RECONFIGURATION MANAGEMENT OF CRISIS MANAGEMENT SERVICES

J.B. van Veelen, S. van Splunter, N.J.E. Wijngaards, F.M.T. Brazier [*]

Thales Research & Technology/D-CIS Lab/Vrije Universiteit Amsterdam

Keywords

Software Architecture, Distributed Systems, Crisis Response and Management, Autonomous Systems

Abstract

Management in crisis response requires continuous adaptation, for crisis situations are highly dynamic. Crisis response in general involves multiple parties, each with their own autonomy and capabilities, leading to differentiations in structure, goals and strategies, and constraints for cooperation. A crisis management system needs to support distributed and continuous adaptation on different levels of organisation, in a reliable fashion, ensuring at least some minimal level of service for every defined task. This paper presents an architecture of a generic reflective autonomic management system (GRAM). The GRAM system tackles this real-time configuration challenge by the combination of a template-based configuration system (COWS) and a workflow-based configuration system (SMDS). The first proof-of-concept GRAM system shows a reliable and predictive performance in changing environments. Future work includes extending the current realisation and validating its performance in more realistic settings.

Introduction

Within crisis management continuously ad-hoc organisations are constructed and adapted. Likewise, the systems supporting these organisations have to be put into place and adjusted continuously. In other words, a crisis management system needs to support distributed and continuous adaptation on different levels of its crisis management-organisation. As the systems within an organisation can vary, a crisis management system effectively is a system-of-systems, aimed to gain and maintain control over the crisis at hand and the operations deployed. This process is (always) complicated by the dynamics of the problem (a crisis situation), the numbers of players and their changing requirements, and limited availability of time and resources. Although modernization of information technology in use in crisis management is rapidly taking place, the deployment and support for complex systems-of-systems involving multiple organisations have insufficiently matured so far.
These shortcomings are already visible in the exercises that are designed for training of collaboration of different professional organizations. A number of recent crisis management exercises ((NSO-N2005), (Viking), (Bonfire, 2005), (Lindgren, 2003), (Ringland, 2006), (Gouman, 2007)) demonstrate that cooperation within one team (horizontal) is already a

[*] J.B. van Veelen: bernard.vanveelen@decis.nl, D-CIS Lab, P.O.Box 90, 2600AB Delft, The Netherlands. S. van Splunter: svsplun@cs.vu.nl, N.J.E. Wijngaards: niek.wijngaards@icis.decis.nl, F.M.T. Brazier: frances@cs.vu.nl .

challenge, let alone cooperation between multiple teams (horizontal) or cooperation with higher authorities and lower echelons (vertical). Among the problems encountered are difficulties in managing information and decision making regarding the incident(s): too detailed information at a too-high staff or managerial level makes people abandon their roles and start addressing issues at a much lower level.

A major hurdle in the crisis management domain lies in multi-level deployment of resources, using both local (horizontal) policies and global (vertical) policies, while continuously optimizing over both dimensions. Therefore the apparent paradox that needs to be addressed is: How to find the right short term optimum for multi-level policy-based resource deployment, while ensuring that the used techniques offer better or the best real-time and runtime performance in the long term.

A major 'feature' of crisis management is that an unpredictable environment leads to unpredictable changes in goals and plans for the current crisis management organization. The resulting configuration problem needs to address multi-criteria optimizations of multiple interdependent goals and preferences together with interdependent agreements on selected plans. In this domain, both goals and agreements on plans must be considered to be pragmatic and renegotiable in real-time and runtime. Reasons for adaptation include unforeseen positive or negative developments in the crisis environment, as well as changes in the crisis management organization, including changing objectives and availability of resources.

A second 'feature' of crisis management is that decisions need to be made; even the absence of a decision is significant. The approach taken must not only find satisfactory solutions, but also provide alternative solutions (of lower quality), when conflicting goals and/or plans are in place. A secondary challenge lies in the combined robustness, stability and scalability of this approach: how can we prevent local changes to result in (undesired) global changes.

To provide support for crisis management organisations, especially in the areas sketched before, the Generic Reflective Autonomous Management (GRAM) architecture we propose offers:

- infrastructure for dynamically structuring, configuring, and re-configuring organisations to comply with actual goals and agreements on plans in crisis management conditions,
- different approaches to adaptation and (re-)configuration to be integrated,
- support for dynamic adaptation at a local level, minimizing the extent of the adaptations.

This paper describes the GRAM architecture. After a brief discussion of state of the art, the GRAM architecture is introduced, specifically aimed at the needs of dynamic time- and safety critical systems. Next the current prototype implementation is reported, followed by an explanation on how GRAM works in an application scenario. Finally, this paper presents a discussion on application of GRAM in a crisis management context and statements of future work.

Theory and Method

Nowadays, autonomous systems more and more share information to improve the quality of the decision making process. This information includes observations, but also intent, hypotheses and achieved results. On the same account, autonomous systems collaborate to improve efficiency and effectiveness in achieving common goals. In the resulting complex dynamic collaboration systems, the control of every detail of the organization can no longer be left entirely to human operators. That would be an error prone and dangerous approach in some cases; due to the complexity and the dynamics of the decision making processes involved, it would be an impossible approach in most cases. In time- and safety critical systems these drawbacks are even more pronounced, since human lives or economic interests are at stake. So, along with the trend to have dynamic collaboration systems, the need for reliable automated system's management arises. Also the aspects of self-formation and self-healing need to be addressed and implemented in the dynamic collaboration system.

The problem of organization and coordination for such dynamic collaboration systems has been studied in a number of contexts (e.g. (Mintzberg, 1983), (Alberts, 1999)) and numerous approaches for artificial systems have been proposed. Most noteworthy are efforts in Multi-Agent Systems, (such as (Bratman, 1999), (Wooldridge, 2002)), Swarm or Ant-Based Systems (Bonabeau, 1999), OpenWings (Bieber, 1999), Mape-K (Kephart, 2003) and Service Oriented Architectures (SOA) (Erl, 2005). These approaches share a bottom-up perspective, that is, organization and coordination is established as a *result* of the behaviour of each individual entity in the system. The organization is a result of negotiation (Multi-Agent Systems), environmental traces (stygmergy in ant-based systems) or complex client-server transactions (OpenWings, SOA). Each entity autonomously decides to participate in collaboration efforts with peers, for whatever reason.
Fundamentally different to this bottom-up approach is the top-down (hierarchical) approach, which claims to yield the most effective and efficient organization in safety- or time-critical situations. This approach is based on the assumption that the central command, by having some global overview, can make superior plans and decisions and can best tell individual entities what to do. This approach is predominantly being used in military (as in (Gerber, 1987), (van Creveld, 1987), etc), and indeed, crisis response organizations. Note that a hierarchical nature does not prohibit distributed management in artificial systems, see for example the approach taken in SMDS (van Veelen, 2006).
In the domain of web service (re-)configuration top-down task decompositions are used, and captured in templates, e.g., COWS (van Splunter, 2005), ReFFlow (Karastoyanova, 2004). Computational reflection (Cox, 2005) in web service (re-) configuration is enabled by the use semantic annotations, e.g., OWL-S (Martin, 2005).

GRAM Architecture

Within crisis management organisations multiple real-world parties are collaborating, (such as fire brigade, police force and government), each with their own resources and policies, forming an Actor-Agent Community: interacting human and artificial systems pursuing a common mission or supporting a shared process (Wijngaards, 2004). The interaction in an Actor-Agent Community (AAC) is based on a certain level of common knowledge. This common knowledge includes all knowledge regarding the current interests, roles, responsibilities, capabilities and activities of the organisation.
Given the interactions and collaborations between individuals of different parties, the crisis management organisation soon grows into a complex system-of-systems. Due to the dynamics of the crisis scenario, the crisis management organisation may frequently change its behaviour, its composition or adjust the (priority of) its goals. To manage all this complexity, GRAM structures interactions in layers, each layer performing different tasks. Using the simpler functionality offered by systems on a lower layer, a higher level system is capable of accomplishing a more complex task. The resulting intricate network of mutual dependencies for services and communication is managed in GRAM by introducing self-management at all levels. Each local self-management system needs to be aware of goals, roles and obligations at that layer, in order to successfully coordinate activities and help achieve the organisation’s mission. Conflict resolution and adaptations are preferably done as local as possible, that is, within the layer at which a conflict or need to adapt occurs. However, escalation to a higher level is possible if a local solution cannot be found.

Collaboration between systems and layers

For interactions between layers or parties, requirements need to be translated. Between layers (vertically) this translation can be a mapping to different levels of abstraction. Between parties (horizontally), requirements either need to be mapped to the different contexts of parties, or to an agreed up common context. The requirements from higher levels imposed on lower levels, and the agreed upon level of service between parties are laid down in Service Level Agreements (SLA) and Service Level Objectives (SLO). These formalizations of demand and supply can be used as a means to structure coordination, since it defines the

degrees of freedom (room to manoeuvre) and indicates the events in which synchronisation with stakeholders is required.
Figure 1 displays our view on policy definition within organisations, and on dependencies between interactions within and between organisations.

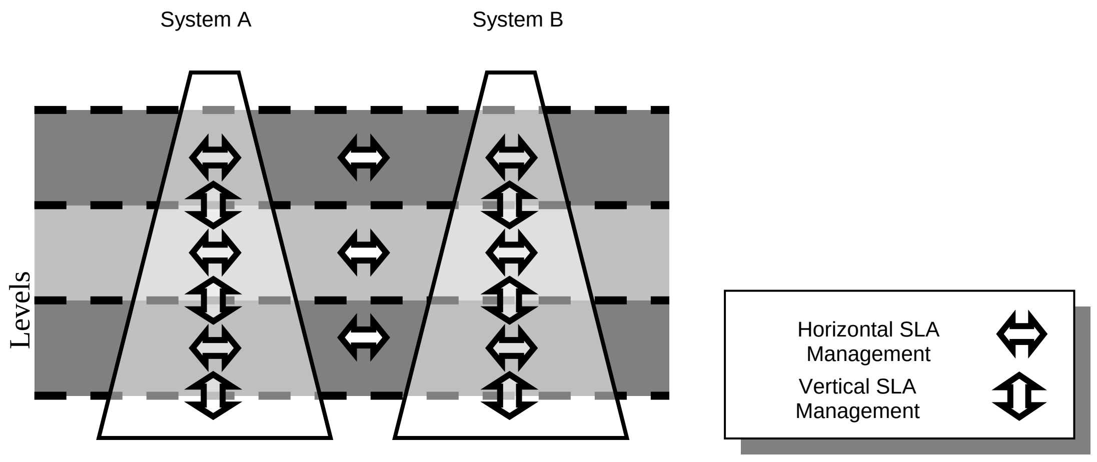

Figure 1: Illustration of multi-level SLAs consisting of horizontal and vertical SLAs

In Figure 1 we can distinguish three types of SLA management: horizontal SLA management within a system (**horizontal intra-system**) and between systems (**horizontal inter-system**), as well as **vertical** SLA management, always within a system. These types are specified as:

- **Horizontal intra-system** management aims to coordinate actions within a single system and a single level to fulfil the requirements received for this layer. Local management decisions are based on feedback of the entities defined within this layer of organisation.
- **Horizontal inter-system** management manages co-operation between systems. Co-operation is currently limited to stay within a level. Possible co-operations are specified as protocols that contain SLA's for the parties involved.
- **Vertical management** separates the decomposition of problems and the instantiation of tasks. Inter-level interactions specify requirements and objectives resulting from the decomposing layer in the context of the lower layer.

Local Management

The local management on each position in the organisation is executed by an adaptive system that responds to the desires of a set of stakeholders, and operates in a context of other systems and an environment. Our work uses the model presented in (Haydarlou,2006). Figure 1 illustrates that this system has two components: a managed system and a Reflective Autonomic Manager. The main tasks of the Autonomic Manager are monitoring, diagnosis, planning, decision making and configuration.
The stakeholders require services with constraints (the desired quality of service) from the manager. To fulfil these requirements, the Autonomic Manager composes and configures resources in the managed system, using knowledge regarding the available capabilities in the managed subsystem. If no possible composition satisfying all constraints can be found, the manager reports this back to the stakeholders.
A key notion is that the activities managed by the Autonomic Manager have a purpose in a greater context; any part of the system depends on other parts for input or services and provides itself output or services to other parts of the system.

This model can be applied recursively, when components in the managed system are allowed to have a full GRAM architecture themselves.

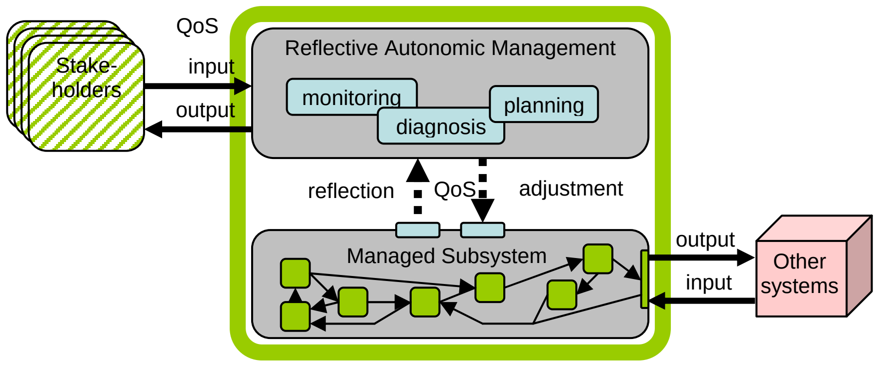

Figure 2: Representation of Reflective Autonomic Management component

Results

This section discusses the implementation of the GRAM architecture and the application scenario on which the prototype is applied.

Implementation

A prototype was constructed according to the GRAM principles. The first goal of the prototype is to show the feasibility of supporting alternative approaches to adaptation, and the second goal is to evaluate the strengths of a layered adaptation mechanism in multi-layered organisation. In this initial implementation two different approaches to system management were chosen, COWS and SMDS. First the main principles of the systems COWS and SMDS are described, after which the collaboration between the two in a multi-layered setting is discussed.

The main principle of COWS is automated template-based reconfiguration, which has previously been applied on complex web services (vanSplunter, 2005). The approach focuses on autonomous adaptation of a complex web service after failure of one of the services of the composition. Templates contain an abstract decomposition of tasks together with a control structure. Each subtask is constrained by a set of requirements; additionally interdependencies between tasks can be expressed within a template. The template definitions are based on OWL-S with a few minor extensions. Based on the requirements specified in the template, and the pre- and post-conditions of the other services used in the composition, the failing service is replaced without human intervention.

The main principle of SMDS (vanVeelen, 2006) is to solve information and service requirements, by using explicit knowledge in an automated reasoning process that matches supply and demand. Services publish the information or function they supply and/or consume. The automated reasoning process constructs a workflow by chaining providers; this workflow can be valued, and a comparison of values yields a ‘solution’, which is considered the best solution given current state of the system. The solution is instantiated, by starting services and sending configuration messages. The instantiation is monitored for progress and yielding the predefined level of service. If the instantiated solution fails to achieve the required progress or the required level of service, the SMDS management either selects another solution or notifies the stakeholders it is not able to deliver requested information or services.

The prototype contains interaction with a human user, the COWS and SMDS as separate reflective autonomous managers, and a simulation environment in which services and agents are managed. The environment generates events, which trigger the creation and instantiation of tasks. As COWS offers abstractions for tasks decompositions, and SMDS offers support for dynamic instantiation and monitoring, the layers in the prototype from high to low are implemented as: humans, COWS, SMDS, and the agents and services.

The highest layers (in our experimental setup) are autonomously decomposing tasks. The highest layer receives its requirements from human stakeholders, through some user interface. A task decomposition is generated by the COWS system. Each task resulting from the decomposition is delegated to the next layer. The COWS manager has a choice of either delegating it to another COWS manager, which will further decompose that task, or delegating it to an SMDS manager. In case neither option is feasible, the COWS manager will give the request back to the higher layer and, ultimately, the human stakeholder. The SMDS layer takes care of planning for resources and applications. The SMDS managers plan, instantiate and monitor a workflow. If the workflow (no longer) behaves according to plan, the SMDS manager has the options of either instantiating adjusting the workflow or instantiating another workflow, such that original requirements are met, or reporting back to the higher level that currently no suitable workflow can be found. Finally, at the level of individual services, which are implemented by software programs or software agents, each software process has the autonomy to act within set bounds. Act within set bounds means that it can spend resources interact with peers regarding information and services, and give feedback to SMDS managers.

Application Scenario

In the Netherlands fire fighting is managed based on so-called Veiligheidsregio’s (safety regions). The Netherlands is geographically divided in safety regions, structuring regional responsibilities and cooperation for fire fighters. The same safety region structure applies to police and medical services, and efforts are made to align the services within and between the regions to facilitate cooperation in crisis situations. This scenario focuses on fire fighting services, in which organisation is structured in station, regional and interregional levels. The implementation allows experiments with different policies for cooperation at different levels in the organization of fire fighters. Our main interest is in situations saturating the demand for resources, such as an extensive crisis or a high density of incidents. Though occurrence of these situations is rare, the importance of maintaining the ability to respond is critical for emergency services in these contexts.

In the scenario each station has a limited amount of personnel and equipment available, with optional specialization (e.g., equipment appropriate for handling chemical incidents). In a standard situation, incidents are handled on a routine basis, requiring no cooperation. In case incidents escalate, equipment fails, or new incidents occur with a high frequency (optionally within the same geographic region) cooperation is required. Within our scenario new incidents and escalations on a geographic location are introduced based on a time-line. The modelled fire fighting organizations need to respond dynamically. By using GRAM different policies and approaches to organization can be explored: for example strict hierarchical organization, bottom-up organization, dynamic virtual organization.

In a *strict hierarchical organization* cooperation regions are internally hierarchically organized and inter-regional cooperation is handled by policies at the regional level. A lower level needs escalate to a higher level for cooperation. A strict hierarchical organization leads to a clear separation of responsibilities; however performance might not be ideal. In a *bottom-up organization* stations contact other nearby stations in case of escalation, and cooperation is organised at the lower levels of the organization. The safety regions could be ignored, and responsibilities need to be determined on the fly. Though this approach is very dynamic, and able to handle incidents locally, performance on a larger scale can be suboptimal, as only local policies and local requirements are taken into account (e.g. sustaining a regional capacity for handling new incidents, needs policy enforcement from higher level). In a *dynamic virtual organization* at each level the full potential of the GRAM architecture is used, combining the hierarchical and bottom-up approaches to organization. The internal hierarchical structure of a region is modelled, but in addition cooperation can also be initiated at lower levels. Regional requirements can be met, by the specification of policies within the regions internal organisation. Each individual component of a system has its own local

Autonomic Manager and Quality of Service policies. The decision making facilities enable the Autonomic Manager to adjust or reconfigure the managed sub-system or notify a stakeholder due to immanent failure. The Quality of Service policy of a component in the managed system defines under which circumstances the Autonomic Manager, or one or more stakeholders, should be alerted. It can also define control measures (actions/messages) that need to be deployed in order to achieve the desired results.

Discussion

Meaning in Crisis Management Context

The initial setup of GRAM demonstrates that at some point, automated processes can facilitate management of service- and information requirements, by combining two approaches. Initially, when confronted with a problem, these approaches will construct an effective solution in collaboration:

First, by applying a condensation of best practices in the form of templates, a task can be decomposed in a logical, functional fashion. Human experts will define these templates. This guarantees the decomposition will have a general form similar to one that would have been generated by a human counterpart. This means that management will operate in a predictive fashion and that the decomposition will 'make sense' to human decision makers. When decomposing, the Autonomic Manager has to take care to precisely define the rules and conditions describing the dependency between the resulting templates.

Second, the workflow planning will take into account all the current activities, requirements and constraints when computing the best solution. Keeping track of all these dynamic variables is a task that cannot be executed by human decision makers efficiently. However, by checking the sanity of the rules and requirements in the templates (beforehand) every realization of a template conforming to the rules and conditions it defines, will necessarily constitute a solution that meets demands. Since the rules and conditions describing the dependencies between templates are included in the template, a combination of solutions, one for each (set of) requirements as laid down in a template, will constitute a coherent system of systems, where all activities are contributing to achieving the current mission.

When the current organisation runs into trouble, due to disturbances in the environment, failure of hard- or software processes, the management subsystem detects and responds to the need for adjustments.

The management system aims to keep adjustments as local as possible, preferably contained to the location where the disturbance occurred. If local adjustment is possible, this will be implemented and other layers will not be disturbed. However, if local adjustment is not possible within the given bounds and conditions, a next-higher layer is notified and asked for adjustments at that level.

The need for adjustment can also be initiated by the (human) stakeholders, who, due to the phase of the crisis scenario or the information they possess, decide the organisation should follow a redefined set of priorities. In this case, the new priorities propagate downwards through the system, where each consecutive Autonomic Manager deals with the new situation as if it were a fresh start, as described earlier.

This reliable, predictive mode of operation enables GRAM-based systems to relieve human decision makers of at least some of the burden in managing a complex, dynamic, distributed organisation, composed of multi-disciplinary autonomous teams.

Outlook

The architecture proposed in this paper offers support for dynamic (re-)configuration of systems-of-systems. Per level separate goals can be specified and managed by reflection. The prototype illustrates the architecture supports the integration of different approaches to adaptation. Adaptations kept as local as possible, as each system has reflection mechanisms, and adaptation mechanisms to act on the reflection. Agreements can be specified as based on an agreed requirement set. In future work this is to be extended to express these as Service Level Agreements. Already some initial work has been done in this respect (Rana, 2007).

In the near future, we intend to extent the current layered management system. First we want to be able to have multiple separate subsystems managed by one overall management system. In this case, there is no direct communication between the management layers of the separate subsystems, however, coordination between the two takes place in the next-higher layer. As an example observe Figure 2. The organisation style of Figure 2 can be typically useful in case the organisation is geographically distributed, or when it is impractical or inconvenient to share low-level information or resources. Another situation might be the case where the sub-systems have different and unrelated responsibilities, for example tending to wounded and fighting fire. In both cases individual entities of different sub-organisations will not contact each other directly, but instead use synchronisation and coordination mechanisms provided by the encompassing, higher layers.

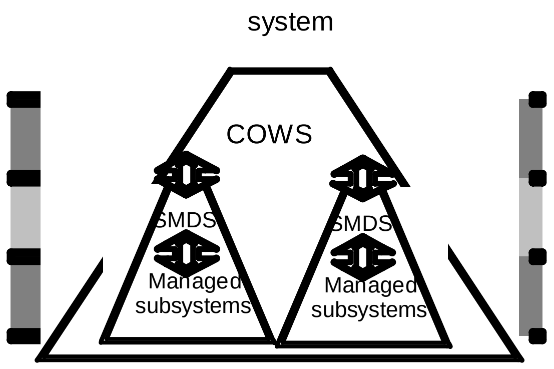

Figure 3: Levels of hierarchy of a compositional organisation, containing 2 sub-organisations

Acknowledgements

The research reported here is part of the Interactive Collaborative Information Systems (ICIS) project (http://www.icis.decis.nl/), supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024. The authors thank the NLnet Foundation (http://www.nlnet.nl) for supporting this work.

References

Alberts, D.S., Gartska J.J. and Stein, F.P. (1999). Network Centric Warfare. CCRP, ISBN 1-57906-019-6.
Bieber, G and Carpenter, J. (2001). Openwings a service-oriented component architecture for self-forming, self-healing network centric systems. Openwings, Tech. report, http://www.openwings.org
Bonabeau, E., Dorigo, M. and Theraulaz, G. (1999). Swarm Intelligence: From Natural to Artificial Systems. Santa Fe Institute studies in the Sciences of Complexity, Oxford University Press, ISBN 0-19-513159-2.
Bonfire, COT, observation report Bonfire, 2005
Bratman, M. E. (1999). Intention, Plans, and Practical Reason. CSLI Publications. ISBN 1-57586-192-5.
Brunner, M., and Keller, A. (Eds.) (2003). Self-Managing Distributed Systems, Lecture Notes in Computer Science, vol 2867, Springer Verlag, ISBN 3-540-20314-1
Cox, M.T. (2005). Megacognition in computation: a selected research review. Artificial Intelligence, 169(2):104–141.
Erl, T. (2005). Service Oriented Architecture, Concepts Technology and Design. Prentice Hall, ISBN 0-13-185858-0
Gerber, D.K. (1997). Adaptive Command and Control of Theater Air Power, see http://stinet.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA391859.

Gouman, R., Kempen, M., de Vree, Capello, Van der Heijden, E. and Wijngaards, N.J.E. (2007) “The Borsele Files: The challenge of acquiring usable data under chaotic circumstances”. Proceedings of ISCRAM 2007, pp 93-104.
Haydarlou, A. R., Oey, M. A., Overeinder, B. J. and Brazier, F. M. T. 2006. Using Semantic Web Technology for Self-Management of Distributed Object-Oriented Systems. In Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI-06), 489-493.
Karastoyanova, D. and Buchmann, A. (2004). ReFFlow: A Model and Generic Approach to Flexibility of Web Service Compositions. In: Proceedings of International Conference on Information Integration and Web-based Applications and Service, Jakarta, Indonesia, 27-29.
Kephart, J. and Chess, D. (2003) The vision of Autonomic Computing, IEEE Computer 36(1):41-50.
Lindgren and Bandhold, (2003). Scenario planning, the link between future and strategy, Palgrave Macmillan, 2003.
Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., and Sycara, K. (2004). Bringing semantics to web services: The OWL-S approach. In: Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), San Diego, Califonia, USA.
Mintzberg, H. (1983) Structures in Fives: Designing effective organizations. Prentice Hall, ISBN 0-13-854191-4.
NSO-N2005, Systeemevaluatie Nationale Staf oefening Nucleair (2005). By the Dutch Ministries of Housing, Spatial Planning and the Environment (VROM) and Interior and Kingdom Relations (BZK), published by Ministry VROM, September 2005.
Rana, O., Warnier, M., Quillinan, T. B., Brazier, F. M. T., and Cojocarasu, D. 2007. Managing violations in service level agreements. In the Proceedings of the Usage of Service Level Agreements in Grids Workshop.
Ringland (2006). Scenario planning, 2nd edition, Wiley & Sons, 2006.
van Creveld, M. (1987), Command in War. Harvard University Press, ISBN 0674144414
van Splunter, S., van Langen, P.H.G. and Brazier, F.M.T. (2005). The Role of Local Knowledge in Complex Web Service Reconfiguration. In: Proceedings of the 2005 IEEE/WIC International Conference on Web Intelligence (WI 2005), Compiegne, France, 495-499.
van Veelen, J. B. (2006). SMDS: a top-down approach to self-management for dynamic collaboration systems. In: Proceedings of the 2006 international Workshop on Self-Adaptation and Self-Managing Systems (Shanghai, China, May 21 - 22, 2006). SEAMS '06. ACM, New York, NY, 58-64. DOI= http://doi.acm.org/10.1145/1137677.1137689.
Viking Programma VIKING, Evaluatierapport Hagar, november 2005. and Programma VIKING, Evaluatierapport Helga, september 2006.
Wijngaards, N., Nieuwenhuis, K. and Burghardt, P. 2004. Actor-Agent Communities in Dynamic Environments. In Proceedings of the workshop ICT Agents, November 2004, TNO Defence, Security and Safety, The Hague, The Netherland
Wooldridge, M. (2002). An introduction to Multi-Agent Systems. John Wiley and Sons, ISBN 978-0-471-49691-5

Author Biography

Bernard van Veelen works as a research fellow for Thales Nederland B.V, focussing on topics of autonomous, agent-based, and distributed systems.

Sander van Splunter is a researcher at VU University Amsterdam and D-CIS Lab, focusing on automated reconfiguration of compositional systems.

Niek Wijngaards is a senior researcher and program manager for Thales Research & Technology Netherlands, and researches actor-agent teams applied in practical applications, e.g., at the Dutch Police organisation and the Dutch Railroads. Niek received his PhD in 1999 for research on self-modifying agent systems based on re-design.

Frances Brazier is a full professor at the VU University Amsterdam and chairs the IIDS group. IIDS' research focus is on (self-) management of large scale distributed autonomous systems. AgentScape is the middleware designed to support large scale distributed autonomous systems.