, \delta])$ is a random variable uniformly distributed on $[0, \delta]$.

IV. Implementation

For the implementatation of the MAS, Java agent development framework (Jade) [9] has been used. The Jade runtime environment implements message-based communication between agents running on different platforms connected by a network. The reasons for choosing Jade are:

- the system presented is a multi-agent-based system. Jade provides the requirements for multiagent systems;
- the agent communication standard ”Foundation for Intelligent Physical Agents” (FIPA) [10] is included in Jade;
- Jade is Java-based and it has a low learning curve for Java programmers. Java is a versatile and powerful programming language;
- Jade is developed and supported by an active user community.

The prototype has been developed and implemented on a standard Linux-based laptop. It should be possible to operate the system on any small device capable of running Java such as the Raspberry Pi [11]. Though the Jade-platform was selected for the prototype, this does not preclude development of a medical MAS in another framework or language. The concepts explored here can be implemented in any language, though support for a solid agent-development framework would be a serious asset. Nevertheless, if better performance is needed, the same principles could be implemented in a lower-level language, such as C, reducing much of the overhead at the cost of lower maintainability.

The prototype has been built and the working has been tested. In summary the following results have been achieved:

- The concept of a medical MAS consisting of three types of agents working together to monitor the patient and communicate the result.
- A method of collecting data from medical experts and utilising this knowledge to teach an agent to evaluate readings provided by sensors.
- The beginnings of a generalised framework upon which to build agents for inclusion in a medical MAS.

V. Related work

Agent-based monitoring for computer networks has been proposed and implemented by Burgess. Burgess [12] [13] describes Cfengine that uses agent technology in monitoring computer systems and ICT network infrastructure. In Cfengine, agents will monitor the status and health of software parts of a complex network infrastructure. In [14], an agent-based monitoring system is proposed. A so-called product agent is responsible to monitor the working of a system in several different phases of its lifecycle. The actions performed by the agent are limited to prevent disasters or misuse. The aforementioned concept of a product agent that supports a product during its lifecycle from production to recycling is described in [15].

A lot of literature is available regarding health monitoring systems. Pantelopoulos and Bourbakis [16] give an overview of wearable sensor-based systems for health monitoring and prognosis. Their work focusses on the hardware implementation of the monitoring systems as well as communication technologies that might be used by such systems. The work of Milenkovic [17] is dedicated to wireless sensor networks in personal health monitoring. The system they describe collects data that is transferred to a central monitoring system whereas the system described in our paper aims for autonomous operation. Furthermore, monitoring systems that focus on special health related situations exist, such as the work of Marder e.a [18] where a system for monitoring patients with schizophrenia is described. An agent-based health monitoring as a concept for application of agent technology has been proposed by Jennings and Wooldridge in [19].

VI. Conclusion

In this paper, a complex, expandable and agent-based monitoring system has been proposed and a proof of concept was built. The system turned out to work as expected. Special attention has been given to the way the system builds its knowledge-base, resulting in an efficient system that focusses on the borders of operating space where transitions from one situation to another situation are possible. In the case of the medical monitoring system, this could result in a personal adapted monitoring system that can also be easily changed. Though the system is designed for use in a medical context, the concepts can be used in other domains as well.

References

[1] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, "A versatile agile agent-based infrastructure for hybrid production environments," IFAC Modeling in Manufacturing proceedings, Saint Petersburg, pp. 210–215, 2013.

[2] J. T. Parer and T. Ikeda, "A framework for standardized management of intrapartum fetal heart rate patterns," American Journal of Obstetrics and Gynecology, vol. 197, no. 1, pp. 26.e1 – 26.e6, 2007.

[3] M. Wooldridge, An Introduction to MultiAgent Systems, Second Edition. Sussex, UK: Wiley, 2009.

[4] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, "Embedded autonomous agents in products supporting repair and recycling," Proceedings of the International Symposium on Autonomous Distributed Systems (ISADS 2013) Mexico City, pp. 67–74, 2013.

[5] Wolfram MathWorld. Delaunay Triangulation. [Online]. Available: http://mathworld.wolfram.com/DelaunayTriangulation.html [retrieved: april, 2016]

[6] P. Stein, "A note on the volume of a simplex," The American Mathematical Monthly, vol. 73, no. 3, pp. 299–301, 1966. [Online]. Available: http://www.jstor.org/stable/2315353 [retrieved: april, 2016]

[7] Wolfram MathWorld. Circumcenter. [Online]. Available: http://mathworld.wolfram.com/Circumcenter.html [retrieved: april, 2016]

[8] R. J. McCann, "Cube face," 2010. [Online]. Available: http://www.math.toronto.edu/mccann/assignments/199S/cubeface.pdf [retrieved: march, 2016]

[9] Telecom Italia. JAVA Agent DEvelopment Framework. [Online]. Available: http://jade.tilab.com/ [retrieved: januari, 2016]

[10] Foundation for Intelligent Physical Agents. FIPA. [Online]. Available: http://www.fipa.org/ [retrieved: januari, 2016]

[11] E. Upton. Oracle Java on Raspberry Pi. [Online]. Available: https://www.raspberrypi.org/blog/oracle-java-on-raspberry-pi/ [retrieved: april, 2016]

[12] M. Burgess, "Cfengine as a component of computer immune-systems,," Proceedings of the Norwegian Informatics Conference, pp. 283–298, 1998.

[13] M. Burgess, H. Hagerud, S. Straumnes, and T. Reitan, "Measuring system normality," ACM Transactions on Computer Systems (TOCS) Volume 20 Issue 2, pp. 125–160, 2002.

[14] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, "Monitoring agents in complex products enhancing a discovery robot with an agent for monitoring, maintenance and disaster prevention," ICAART 2013 proceedings, vol. 2, pp. 5–13, 2013.

[15] L. v. Moergestel, J.-J. Meyer, E. Puik, and D. Telgen, "The role of agents in the lifecycle of a product," CMD 2010 proceedings, pp. 28–32, 2010.

[16] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 40, no. 1, pp. 1–12, 2010.

[17] A. Milenković, C. Otto, and E. Jovanov, "Wireless sensor networks for personal health monitoring: Issues and an implementation," Computer communications, vol. 29, no. 13, pp. 2521–2533, 2006.

[18] S. R. Marder, S. M. Essock, A. L. Miller, R. W. Buchanan, D. E. Casey, J. M. Davis, J. M. Kane, J. A. Lieberman, N. R. Schooler, N. Covell et al., "Physical health monitoring of patients with schizophrenia," American Journal of Psychiatry, vol. 161, no. 8, pp. 1334–1349, 2004.

[19] N. R. Jennings and M. Wooldridge, "Applications of intelligent agents," in Agent technology. Springer, pp. 3–28, 1998.