robotics

MDPI

Article

Long-Term Adaptivity in Distributed Intelligent Systems: Study of ViaBots in a Simulated Environment

Arturs Ardavs, Mara Pudane *, Egons Lavendelis and Agris Nikitenko

Department of Artificial Intelligence and Systems Engineering, Riga Technical University, LV-1658 Riga, Latvia; arturs.ardavs@rtu.lv (A.A.); egons.lavendelis@rtu.lv (E.L.); agris.nikitenko@rtu.lv (A.N.)

* Correspondence: mara.pudane@rtu.lv

Received: 28 February 2019; Accepted: 26 March 2019; Published: 29 March 2019

Abstract: This paper proposes a long-term adaptive distributed intelligent systems model which combines an organization theory and multi-agent paradigm—ViaBots. Currently, the need for adaptivity in autonomous intelligent systems becomes crucial due to the increase in the complexity and diversity of the tasks that autonomous robots are employed for. To deal with the design complexity of such systems within the ViaBots model, each part of the modeled system is designed as an autonomous agent and the entire model, as a multi-agent system. Based on the viable system model, which is widely used to ensure viability, (i.e., long-term autonomy of organizations), the ViaBots model defines the necessary roles a system must fulfill to be capable to adapt both to changes in its environment (like changes in the task) and changes within the system itself (like availability of a particular robot). Along with static role assignments, ViaBots propose a mechanism for role transition from one agent to another as one of the key elements of long term adaptivity. The model has been validated in a simulated environment using an example of a conveyor system. The simulated model enabled the multi-robot system to adapt to the quantity and characteristics of the available robots, as well as to the changes in the parts to be processed by the system.

Keywords: multi-robot systems; adaptivity; multi-agent systems; viable systems model

1. Introduction

Today robots play an important role in modern manufacturing, service domains, and households providing behaviors of significant complexity. The current mechatronics approach used in the design of modern robotic systems has proven to be sufficient to automate the routine processes. However, the requirement for robotic systems that can complete increasingly complex tasks is growing rapidly. This need calls for a change in how such tasks are performed; in 2016, it was identified that there was a need for novel design principles to make robots more intelligent, adaptive, and capable to collaborate [1].

The fact is that currently widespread robotic systems are capable to do only pre-programmed tasks, thus lacking adaptivity at the level required by the trend in the modern use of robotic systems. While many specific tasks can be indeed completed at a sufficient quality based on predetermined instructions and a system's configuration, such systems still require human intervention when changes occur either in the external environment or in the system itself. A good example that illustrates this problem is commercially-used feed pushing robots for cattle farms. Such robots are only capable of following the pre-programmed path; if for some reason they stray off the trajectory, a human worker must move the robot back manually [2]. Another example is fully automated production lines in manufacturing plants; these systems can only do the work if the task does not change and the system itself is fully functional. If something breaks down, it is not possible for the system to continue its

operation even if factually it is capable to carry on. Moreover, usually even insignificant changes in that manufacturing process require full manual reconfiguration to continue the production [3].

To address these issues, much effort has been put into building increasingly intelligent autonomous systems that would be more adaptive. One of the approaches used during the last years is neural network-based machine learning to acquire new knowledge for adapting to environmental changes [4]. Significant success has been achieved in specific domains, such as game playing [5] and autonomous driving [6]. Still, for now, the ability to learn general models is limited to one domain, or to a single machine (e.g., in case of autonomous cars); neural network-based approaches cannot explain the reasoning behind their decisions.

From a practical perspective, a conditioning control system for all possible scenarios is not considered as a viable solution: The designer simply cannot foresee and design all the possible situations that may occur during the lifetime of a system. It is particularly impractical and nearly impossible in a system with a large number of components and multiple hierarchy layers, such as a multi-robot system. Although there are certain challenges, multi-robot systems in general are considered to have a high potential for future manufacturing plants. The main expected advantage is possible synergy among teaming robots while a single large system might be less effective or too costly. One highly anticipated example of the cooperation is the following: In case of malfunctioning of a single robot within the team, other team members can rearrange the tasks among themselves to continue production until the robot is available again. While the obvious benefits are desired by the robotics and manufacturing communities, there are a number of challenges to develop such systems arising from the fact that multi-robot systems can have innumerable different configurations depending on type and number of robots within the teams. A multi-agent paradigm provides mechanisms for standard situations, for example, Contract-Net protocol for allocating an isolated task to the best performer [7]. Still, many other mechanisms for allocating multiple tasks, identifying the need to reallocate tasks, etc., are necessary to ensure viability. Implementation of such capabilities requires novel design approaches that manage this complexity.

Based on all these considerations, one can conclude that to enable the full potential of autonomous robot-based solutions, multi-robot systems need capabilities to autonomously adapt to changes in the external environment and system itself; this adaptivity must also be foreseen in the design phase of the system. This is a high bar to achieve in robotic systems, yet in the system theory, in particular, organization theory, where such systems have been researched considerably longer, this challenge has already been tackled. A viable system model (VSM) has been developed to specify the functions that must be implemented within human organizations to make them successful in the long term, (i.e., viable) [8]. Human organizations and multi-robot systems (just as any system) are viewed as similar from the system theory viewpoint. This leads to the hypothesis that system theory models that refer to organizations can be adapted to technical systems providing benefits that have been available in human organizations for decades.

The paper adapts the VSM to multi-robot systems in combination with novel task allocation mechanisms among robots to enable long-term autonomy viability. The developed model has been tested to ensure adaptivity at a case study of a conveyor system where autonomous robots must process parts arriving via a conveyor belt. The experiments have been carried out within a simulated environment where each robot has been simulated by an intelligent agent.

The remainder of the paper is organized as follows. Section 2 describes the related works in two parts: First, the current approaches that can be used to develop adaptive autonomous systems, and second, the domains where the VSM has been applied. Section 3 describes the ViaBots model and VSM adaptation to robotic systems. Section 4 describes the application of the model to a conveyor belt case study in a simulated environment. Section 5 analyzes the results achieved by the simulations. Section 6 discusses the limitations of the ViaBots model and possible challenges in its implementation for other scenarios. Section 7 concludes the paper and outlines future works.

2. Related Works

The related research can be viewed from two perspectives: Studies and applications addressing long-term autonomy as well as cybernetics approach of long-term autonomy as a general feature of multi-robot systems.

Over the past decade, the research on autonomous multi-robot systems has come much closer to a point where truly cooperative robot-robot and robot-human teams can be developed. Some early approaches like Nerd Herds swarming or ALLIANCE subsumption architectures have been introduced, and are well suited for focused tasks of limited complexity. These approaches, however, are not designed for long-term autonomous operation [9,10].

To be fully autonomous, the system needs to address several specific issues. Both robot–robot and human-robot scenarios have been extensively studied by the research community, and as a result, certain challenges have been identified. While these include simpler, technology-related issues, such as hardware and software malfunctioning that leads to performance drops or function failures, or human-machine interaction failures due to improper system use [11], others are much more complex and relate to the cognitive behavior of the system as a whole. This includes, but is not limited to, improper priority selection in decision making, improper strategy selection, reduced trust, which leads to mission level failures, conflicting goals of team members that leads to reduced collaboration or even mission failures in case of human-machine scenarios [11]. In different application domains, these challenges are addressed using different methods. For space exploration applications, NASA has elaborated the Autonomous NanoTechnology Swarm (ANTS) concept, which is based on swarm intelligence [12] ideas and comprises robot teams with the following main features: Self-configuring, self-optimizing, self-healing and self-protecting [12]. The main enabling functionality in ANTS is awareness, which maintains a dedicated control loop for knowledge updates and checking for changes, thus comprising monitoring, recognition, assessment and learning capabilities. To ensure long-term resilience, ANTS awareness is split into self-awareness and context-awareness [12].

Some narrow robot team applications like formations and flocking have been in research concentrating on both environmental uncertainties, uncertain robot dynamics, and communications, which usually are wrongly considered as available. Unfortunately, in the long-term, autonomous operations all of the mentioned capabilities are challenging, therefore [13] proposed to use the extended Brain Emotional Learning Based Intelligent Controller (BELBIC) algorithm. Resilient BELBIC or R-BELBIC is based on flexible biology-inspired intelligent control, in order to achieve a long-term operation [13]. Ref. [14] assumed instead that cooperativeness as a key element of resilience is not guaranteed, thus compromising the whole system. Therefore, a Weighted Mean Subsequence-Reduced (W-MSR) algorithm [15] was used to achieve consensus within the team at all times. Other applications like logistics in [16] or even single robot resilience in [17] have been studied extensively as well.

Unfortunately, while most of the studies concentrate on specific missions or applications of robot teams, only very few propose formal frameworks or methods of system design. One of them presented at [18] used formal design methods, such as Event-B and PRISM, to derive a technical design of the system through iterative steps and assessed the probability of goal achievement thereby providing guidance for further developments.

Overall, it can be concluded that the need for long-term autonomous robotic systems has emerged and rapidly grown over the past decade. The developments now are early and specific to narrow applications. We see an obvious gap of general frameworks enabling a long-term resilient multi-robot system design.

The question of long-term adaptivity and autonomy, however, is not distinctive only to technical systems; it has been one of the key themes in system theory and organizational theory for more than half a century. Initially, the research started with an empirical observation: Some enterprises and organizations are persistent for long periods of time while others cease to exist rather quickly. Research devoted to understanding this phenomenon resulted in the VSM—a framework which specifies the characteristics and functions that ensure long term survival, (i.e., viability) [8].

Primarily VSM has been intended to design an organization. However, it has been found that VSM can be useful not only in organizations, but also in the technical systems area due to the properties that are shared by organizations and technical systems (such as increasing complexity), and that it is at least theoretically adaptable to technical systems [19]. The challenges that modern autonomous multi-robot systems and organizations face are very similar. Both types of systems need to work autonomously in a changing environment with a high quantity of components and increasing internal complexity. Moreover, from the systems theory viewpoint, all systems are viewed as having a similar structure [20], and the field of operation is merely an application. This implies that the approaches that work cybernetically for non-technical systems, should also work in technical systems.

The two main aspects of the VSM are managing complexity and adapting to changes: Organizations, as a rule, have a high quantity of elements, lots of internal links on the inside, and a highly turbulent external environment on the outside [20]. This closely relates to the autonomous multi-robot systems that have high complexity as well. The adaptivity is ensured, and structural and functional complexity is managed and dissolved in the VSM by two main aspects: A well-defined functional structure that involves information channels among functions and the fractal nature of the model.

All systems, in particular, complex systems, consist of several to many abstraction levels. For the system to be viable, each of these layers must be able to adapt and do so in a similar fashion—this structural and functional hierarchy and repetitiveness on different hierarchy levels is called recursivity. In the organization, recursivity is carried out via departments and the general structure of the organization. When properly recognized, this property grants a crucial benefit—the system can continue to work even if some part of it malfunctions or is completely lost (such level of adaptivity characterizes living natural organisms, (i.e., human beings)). From the perspective of the design, the recursive approach is resource-effective since the same design can be applied to different abstraction levels enabling higher scalability [19]. In this sense, using a recursive model, such as VSM, for modeling multi-robot systems would be highly beneficial.

In addition to its recursive nature, VSM strictly defines functional blocks and links among these blocks thus defining the functions needed for long-term adaptivity. These functional groups are structured in a way that allows self-determination to components of the system, yet keeps them in touch. Such operational independence allows for a system to be less prone to fatal failures as well as gives freedom to achieve goals in alternative ways, thus ensuring adaptivity. The management in VSM is structurally de-centralized, (i.e., any of the system's components can perform management functions).

Application of the VSM in organizations provides better fault-tolerance, more effective and responsive management, and an overall structure that supports adaptivity of a system [19]. These benefits would also be crucial for the autonomy of technical systems; we propose to use a viable systems model [8] as a core framework for a multi-robot systems design.

The functional structure of VSM consists of five functional blocks that are often called subsystems 1, 2, 3, 4, and 5 [8]. They are however also denoted by their functionality, respectively operation, coordination, control, intelligence, and policy [21]. These groups are defined and explained in detail in the next section. The model also defines the links among these functional groups which are characterized from two perspectives: The structure of the channel itself, as well as the dynamics, (i.e., the data, information, or knowledge flowing). Stafford Beer, the author of the model, himself has defined the structure of the channels [20]; these channels have also been classified based on the semantics and type of flow [22].

Since the model primarily is focused on organizations, so far it has mostly been validated in organizational structures and related areas. Some examples include information and knowledge circulation modeling for organizations in their usual state [23] and in case of fluctuations (such as emergency or crisis scenarios) [24,25]. The goal of the modeling can be either to prove the organization is viable or more often, find flaws in the organization in case of missing functions or flows [21,26]; a key research area is managing complexity either in normal or crisis situations [27]. Arguably the

most massive validation of the model was carried out by establishing a VSM-based economy in Chile during the 1970s [28].

Increasingly often VSM is applied to the engineering (including IT) field, although mostly from the perspective of project management [29,30]. In [31], VSM was used in the design of large and complex IT projects with the aim to manage complexity. In general, VSM-based analysis is used in designing information systems since slow or non-existent knowledge and information flows can be enhanced or created with the help of information technology [23]. There has also been recent research on how VSM could be adapted to the design of technical systems [19]. More specific technical VSM-based designs include the design of smart distributed automation systems [32] and cyber security management [33].

It can be concluded that the idea of using a VSM for the design of complex technical systems has been around for some years, yet the model has not been fully validated in technical systems. We see two main reasons for that. First, the requirement for viability in technical systems has become crucial during the past decade with the increasing complexity of the systems themselves as well as the tasks given to the systems—to put it simply—there was no need to explore such models before this requirement emerged. That is why the existing models that mix VSM and technical systems are in an early development stage. Secondly, the fact that VSM is organization-oriented creates several ambiguities and issues open to interpretation when adapting the model to technical systems. While the adaptation of the overall organization of the system is straightforward, the content or semantics of separate functional blocks, as well as the model dynamics, needs to be formalized and validated. The next section explains VSM adaptation to technical domain.

3. ViaBots Model

Applying VSM for technical systems, including robotic systems, requires several interpretations and adaptations. This section describes the ViaBots model for viable multi-robot systems' implementation. First, the VSM and the necessary adaptation for robotic systems is discussed, followed by the description of the use of the intelligent agent paradigm within the model.

3.1. VSM as a Basis for Technical Systems

VSM serves as both the functional structure and the source of function's semantic meaning. For this reason, this subsection describes VSM in general and its subsystems in particular. While in general VSM is meant to be used in organizational systems, there have been some efforts to apply it in robotics; some examples of VSM function application to robotic systems are explained here.

The structure of the model can be roughly divided into operation and management. The functional blocks of the VSM mostly contain managerial functions (subsystems 2, 3, 4, and 5) and are based on the idea that the general structure of management is independent of the operation domain of the system [34]. Figure 1 represents the structure of VSM.

S1 is the operation of the system and represents divisions and departments in an organization. In a multi-robot system, depending on the abstraction level S1 functions can be carried out by individual robots or by robot actuators [19]. There are more than one S1 in each system; S1 is the part of the system that carries out the actual work. S1 also is a crucial part of the implementation of recursivity since each S1 is organized based on VSM one abstraction layer lower (see Figure 1).

Each S1 has system-level local management attached. The main function of it is coordination and it is the second subsystem, S2. S2 consists of a management unit that directly oversees each S1, and a coordination block that coordinates all the S1 activities. In the organization, these functions are carried out by department heads and their meetings, while in robotic systems the S2 functions are low-level control mechanisms or communication mechanisms among several robots [19].

The S1 and S2 are needed to carry out the system's purpose. However, it is not enough to create a viable system. Such a system is fragile to schisms in case S2 units are unable to agree on resource distribution. Moreover, the system cannot follow the same goal and only holds the view of the current situation without any regard for future states. To solve these issues, senior management consisting

of S3, S4, and S5 is introduced. S3 exerts control over S1 and S2. It compares the actual state to a desirable state and steers the system towards its goals by allocating resources. In the organization, there are several departments that carry out these functions, including the financial department as well as human resources. In a multi-robot system, planning algorithms are used here [19].

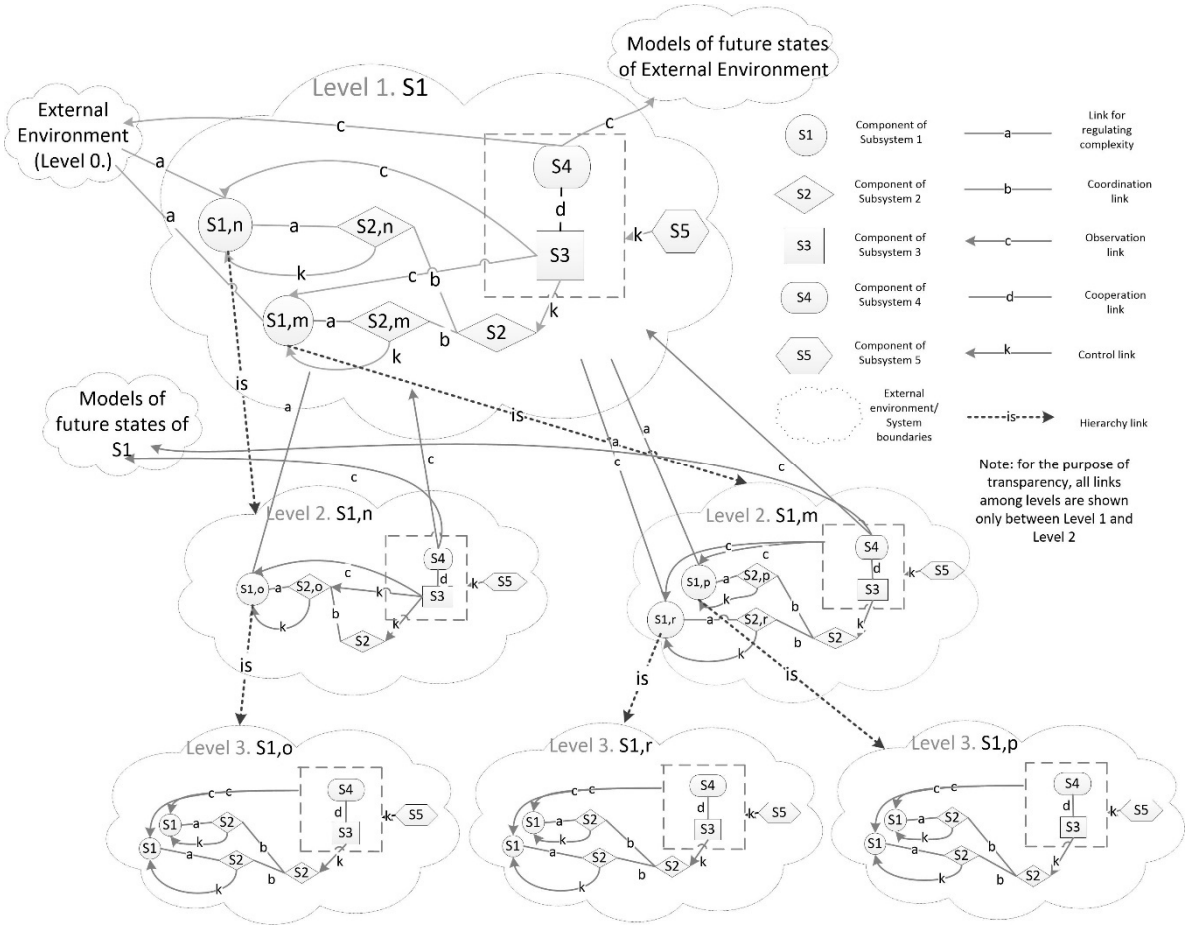

Figure 1. The structure and recursivity of a viable system model (VSM). Adapted from [19].

A system that carries out S1, S2, and S3 functions is a system that has a goal and ability to work towards it in an environment with minor fluctuations. However, such a system is unable to adapt to major changes and a turbulent external environment. To predict the future states of the external environment and the system itself, an intelligence function, or S4, is needed.

S4 tries to predict the future by applying learning and analysis of the current situation. In the case of an organization, S3 and S4 are often integrated. In robotic systems, this part must be carried out by artificial intelligence techniques (e.g., distributed adaptation algorithm) [19].

Finally, a viable system needs to be able to redefine its goals and policies since otherwise, the system is unable to adapt to drastic changes. S5, or policy function, monitors S3 and S4 as well as the external environment to adjust the policy. In an organization, S5 is executed via shareholders while in robotic systems a more dedicated planning algorithm might be needed [19]. Alternatively, the role of S5 can be partially executed by the human user by defining the current goal of the system.

Among these subsystems, there are a variety of links. Based on the information carried, frequency, and general qualities of channels these links are classified into five groups denoted by the letters a, b, c, d, and k [22] in Figure 1.

Type a links are bidirectional links used to connect functions that have different degrees of variety. In organizations, these are the channels via which the organization acquires the information needed for the operations and communicates the results back to the external environment. These links have a

specific construction: Attenuators on one end to reduce the number of states and an amplifier on the other to amplify the effects of the system in order to have an impact. In the case of robotic systems, these links enable interaction with the external environment via sensors and actuators by ensuring that a robot only perceives a finite number of external environment states. It is up to the designer to choose the appropriate physical architecture of the robots.

Type b links are specific to S2 and are used to coordinate (i.e., send content back and forth) several units with the aim to come to consensus between independent units; these links allow for the robots to communicate among themselves and share limited resources, including capabilities, physical restrictions for performing one, or the other task.

Type c links are monitoring links which are directed to the observer and allow for higher systems to make informed decisions. Type d link is an integration link—a bidirectional link between elements of the same variety that cooperate and allow overall functioning of the system by balancing the actual and future states. This means that the robots that fulfill functions connected with type d links need to be in constant communication. Finally, type k links are control links that are used to give the orders; the link itself serves as an amplifier and in the robotic system is used to pass commands.

3.2. Using a Multi-Agent System at the Logical Level of the System

Mobile robots comply with the notion of an intelligent agent that is characterized by interaction with the environment via perceiving inputs and influencing it through the outputs. Main characteristics of intelligent agents within multi-agent systems are autonomy, reactivity, and social capabilities. Robots forming a multi-robot system need all of them as well. For these reasons, we are using an intelligent agent paradigm at the design level by deeming robots as intelligent agents. Using agents at the logical level offers several additional benefits. One of them is the possibility to use already existing interaction mechanisms necessary to build collaborative multi-robot systems. Moreover, this enables using agent-based simulations during the model development to prove the concept before implementing the model on the actual multi-robot system.

Consequently, within the ViaBots model each agent represents a robot and agents, in turn, have functions corresponding to a VSM; within this approach, all the elements of the system (robots, user interface, any type of control algorithms that can run on hardware or software) are designed as agents. The percepts and actions of the agents are taken directly from the VSM links and defined by the design of the multi-agent system; the way how an agent processes the percepts and maps them to the actions, (i.e., the internal architecture of the agents can greatly vary based on the purpose of the system and the complexity of VSM functions). An agent-based approach at the logical level enables defining interconnections between a VSM and technical system implementation. The overall structure of the mappings is depicted in Figure 2. The concepts of the viable systems have been formulated in the abstract domain which is the VSM itself. At the logical domain, the ViaBots model has been designed based on general concepts of multi-agent systems. At this level, VSM concepts are tied to concepts associated with the multi-agent system domain. Finally, to practically implement the ViaBots model in a real development environment, the third mapping with concepts used at the implementation domain is needed. Such a mapping can be performed on any of the environments for implementing simulation or a real model. For example, Java Agent DEvelopment framework (JADE) [35] can be used to develop a simulated environment.

First, concrete mappings to the logical domain must be performed which are based on a multi-agent system design. Most multi-agent system design approaches include two general phases: (1) External design, (i.e., designing a multi-agent system as a whole) and (2) internal design, (i.e., the internal structure of individual agents) [36]. In this model, we focus on the internal design only as far as the VSM functions go. The internal design of the agents involving implementations of the actual tasks depends on the domain. One example of the internal design will be given within the description of the case study.

External design usually consists of two major parts. Firstly, the tasks that the system must do have to be defined, grouped into conceptual structures (for example, roles), and assigned to the agents. Secondly, the interactions must be designed, including simple message passing as well as more complex interaction protocols.

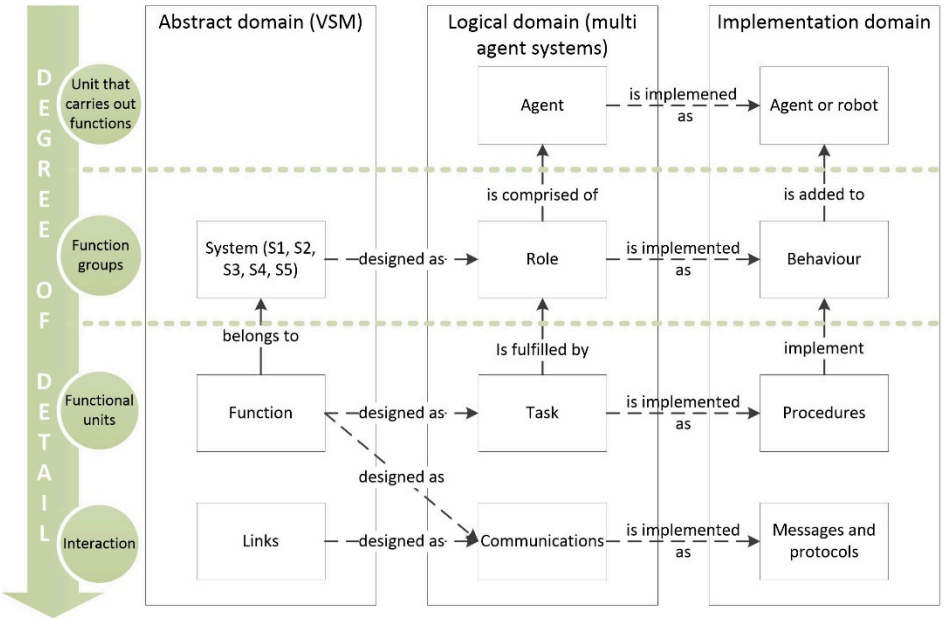

Figure 2. Mappings among the abstract, logical, and implementation domains used for ViaBots model.

The VSM eases the external design since it already defines the requirements and thus determines what kind of tasks and communications the system should have, as well as how these tasks should be grouped into roles. The overall structure of the agent tasks and roles is depicted in Table 1. To define ViaBots model based on the VSM, the functions (available in [19]) have been sorted and analyzed to determine which functions belong to the agent functions and which belong to the interaction design. It included the following four steps.

1. Organization-typical functions were excluded, in particular, "creation of working atmosphere" which refers directly to the emotional and social needs of humans.
2. Functions that directly refer to the systems' communication to other systems (such as "control channel functions") were assigned to interaction design, (i.e., communication).
3. The functions that are related to subsystems without the involvement of other subsystems were mapped to tasks (such as "learning").
4. Functions that involve communication with other subsystems, as well as specific tasks, were split between the design of tasks and interactions ("resource relocation" serves as both, the task for S3 and as an interaction between two subsystems).

Since the VSM is structurally independent, the subsystems correspond to the roles that can be given to any of the agents (structural units), and tasks correspond to internal design-specific functions acquired before. Tasks are mapped straight-forward on the functions or parts of the channel (in case of amplifier and attenuator). The only exception is operation (or S1) tasks where tasks are designed depending on the domain of the system and thus there can be multiple tasks. The tasks are grouped into roles according to the VSM; there are (5, n) roles since S1 again can, and most probably will, have more than one role. The rest of the systems have one role assigned.

Finally, the role assignment to agents depends on the capabilities of agents. There are several tasks that require specific capabilities. First, when interacting with the environment, the S1 agents

should be able to have reduction and amplification capabilities, (i.e., sensors and effectors). S4 agents should have environment monitoring capabilities.

Table 1. Task allocation to roles and role allocation to agents in ViaBots model.

VSM	Function	Task	Role	Agent
S1	Depends on the system	(1, n) tasks depending on the variety of the agents	(1, n) roles; depends on the domain	Either physical units that carry out roles (i.e., robots), or the ones that will run on one computer or in one domain. Assigned to agents having variety reduction and amplification capabilities
Type A	Attenuators and amplifiers	Attenuate variety task and amplify variety tasks between external environment and system		
S2	S1 element coordination	Negotiation Task	S2	
	Solving conflicts of S1			
	S1 resource relocation			
Type A	Attenuators	Attenuate variety task between Operation and Management		
S3	Resource distribution	The resource distribution task	S3	Assigned to one agent irrelevant from the already assigned roles
S3	Interpretation of policy decisions	Interpret high-level instructions to lower level task		
S3	Development planning according to environment and system states	Opinion task based on the current situation		
S4		Opinion task based on future states	S4	Assigned to one agent irrelevant from the already assigned roles Assigned to an agent having monitoring environment capabilities
S4	Suggestions for safety policy	Safety and resilience task		
S4	Learning	Learning task		
S4	Management of external contacts	Search task		
S4	Environment monitoring	Monitoring of environment task		
S5	Representation	Human-computer interaction task	S5	Assigned to one agent irrelevant from the already assigned roles
S5	Investment in structure and policy formation	Policy task		

It is also crucial to note, that in the VSM roles are distributed dynamically and the functioning system's structure can differ from the initial system's configuration.

Depending on the function and channel it supports, the following interaction types were identified (overview in Table 2):

- Type a and Type c: Channels that transfer information, in this case, a simple message is sent. In the case when variety amplification is needed, a broadcast message is sent.
- Type k channels are control flows where feedback is required. For this reason, an inform type message is sent and a mandatory confirmation must be sent back.
- Type b and type d channels include cooperation and negotiation that include complex interaction protocols that must be designed separately.

The VSM provides guidelines on how a VSM-based system should be designed, however, there are several functions open for interpretation; these questions (such as the content of the functions) are described in the next subsection.

Table 2. The interaction design in ViaBots model.

In VSM		Function	MAS Interaction Type
System	**Channel**		
S2	Type b	S1 element coordination	Coordination protocol
S2		Solving conflicts of S1	Negotiation protocol
S2		S1 resource relocation	Resource negotiation protocol
S2	Type k	Control channel function	Sent once, message-received response
S3	Type k	Control and monitoring over S1 and S2	Sent once, message-received response
	Type c		Inform message
S3 and S4	Type d	Development planning according to environment and system states	Negotiation protocol
S4	Type c	Monitoring of the system itself	Inform message
S5	Type c	Monitoring of cooperation of S3 and S4	Inform message
	Type k		Sent once, message-received response
Channel	Type a	Amplify variety between operation and management	Broadcast

3.3. Agent-Based Model for ViaBots

This section outlines two contributions for the implementation of the model, namely suggestions on how the tasks can be executed within the multi-agent system and the detailed ViaBots model ensuring adaptation, which is the central part in developing an adaptive system.

For each of the systems S1 to S5 defined in the VSM, a set of functions and some recommendations on how they should be implemented are given; yet task formalization for a technical system is vague in the VSM. Based on the tasks listed in Table 1, Table 3 provides a minimal list of tasks to implement adaptivity in autonomous systems, as well as suggestions on how it should be done (last column). All the functions mentioned in VSM are included in this minimal set except "management of external contacts" since the function is needed only if there are multiple autonomous systems in the external environment.

Overall, the ViaBots model functions are further implemented as described and shown in Figure 3. Operational units (i.e., robots or agents carrying out S1 functions) are capable to execute several tasks. Upon receiving an order from S2, they reconfigure to execute tasks of the requested type. Operational units also perceive the environment and send relevant information about current tasks to the corresponding S2 for aggregation.

Table 3. The implementation of tasks in a ViaBots model.

In VSM	Task	Implemented as
S1	(1, n) tasks depending on the variety of the agents	Various tasks (denoted as Task A, Task B, Task C)
Type A	Attenuate variety task and amplify variety tasks between EE and system	Inputs and outputs
S2	Negotiation task	Negotiating among S2 units based on the control value
Type A	Attenuate variety task between operation and management	Data collection
S3	The resource distribution task	Calculating control value
S3	Interpret high-level instructions to lower level task	Using the agents the user has created
S3	Opinion task based on the current situation	Weights of the calculated distribution
S4	Opinion task based on future states	Weights of the calculated distribution
S4	Safety and resilience task	Calculating the overall resilience parameters, such as battery level
S4	Learning task	The period sequence which S4 learns
S4	Monitoring of environment task	Due to the simplicity of the environment acquired through S1 and S2
S5	HCI task	UI
S5	Policy task	UI

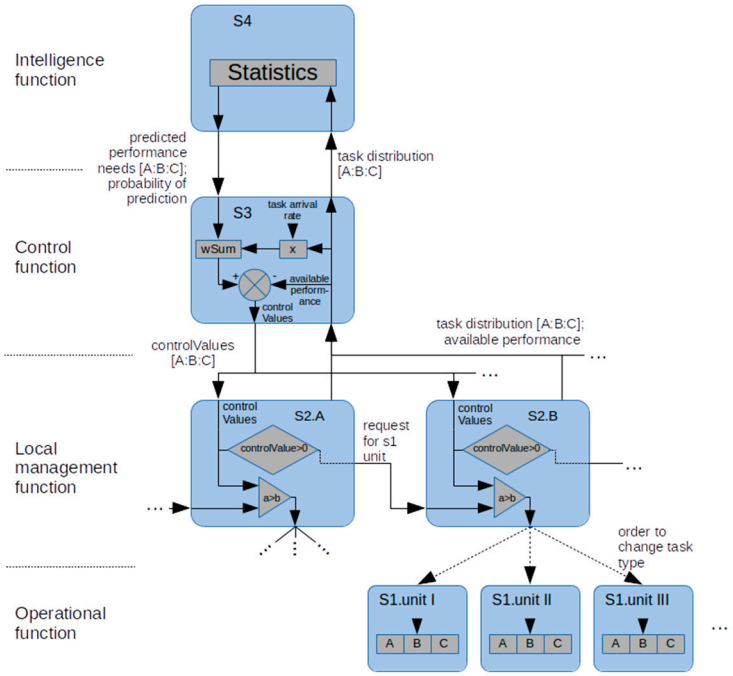

Figure 3. The overall structure of the ViaBots model.

S2 straightforwardly implements operational units' exchange mechanisms, that allow the diverse operational units to be used at the tasks where they are the most productive. The principle for the S2 negotiations is to respond to requests and give away the operational units that are relatively ineffective at their current task. The operational unit exchange is based on a comparison of control values received from the control subsystem, S3. If S2 has positive control value, meaning that task processing power of the corresponding type of operational unit is insufficient, it sends requests to other S2. When S2 receives such a request, it calculates if the commitment of its current least productive operational unit would be valid. The valid commitment of an operational unit means that after fulfilling request the source S2 would stay with a lower control value than receiver S2.

S2 aggregates information from subordinate operational units and provides a control subsystem with information of processing power available for tasks of the corresponding type, and number of available tasks.

S3 collects information from all the S2, to get an overview of the current situation. This way, S3 gets information on a number of tasks of each type available for execution and the total amount of processing power the system has. Using this information, S3 calculates desired power for processing tasks of each type currently available for execution. These values are combined with values of desired power for processing the predicted number of tasks by using a weighted sum. S3 then calculates control values for each of the S2 (i.e., each task type).

S3 provides intelligence subsystem S4 with current information on the system's state. S4 stores this information and uses it to update its model of the world. By using this model, S4 predicts a possible number of tasks of each type and returns this information to S3.

In the case of technical systems, the mission of the system is human-defined. Also, the available resource definition and other global configuration in many cases are done by the user. The S5 is also the most complex; to balance the complexity of implementation and system's ability to show adaptation capabilities, policy level decisions can be reduced to determining the number of operational units and thus are implemented as control elements in the user interface of the model. This means that in the current version of the model, the user fulfills the functions of the S5.

4. Case Study

We used a conveyor as a use case for the ViaBots model simulation and testing. Since the production lines involving multiple robots doing various tasks are the type of systems where adaptivity is crucial to achieve higher efficiency and avoid unnecessary downtime, a simple production line was chosen as a case study within this research. We used agent-based modeling as a core technique for simulation scenario implementation. The roles of the worker-robots and management within the simulated environment are fulfilled by software agents which enables one to one mapping between the agents used during the design and the agent-based model.

During the experiments, our goal was to validate the role allocation mechanisms over time to minimize the conveyor idle time. The goal of this implementation was not to achieve the best possible performance of the production line using given agents but to show how different aspects of the proposed model influences the overall performance of the chosen system.

4.1. Problem Description

The system is based on a conveyor belt that supplies parts to the worker-robots. Each part must be processed by one robot. Each robot can process only one part at a time. The parts are different, each type of them can appear on the conveyor belt with a changing probability. Robots are different in the sense of how much time each of them needs to process a particular part. Moreover, to change the task that the robot is doing (change the type of parts that it is processing), it must change the tools, which takes some time. Whenever there is no robot that can process the part, and the part has reached the end of the conveyor, the whole conveyor system must be stopped until there is a robot that can process it. The model of the system is illustrated in Figure 4. The conveyor carrying incoming parts is considered the environment of the simulated production line because our production line has no direct control over the rate and distribution of incoming parts. Consequently, there should be some regulation/control mechanism to keep operational units effectively distributed.

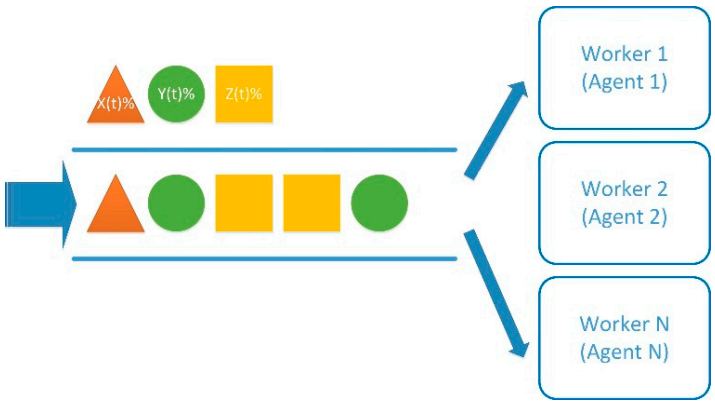

Figure 4. Model of the conveyor system used as a case study.

The production line must adapt to the following variables of the environment and the system itself:

1. The distribution of the incoming parts is changing over time due to varying part distribution on time. To simulate a change in the environment, the simulation process includes four periods with different part distributions. This model simulates four imaginary seasonal changes in the processes and the necessity to adapt to them.
2. The number and types of robots within the system might change using availability distribution of the robots.

The adaptation mechanisms must consider differences of the robots in terms of their capabilities. In general, it should be known how much time (and other resources, if applicable) each robot needs

to do a particular task (process the corresponding part). The system is using the ViaBots model to determine which agent (worker) is taking which role. An example of role allocation between agents is given in Figure 5. Gray color means that the agent is not executing a particular role at the moment while other colors indicate the opposite.

Figure 5. An example of task allocation among agents.

The main performance indicator we use during this case study is the idle time of the conveyor relative to the whole simulation time, which indicates how well the system adapts to the changing operating conditions. The system must minimize the idle time thereby maximizing production performance.

4.2. Implementation

The implementation of the conveyor system working in a simulated environment is based on the ViaBots model described in Section 2. The system has been implemented with a purpose to prove the concept of the ViaBots model for multi-robot systems. This section first describes the implemented simulation environment and then outlines how the functions defined in the ViaBots model are implemented in case of such a multi-robot system.

4.2.1. Simulation Environment

The system has been implemented in a multi-agent system development platform, JADE [35]. An agent development platform was chosen for the implementation because it had built-in mechanisms for agent interaction and operation. The simulation was implemented based on ticks using JADE TickerBehavior that has a cyclic nature of execution. TickerBehavior class has the timer that can be set to the same frequency as a master simulation timer, thus achieving synchronous iteration flow of components running in separate threads. Consequently, each ViaBots model role is implemented as a subclass of TickerBehaviour and all functionality associated with the particular subsystem is executed in these classes. JADE allows each behavior to be added for execution to an arbitrary agent, which is useful since VSM defines the functionality and not the physical details of the unit carrying it out. Physical aspects of the simulated robots are represented by a JADE agent subclass that contains variables describing agents' capabilities to carry out the roles. At each point, the tasks that an agent is executing, are defined by the behaviors that are added to the instance of the agent.

Communication between subsystems defined by the ViaBots model was implemented using JADE message passing mechanism. Topic-based broadcasting was mostly used, where behaviors subscribe to topics they are concerned with. This approach allows virtually direct communication between

behaviors, avoiding direct addressing of agents that execute them. Therefore, no centralized address catalog that maps ViaBots model roles to agents executing them is needed, at least at a logical level.

The average part arrival rate is set by the parameter that describes part arrival probability at each iteration. If a random number drawn from the uniform distribution is less than the given part arrival rate, then the part is generated. The type of generated part is determined in a similar way. The parts distribution is summed as the series of intervals, where the length of each interval is equal to a percentage of one part type from a given distribution. Then a random number is drawn from a uniform distribution and a part of the particular type is generated according to the interval where the random number fits.

The user interface of the implemented simulator is given in Figure 6. The simulator has the following functionality. The user can perform the following functions of S5 and set the following parameters:

- Define the types of parts and the probabilities of each part to be generated. The simulation is divided into four periods simulating seasons and the probabilities of each part can be defined separately for each period;
- The number of workplaces along the conveyor enabling the simulation of the situation when the length of the conveyor is not sufficient for all of the agents to work at the same time;
- Define the robots (agents) included in the simulation. The following parameters can be defined for each agent:
    - The average speed of processing parts of each type,
    - The power consumption of processing parts of each type.

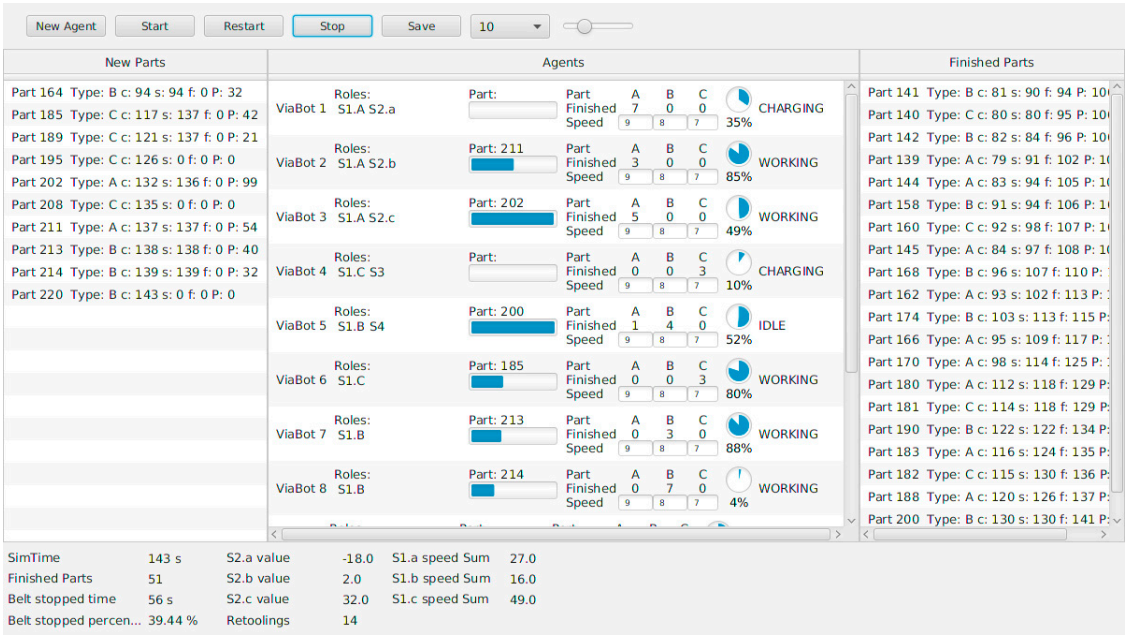

Figure 6. The user interface of the simulated production line environment.

During the simulation, the main window of the simulation environment showed three lists: First represented incoming parts, second—the robots processing parts at the current time, third—finished parts. There were also control elements for creating robots, regulating simulation speed, and starting or stopping the simulation.

Various parameters concerning overall systems performance and parameters of the management subsystems were displayed at the lower part of the simulation window. Parts were represented by their number, type, time of arrival, processing start time, and processing state. Robots were represented by their name, list of roles they currently were carrying out, type of part they were configured to process,

and part number. Part processing speeds, battery level, and general state were shown for each unit as well.

4.2.2. VSM Function Implementation

In the case of the production line, the S1 functions are related to the part processing. The design guidelines allow free interpretation of the number and type of roles and agents associated with S1. For sake of simplicity, there were three different part types possible in this simulation, namely, A, B and C. In this simulation, each agent can carry out S1 function and can be configured to process any of the three types of parts, although just one at a time and thus at any time belongs to just one of the S1 instances. For parts of each type, an agent has a particular speed (% per iteration) and power consumption (Watts). Each agent in the case of S1 represents a robot. This approach means that robots are the resources over which S1 systems compete. Any of the agents can be ordered to reconfigure itself to process another part type, but it takes a defined number of iterations for an agent to be ready to process it. On average a reconfiguration time equal to the time of processing three parts is used throughout tests.

As simulated robots in this production line scenario are considered mobile, each agent also has a battery, that has its capacity. The battery is discharged at each iteration by power consumption that the agent has at that particular moment. Total power consumption for each agent is a sum of the required power for all functions that the agent is carrying out at that moment. When the battery is discharged to a critical level, agent ceases part processing and recharges its battery, but it continues to carry out any management functions it has.

Attenuation and amplification tasks are fulfilled by agent's ability to process incoming parts and perceive these parts.

There is one main task for S2 and that is the ability to negotiate; it is associated with several protocols and control channel function as well. In our simulation environment, it is implemented as follows. The distribution of S1 instances over the agents is done by S2, based on control data provided by control subsystem S3. There is one S2 function for each S1 instance (i.e., part type). According to the ViaBots model, each S2 functionality (implemented as behavior) is concerned with a particular type of task. Within the use case of the conveyor system, it is part processing of their corresponding type. S2 possesses information about how many operational units at the moment are available to process parts of the corresponding type. S2 adds up part processing speeds of every operational unit available for processing parts of a particular type to calculate the total processing speed for that part type. This information is passed further to control subsystem S3; this filters the information necessary for S3 functioning and performs the attenuator task as well as control channel function. S2 units negotiate based on the information provided by S3 using the following mechanism: If S2 lacks resources, it requests an agent from all the other S2. When S2 receives a request, it calculates if it can give one of its operational units while remaining in a better position than S2 which made the request. Such an approach ensures that parts will be processed if there is at least one operational unit left.

S3, on the other hand, is concerned with the whole simulated production line performance and provides each S2 with the control value for parts of the corresponding type. According to the ViaBots model, there are three tasks in the S3. Within the conveyor case study, the resource distribution is achieved via the control value which is a difference of desired and available part processing speeds. Desired part processing speed would allow achieving continuous operation of the production line. Positive control value means, that there are not enough operational units for processing parts of a particular type. S2 use this control value provided by control subsystem S3 to exchange available resources (i.e., operational units with other S2 subsystems). S3 interprets policy decisions by following the user's choices.

Control subsystem S3 calculates desired speeds as a weighted sum of current required speeds and predicted speeds. Intelligence subsystem S4 provides S3 with predicted incoming part distributions (rates), and current part distribution on the conveyor is perceived from the environment (via S1 and

S2). Desired part processing speeds are calculated by multiplying part count of each type currently on the conveyor by part arrival rate, because parts should be processed at least at the rate they are arriving, to keep the production line from stopping. This enables the interaction of S3 and S4.

The main goal of S4 is to predict changes in the environment to enable adaptation to the upcoming new situations. In this production line scenario, information of current and upcoming period part arrival rates keeps operational units from unnecessary tool changes, since S3 has information only about current distribution of parts on the line, which may not represent the average distribution. S4 prediction of the distribution of the next period at the end of the current period is also used by S3 to make operational units gradually change their tools so that when the next part distribution period starts, there already are operational units ready to process parts at a different rate corresponding to the new period. For the sake of simplicity and transparency of this simulation, S4 uses the same part distribution values as are used for part generation. This approach allows evaluating the efficiency of ViaBots model if the prediction is impeccably precise.

5. Results

This section describes the experiments done with the simulated multi-robot conveyor system. The following experiments were executed to validate the implemented model. Firstly, the repeatability of the experiments was analyzed to determine what were the differences between different simulations with the same parameters since the environment by its nature is stochastic. Secondly, the importance of prediction and current observations of the part distribution probabilities was analyzed. The model uses both, the observed values and predicted ones, but the weights for both distributions must be determined experimentally. Thirdly, the system's performance depending on the number of the agents processing the parts was analyzed to choose the number of agents for the analysis of the ViaBots model's efficiency since the results may differ in case of inadequately large or small number of agents. Finally, the performance of the system with different active parts of ViaBot model was analyzed to validate the proposed model.

In all experiments, a battery capacity of 100 Wh was used. Power measurement unit W used in this simulation corresponds to Watt in the SI system, with the exception that time units were iterations. The battery iwa charged at constant 5 W. Part processing of types A, B and C consumed 2, 3, and 4 W (per iteration), respectively. Execution of any additional management role consumed one extra Watt.

Also, in all experiments, the length of the conveyor belt was equal to the number of agents, so that all the robots, at least in the simulated environment, were able to work simultaneously. The pattern of periods with distinctive average distributions of incoming part types is the same for all experiments and is shown at the top of Figure 7.

5.1. Repeatability of the Experiments

Due to the stochastic nature of previously described simulation of ViaBot model production line management, several simulation runs were done to determine how different the simulation results can be with the same parameters. Four typical runs of the simulation are given in Figure 7. Twelve identical robots were used in this experiment. Processing speeds for parts were set as a percentage of the total part processing time per iteration (9% for part type A, 8% for part type B, and 7% for part type C). The results differed up to 10% between different runs. Therefore, the mean values of four different runs were obtained for evaluation of systems performance within further experiments.

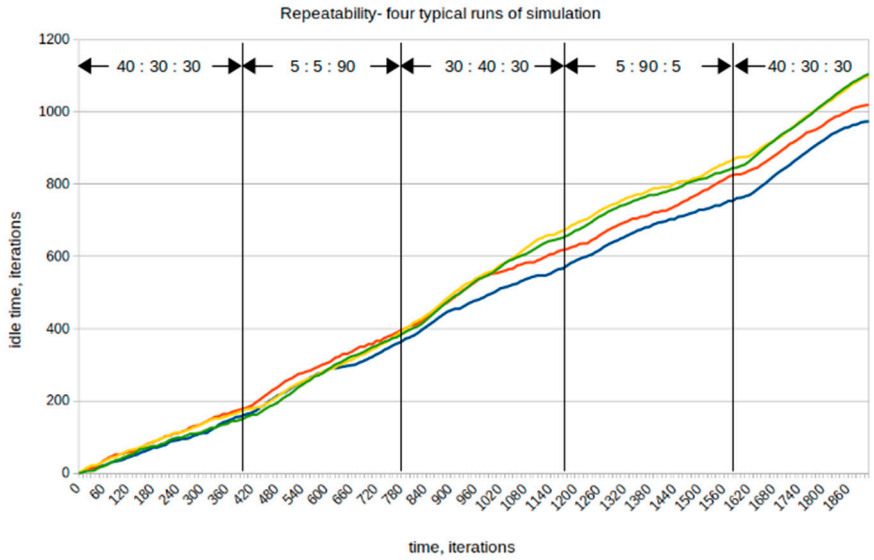

Figure 7. Repeatability of the experiments.

5.2. The Balance between Prediction and Observations

In this evaluation, the balance between the importance of the prediction and the observation in terms of weights to be used by S3 was determined empirically. Generally, optimal weights for predicted and observed incoming part rates would depend on the accuracy of prediction. To avoid analysis of how the prediction accuracy influenced the results, during this study we used the advantage of the simulated environment which allowed exploiting prediction with absolute accuracy. Other factors that influenced these weights were systems characteristics, (e.g., mechanisms for operational unit exchange between S2). The idle time was measured with changing weight distribution. As a result, the dependency shown in Figure 8 was obtained. The results show that the most efficient output of the production line for this specific setup was achieved using the weight of prediction between 0.6 and 0.8. Results were obtained by taking the average value of four consecutive simulation runs at 1900th iteration after four periods of different average distributions of incoming parts. Again, 12 identical robots were used where processing speeds for parts A, B, and C were 9%, 8%, and 7% per iteration, respectively.

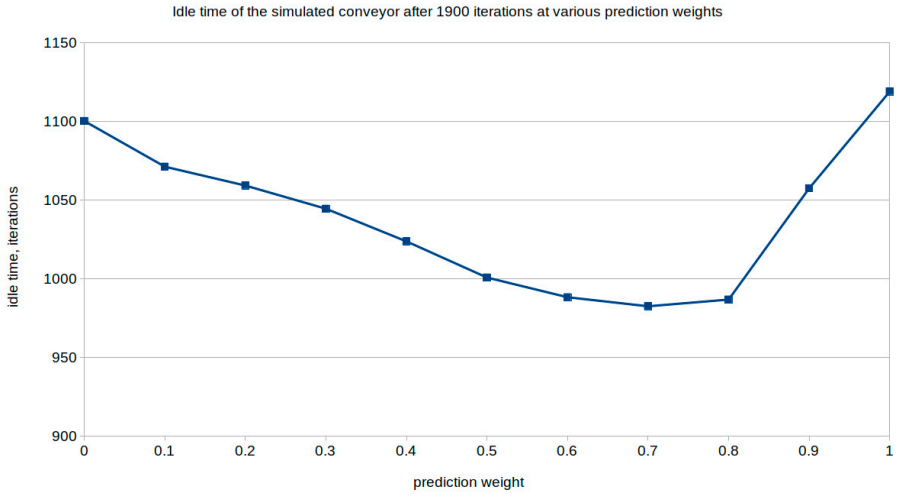

Figure 8. The idle time dependence on influence distribution between S3 and S4.

The fact was that the use of large prediction weight even with real prediction numbers caused the system to ignore the actual distribution of incoming parts on the belt. Consequently, S3 used prediction with a weight of 0.7 for further calculations of the control value.

5.3. Number of Agents

The main factor that influenced the performance of the system was a number of operational units (agents in this case). To compare various aspects of the ViaBots model, the quantity of operational units should not oversaturate the system, since the main goal of management is to make the system viable with limited resources. Consequently, management quality would be visible best when the number of operational units is limited. Figure 9 shows the dependency of the system's performance (the time, when the system is idle, was measured again) from the number of agents available in the system. As in the previous experiment, identical robots were used where processing speeds for parts A, B, and C were, respectively, 9%, 8%, and 7% per iteration. Based on these experiments, it was determined that at 15 agents effective use of any agent can still significantly change the performance of the system, therefore visibly indicating adaptivity.

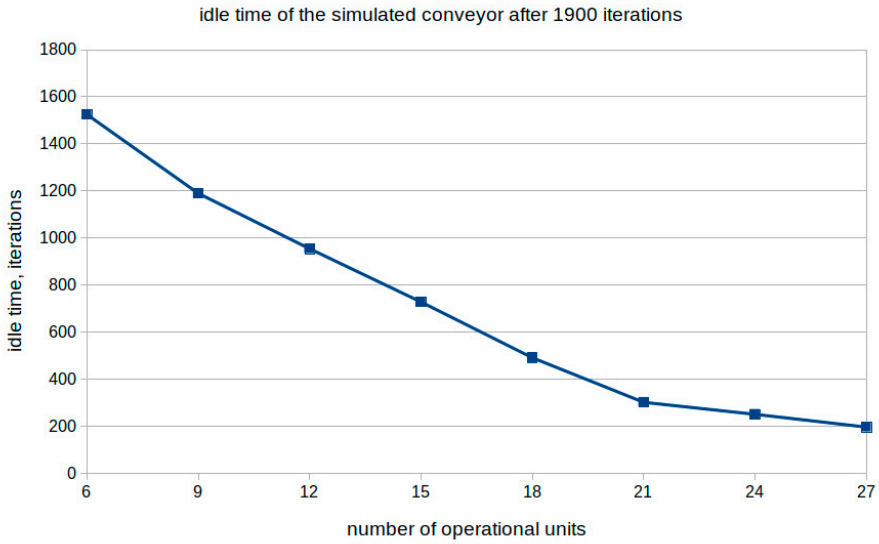

Figure 9. Systems performance depending on the number of agents.

5.4. The Efficiency of the ViaBots Model

Finally, the ViaBots model has been validated based on the simulated production line by comparing different levels of ViaBots model engagement using previously acquired parameters. Four different usage scenarios were analyzed:

1. Normal operation, where ViaBots model is fully applied;
2. Operation without prediction updates from S4;
3. Operation without control value updates from S3;
4. Independent operation of processing units.

Since S5 functions were partially limited by specific implementation structure (i.e., system is configured by the user and the goal of the conveyor system does not change), the implication of S5 subsystem functionality was not considered in this evaluation and only use of systems S2–S4 was validated.

Four periods of distinctive distribution of incoming parts formed environment fluctuations to which a group of 15 operational units of three different ability profiles adapts. Each ability profile had a speed of 0.09 parts per iteration for two types of parts and 0.03 parts per iteration for one remaining

type of parts. That means that there were five robots with speeds 3%, 9%, and 9% per iteration for parts A, B, and C, five robots with speeds 9%, 3%, and 9% per iteration, and another five robots with speeds 9%, 9%, and 3% per iteration, respectively.

When S4 was suppressed, production line continued to operate using prediction data (parts distribution) that may have been be outdated and therefore would not correspond to the actual part arrival rate (either current or future). S3 subsystem still calculated somewhat valid control values, because it also used actual data with a weight of 0.3. The loss in performance after 4 distinct periods was 14 % as shown by the red line in Figure 10.

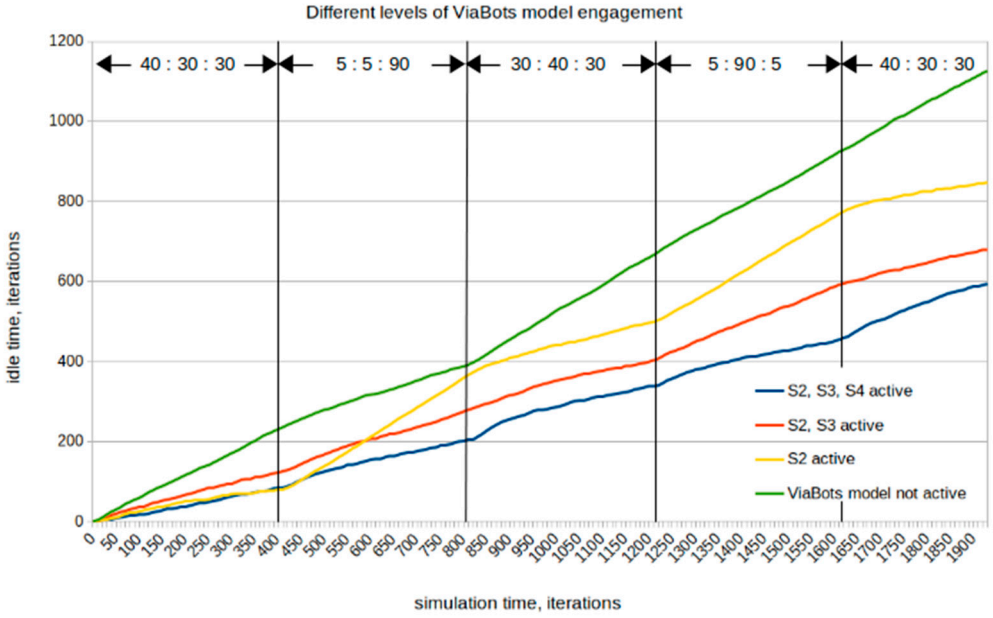

Figure 10. System's idle time with different parts of VSM enabled.

Suppression of control function S3 led to even less efficiency. When the agent that was carrying out S3 role was paused, local management subsystems S2 stayed with the fixed control value, as no updates were received. That can lead even to a complete stop of part processing if incoming part distribution varies substantially. Nevertheless, the distribution of operational units is based on their performance and matches the distribution of incoming parts of the first period. Furthermore, fixed control values prohibit any operational unit transfer thus reducing the loss of part processing power due to the reconfiguration time. However, inability to adapt to the changing environment due to a lack of management overshadows such benefits as shown by the yellow line in Figure 10, leading to further loss in performance by 40% compared to a case of fully engaged ViaBots model.

Finally, if local management subsystems S2 had selected operational units regardless of their performance and ceased further collaboration, then it could be considered that there was no ViaBots model applied at all. In this situation, operational units were not managed. The resources were distributed according to the distribution of incoming parts in the first period without taking into account their ability to process the parts of a particular type. The green graph shows such a scenario, where the system's idle time increased by 90% compared to the performance of a fully applied ViaBots model.

6. Discussion

The ViaBots model for long-term adaptivity in multi-robot systems is general, and the proposed algorithms can be applied to different types of multi-robot systems. The specific part that must be developed for each application of the model is the evaluation mechanism determining the system's capabilities to deal with each particular type of task. The mechanism implemented in the case study is

limited to the systems where each robot at a particular moment can have tools to do only one task. More complicated mechanisms are necessary for systems where robots are capable to do multiple tasks with the same configuration.

The conveyor case study included the typical task allocation challenges for production systems and multi-robot systems overall. The experiments with the implemented simulator showed that the ViaBots model enabled the system to adapt both to the changes in the environment (the tasks that robots must do), as well as to the changes in the system itself, in particular, in the set of robots available. These first results can be interpreted as follows: The ViaBots model is applicable to the systems that need to redistribute tasks as a reaction to the changes in the system or environment itself. At the same time, it is important to notice, that the results outlined in the paper are acquired at one specific setup that is meant to correspond to some possible real-life scenario and could vary over different incoming part rates and operational unit parameters. The actual efficiency of the model for other scenarios can vary even more depending on the quality of implemented functions, for instance, on the accuracy of S4 function, or the prediction model as well as depending on the nature of the system. To evaluate the efficiency of the model further experiments with other systems must be done.

Another limitation of the current implementation was the fact that the S5 subsystem was completely implemented by a human user. The system is not capable to autonomously change its goals instead they can be changed by the user. This fully complies with the needs of production systems where the change of the top-level goal, for example, producing one type of product, is up to a user. Still, in missions where more long-term autonomy is necessary, the system might need also the capabilities to change the top-level goal.

7. Conclusions

The main contribution of the paper is the ViaBots model that combines VSM with the multi-agent paradigm to ensure long-term adaptivity or viability in technical systems. The proposed model is unique in the sense that it applies principles from the organization theory to technical systems. The model was validated on a typical production line scenario where multiple robots are working on the same conveyor and processing parts. The adaptation within the production line scenario is implemented as a capability to reconfigure the system or reallocate tasks among robots in case the tasks change, or any robot is not available anymore (charging, breakdown), or a new robot is added to the system. Application of every system, starting from S2 to S4 gave additional adaptation capabilities and as a result increased the performance of the system. This proves that the functions of all these subsystems are adequately defined and multi-robot systems can benefit from each of them. In total, if the implementation of the ViaBots model is removed from the system, the time when the system is in the idle state increases almost twice (by 90%) in a scenario when the task distributions have significant seasonal change. This proves that efficiency of autonomous systems can be improved by applying principles from system theory to make them adaptive.

At the moment, validation was done only in a simulated environment. Despite the fact that the first results are promising, further validation of multi-robot systems is necessary. This defines the main direction of future works. The model will be validated on two different multi-robot systems. First, it will be implemented on heterogeneous robots with manipulators picking parts from a conveyor belt. This will repeat the current experiments with a conveyor system in a real environment. Second, a scenario with multiple small-scale mobile robots exploring the environment and collectively moving objects in a dynamic environment will be implemented. Later, more intelligent behavior of S4, as well as a partial implementation of S5, based on machine learning or other artificial intelligence methods will further improve the model. Moreover, the use of VSM for other technical systems should be investigated as well.

Author Contributions: Conceptualization M.P., E.L. and A.N.; data curation, formal analysis, software, A.A.; funding acquisition, project administration, supervision E.L.; investigation, M.P. and A.N.; methodology, M.P.,

A.A. and E.L.; resources, A.N.; validation, A.A. and A.N.; visualization, M.P. and A.A.; writing—original draft, review & editing M.P., A.A., E.L. and A.N.

Acknowledgments: This work has been supported by the European Council Seventh Framework Program FLAG-ERA project "Rethinking Robotics for the Robot Companion of the Future" (RoboCom++).

Conflicts of Interest: The authors declare that there is no conflict of interest.

References

1. Dario, P. Fet-Flagship proof-of-concept Project: Rethinking Robotics for the Robot Companion of the future. In Proceedings of the 26th IEEE International Symposium on Robot and Human Interactive Communication, Lisbon, Portugal, 28 August 2017.
2. Ņikitenko, A.; Lavendelis, E.; Ekmanis, M.; Rumba, R. Task Allocation Methods for Homogeneous Multi-Robot Systems: Feed Pushing Case Study. *Autom. Control Comput. Sci.* **2018**, *52*, 371–381. [CrossRef]
3. Wojtynek, M.; Oestreich, H.; Beyer, O.; Wrede, S. Collaborative and robot-based plug & produce for rapid reconfiguration of modular production systems. In Proceedings of the 2017 IEEE/SICE International Symposium on System Integration (SII), Taipei, Taiwan, 11–14 December 2017; pp. 1067–1073. [CrossRef]
4. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Mohamed, N.A.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. *Heliyon* **2018**, *4*, e00938. [CrossRef] [PubMed]
5. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **2018**, *362*, 1140–1144. [CrossRef] [PubMed]
6. Chan, C.Y. Advancements, prospects, and impacts of automated driving systems. *Int. J. Transp. Sci. Technol.* **2017**, *6*, 208–216. [CrossRef]
7. Foundation for Intelligent Physical Agents. FIPA Contract Net Interaction Protocol Specification. 2002. Available online: http://www.fipa.org/specs/fipa00029/SC00029H.html (accessed on 27 February 2019).
8. Beer, S. *Diagnosing the Systems for Organizations*; John Wiley & Sons: Hoboken, NJ, USA, 1985; 152p.
9. Matarić, M.J. Issues and approaches in the design of collective autonomous agents. *Robot. Auton. Syst.* **1995**, *16*, 321–331. [CrossRef]
10. Parker, L.E. ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation. *IEEE Trans. Robot. Autom.* **1998**, *14*, 220–240. [CrossRef]
11. Matthews, G.; Reinerman-Jones, L.E.; Barber, D.J.; Teo, G.; Wohleber, R.W.; Lin, J.; Panganiban, A.R. Resilient Autonomous Systems: Challenges and Solutions. In Proceedings of the 2016 Resilience Week (RWS), Chicago, IL, USA, 16–18 August 2016; pp. 208–213. [CrossRef]
12. Vassev, E.; Sterritt, R.; Rouff, C.; Hinchey, M. Swarm Technology at NASA: Building Resilient Systems. *IEEE IT Prof.* **2012**, *14*, 36–42. [CrossRef]
13. Jafari, M.; Xu, H. A biologically-inspired distributed fault tolerant flocking control for multi-agent system in presence of uncertain dynamics and unknown disturbance. *Eng. Appl. Artif. Intell.* **2019**, *79*, 1–12. [CrossRef]
14. Saulnier, K.; Saldaña, D.; Prorok, A.; Pappas, G.J.; Kumar, V. Resilient Flocking for Mobile Robot Teams. *IEEE Robot. Autom. Lett.* **2017**, *2*, 1039–1046. [CrossRef]
15. Guerrero-Bonilla, L.; Saldana, D.; Kumar, V. Design Guarantees for Resilient Robot Formation on Lattices. *IEEE Robot. Autom. Lett.* **2019**, *4*, 89–96. [CrossRef]
16. Tuci, E.; Alkilabi, M.H.M.; Akanyeti, O. Cooperative Object Transport in Multi-Robot Systems: A Review of the State-of-the-Art. *Front. Robot. AI* **2018**, *5*, 59. [CrossRef]
17. Zhang, T.; Zhang, W.; Gupta, M.M. Resilient Robots: Concept, Review, and Future Directions. *Robotics* **2017**, *6*, 22. [CrossRef]
18. Tarasyuk, A.; Pereverzeva, I.; Troubitsyna, E.; Laibinis, L. Formal Development and Quantitative Assessment of a Resilient Multi-robotic System. In Proceedings of the SERENE 2013: International Workshop on Software Engineering for Resilient Systems, Kiev, Ukraine, 3–4 October 2013; pp. 109–124. [CrossRef]
19. Pudāne, M.; Lavendelis, E.; Ņikitenko, A.; Ekmanis, M. ViaBots: A Concept for Viability for Distributed Systems. In Proceedings of the STO-MP-AVT-241—Technological and Operational Problems Connected with UGV Application for Future Military Operations, Rzeszow, Poland, 20–22 April 2015; pp. 10-1–10-12.
20. Skyttner, L. *General Systems Theory: Ideas & Applications*; World Scientific Publishing Co. Pte. Ltd.: Singapore, 2001; 459p. [CrossRef]

21. Peres Rios, J. *Design and Diagnosis for Sustainable Organization*; Springer: Berlin, Germany, 2012; 250p. [CrossRef]
22. Pudane, M. Knowledge Flow Analysis using Viable Systems Model. Master Thesis, Riga Technical University, Riga, Latvia, 2013.
23. Kirikova, M.; Pudane, M. Viable Systems Model Based Information Flows. *Adv. Intell. Syst. Comput.* **2014**, *241*, 97–104. [CrossRef]
24. Kontogiannis, T. Modeling patterns of breakdown (or archetypes) of human and organizational processes in accidents using system dynamics. *Saf. Sci.* **2012**, *50*, 931–944. [CrossRef]
25. Kuusisto, T.; Kuusisto, R. The management of geographic information flows in crisis situations. In Proceedings of the 11th Americas Conference on Information Systems, AMCIS 2005, Omaha, NE, USA, 11–14 August 2005; pp. 1114–1122.
26. Espejo, R.; Reyes, A. *Organizational Systems: Managing Complexity with the Viable System Model*; Springer: Berlin, Germany, 2011; 278p. [CrossRef]
27. Preece, G.; Shaw, D.; Hayashi, H. Using the Viable System Model (VSM) to structure information processing complexity in disaster response. *Eur. J. Oper. Res.* **2013**, *224*, 209–218. [CrossRef]
28. Allenna, L. The Viable System Model and Its Application to Complex Organizations. *Syst. Pract. Act. Res.* **2009**, *22*, 223–233. [CrossRef]
29. Murad, R.S.A.; Cavana, R.Y. Applying the viable system model to ICT project management. *Int. J. Appl. Syst. Stud.* **2012**, *4*, 186–205. [CrossRef]
30. Wilberg, J.; Tommelein, I.D.; Elezi, F.; Lindemann, U. Supporting the Implementation of Engineering Change Management with the Viable System Model. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Kowloon, China, 9–12 October 2015; pp. 731–736. [CrossRef]
31. Kummamuru, S.; Hussaini, S.W. Designing an organization structure for large and complex IT programs using the Viable System Model (VSM). In Proceedings of the TENCON 2015—2015 IEEE Region 10 Conference, Macao, China, 1–4 November 2015; pp. 1–5. [CrossRef]
32. Bonci, A.; Pirani, M.; Cucchiarelli, A.; Carbonari, A.; Naticchia, B.; Longhi, S. A Review of Recursive Holarchies for Viable Systems in CPSs. In Proceedings of the IEEE 16th International Conference on Industrial Informatics (INDIN), Porto, Portugal, 18–20 July 2018; pp. 37–42. [CrossRef]
33. Spyridopoulos, T.; Maraslis, K.; Tryfonas, T.; Oikonomou, G.; Li, S. Managing cyber security risks in industrial control systems with game theory and viable system modelling. In Proceedings of the 2014 9th International Conference on System of Systems Engineering (SOSE), Adelade, SA, Australia, 9–13 June 2014; pp. 266–271. [CrossRef]
34. Hilder, T. The Viable System Model. Available online: http://www.users.globalnet.co.uk/~{}rxv/orgmgt/vsm.pdf (accessed on 27 February 2019).
35. Bellifemine, F.L.; Caire, G.; Greenwood, D. *Developing Multi-Agent Systems with JADE*; Wiley Series in Agent Technology; John Wiley & Sons: Wes Sussex, UK, 2007; 286p. [CrossRef]
36. Lavendelis, E. Open Multi-Agent Architecture and Methodology for Intelligent Tutoring System Development. Ph.D. Thesis, Riga Technical University, Riga, Latvia, 2009.