SMDS: A Top-down Approach to Self-Management for Dynamic Collaboration Systems

J.B. van Veelen
DECIS Lab - Thales Research & Technology,
Delftechpark 24
Delft, The Netherlands
bernard.vanneelen@decis.nl

ABSTRACT
In this paper we present a distributed hierarchical system management concept for dynamic collaboration systems. Dynamic collaboration systems are composed of platforms containing resources; the platforms join forces to achieve a common mission. The automated system management concept discussed here is called the Self-Managing Distributed Systems (SMDS) concept. The SMDS concept distinguishes four segments of management: the Planning Segment, the Instantiation Segment, the Monitoring Segment and the Federation Segment. In this paper we will introduce these four segments, explain the functionality in each segment and how the segments interact. Furthermore, we will compare this SMDS concept with a service oriented approach.

Categories and Subject Descriptors
D.2.11 [Software Engineering]: Software Architectures; I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—Coherence and Coordination, Multiagent systems

General Terms
Design, Management, Performance, Theory

Keywords
Autonomous systems, Coordination, Autonomous system’s management, Distributed systems

1. INTRODUCTION
Nowadays, autonomous systems more and more share information to improve the quality of the decision making process. This information includes not only 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 (DCS), 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 huge economic interests are at stake. So, along with the trend to have dynamic collaboration systems, the need for reliable automated system’s management arises.

Examples of complex dynamic collaboration systems for time- and safety critical situations include network centric warfare systems [2, 5], crisis management systems [3, 8, 9] and airport management systems [6, 14]. The extremely dynamic nature of such systems and the high stakes involved, have caused the necessity of a safe and secure system of systems management paradigm. This paradigm should regulate the autonomous organization and management of the hard- and soft-ware layers of the DCS, cued by high-level control of human operators. This requires novel approaches to middleware and application design.

Advances in networking and middleware technology seem to justify the assumption we can expect mature technology to support such a DCS management paradigm soon [1, 4, 5, 14, 16]. On the application level dynamic DCS management dictates a radical change from the conventional ‘stovepipe’-based systems. The first approach is to divide the stovepipe into independent, reusable and interoperable elements, as can be seen in Object Oriented and Agent Based Systems ([7, 11, 13, 17]). Interoperable and independent modules enable the more desirable features of a DCS: runtime integration of services, overall system performance optimization and predictable behavior in dynamic situations. However, division in such elements in itself does not provide these features. Also self-forming and self-healing capabilities need to be addressed and implemented in the DCS [7, 15].

Therefore, we introduce the Self-Managing Distributed Systems concept, aimed at exactly those features: system integration on the level of process collaboration and information sharing, performance optimization and predictable behavior in dynamic situations. A first report of this approach is documented in [25].

The scope of this paper is to describe a possible approach to dynamic systems management, with a focus on the underlying functional composition of such a system. In section 2, we will define the problem of dynamic systems management for a DCS. Section 3 investigates the current state of the
art. The SMDS approach is presented in detail in section 4 and explained in section 5. In section 6 we compare the SMDS approach to the state of the art. Section 7 contains some preliminary conclusions and states future work.

2. PROBLEM STATEMENT

When dealing with complex distributed collaboration systems, the actual collaboration is often complicated by the differences between the parties involved. The participants may play very different roles, have very different views on the world and very different interests. For example, the medics in a crisis response scenario will try to save lives by aiding the wounded, while the firefighters will aim to put out fires. However, there is a clear understanding of a common objective, to which end all participants strive. In [18] this common objective is defined as the Joint Persistent Goal (JPG).

To achieve this JPG, the parties involved will have to connect, exchange information regarding their perceptions, their intentions, observations and achieved results (or failures). Furthermore, they have to synchronize activities and divide tasks and responsibilities among themselves. To achieve the JPG, each party inputs an amount of resources (people, fire-engines or ambulances for example), which are capable of forming a network to exchange information with peers and central coordination groups. We define platforms as physical entities, containing a number of hardware and software components (like sensors and communication devices). There may be people aboard a platform to navigate it or operate its components. In our view the DCS is composed of multiple platforms, each equipped with various hard- and software resources, enabling it to perform tasks in the unfolding scenario. People are assumed to be represented in this DCS by the interfaces they operate.

The problem faced in a DCS is to manage the flow of information and activities in a way the common objective is achieved efficiently. The solution to this problem, coordination, is: "managing interdependencies between activities, enabling all resources to work together harmoniously in achieving a common goal" [19]. To coordinate the exchange of information and the synchronization of activities in a reliable fashion for time- and safety critical systems, we propose the SMDS concept.

The SMDS concept describes how to construct process collaborations (called Solutions) on demand, based on high level service need specifications. Service need specifications are expressed in terms of functional and non-functional requirements. As an example of a Solution, consider an ‘assessment system’. An assessment service A can assess a situation when supplied with recognized observations and intelligence data. Recognized observations are produced by an image extraction service C based on filtered sensor data. Raw sensor data from sensor service G can be filtered by a filter service D. Intelligence data is retrieved by a data collection service B from an image repository service E and an intelligence repository service F. This solution is depicted in figure 1. We will use this assessment solution in this paper to explain how process collaborations are constructed.

3. STATE OF THE ART

A common approach to coordination is to establish a runtime organization using a bottom-up approach: each entity is responsible for finding sufficient and adequate peers for the services it requires. This way, the run-time configuration emerges as a result of requirements and decisions of individual entities.

To examine the bottom-up approach, let us consider how the ‘assessment system’ is constructed at run-time. This is roughly the same for the object-oriented [23], agent-based (negotiation based) [11, 13, 22, 24, 27] and service oriented architectures [5, 10]. The basic principle is that entity A looks up candidate entities B and C and contracts these entities. Entity C looks up and contracts D, while D looks up and contracts G, and so on. Each look up activity involves an exchange with a repository service, which returns a list of candidate entities. Contracting involves a negotiation between the client entity and one or more candidate entities to establish a working relation.

Although the bottom-up approach is very adequate in generic systems, it also has a number of disadvantages, especially when applied in time- and safety critical systems. We point out the following drawbacks:

I. The entities spend time discovering and contracting required services, although there is no guarantee the outcome of this process will be satisfactory. The client entity will not proceed until a server entity has been contracted; this will not happen before the server entity in turn has contracted the services it requires. When the chain of required services is long, the process of establishing a solution will take a lot of time. If at some point the subcontracting has no satisfactory outcome, the whole contracting process has to start from scratch at that point, with a different candidate.

II. No single entity possesses an overview of the entire solution, which means alternative solutions are not considered, except on a bilateral level. There may be far better overall solutions than the one eventually selected, but due to the local choice of a (cheap) peer, this solution will never be available.

III. The bottom-up approach lacks global overview. Coordination is guided by a first-come-first-served principle, which implies the decisions made for one solution might heavily impair solutions for other needs or even render these needs unachievable. Brokerage remedies this drawback to some extent, but at the same time introduces the drawbacks of centralized processing.

IV. When a solution is achieved, the bottom-up approach lacks sufficient flexibility to adjust elegantly when performance or service quality is less than required, or when new resources become available. Only a new contracting phase (starting in some subtree of the solution) can possibly improve performance or service quality.

V. Finally, in this approach each component is required to possess the capability to find, contact and negotiate a collaboration with peers. This not only results in components that
have a lot more capabilities than required by their purpose, but also limits the type of components that can participate in the DCS to those that possess identical behavioral capabilities.

The SMDS approach attempts to avoid the drawbacks identified above. In section 6 we will evaluate this.

4. SMDS APPROACH

The SMDS approach is based upon the delegation of planning, execution, monitoring and federation responsibilities to dedicated system segments. The Planning segment executes all planning activities, based on Needs. It produces a partial plan, the MasterPlan, which is used by the Instantiation segment to determine an allocation of the tasks to the available resources. This allocation is based upon a topology description provided by the Federation segment and status data provided by the Monitoring segment. The Monitoring segment monitors and evaluates the progress of the MasterPlan and compares this progress with the initial requirements, as stated in the Needs. The Federation segment joins and separates the platforms in the collaborative system, federates servers into service networks and provides a topology description to the Instantiation segment.

These four segments form the self-management capability of a self-managing DCS. In combination they coordinate the behavior of the DCS, by instantiating and terminating processes, allocating tasks to processes and interconnecting processes. In figure 2 the segments and the information they exchange are depicted. We will now closer investigate these four segments.

4.1 Planning segment

In response to a service or information request (a Need), the Planning segment calculates Solutions to satisfy that Need. Each Need possesses a priority, a cost budget and a time-validity. From the Solutions the Planning segment produces a MasterPlan, which is a combination of Solutions, satisfying all current Needs. The MasterPlan is annotated with global constraints of the DCS and non-functional requirements stated in the Needs. The MasterPlan is delivered to the Instantiation segment for execution.

The Planning segment consists of a knowledge base service, reasoning service and masterplan service. The knowledge base service is intended to store and manipulate resource and domain knowledge. The reasoning service composes Solutions for Needs. The masterplan service compiles a MasterPlan from the proposed Solutions.

4.1.1 Knowledge Base Service

The knowledge base service maintains a knowledge base, where it stores, retrieves and manipulates knowledge used in the SMDS planning process. The knowledge in the knowledge base is organized in three categories:

- Domain knowledge, documenting relevant terms and relations in an application domain. Domain knowledge can take for example the form of an ontology or another knowledge construct.
- Product knowledge, which defines a resource, its attributes, capabilities, associated costs and requirements. This knowledge could for example be derived from design documents used to develop the system, like UML models ([17, 21]).
- Resource information, like status and availability information of the resources in the DCS. This information can originate from the Monitoring segment, the Instantiation segment or the resources themselves.

In principle the knowledge base service is a distributed service, where each service provider can hide or reveal the knowledge regarding resources to the rest of the system, depending on the actual policy of the knowledge base server.

4.1.2 Reasoning Service

The reasoning service determines Solutions for a Need, based on the knowledge made available by the knowledge base service. A Solution is a composition of resources, where each resource is assigned a task (or a number of tasks), in combination achieving the Need. Each Solution has a value, which is determined by adding the cost of the participating resources and their actions.

As a possible approach to determine Solutions for a Need, consider the case where the reasoning service translates the Need to an start-goal, where a goal is a set of functional requirements and conditions. The calculation uses an ordered list of partial Solutions, which contain the set of actual goals, a set of tentative allocations and the value of the Solution. The reasoning service extracts the first partial Solution in the list, and retrieves abstract capability descriptions of resources matching one of its goals from the knowledge base. Each description retrieved is used to build a new partial Solution, tentatively allocating the described resource, and adding its requirements to the set of goals. The value of the new partial Solution determines its place in the list of partial Solutions. Partial Solutions which are too expensive are discarded. The partial Solution which has been processed is discarded and next iteration starts. Eventually, if solutions exist, the reasoning service will find partial Solutions with an empty set of goals, which constitute Solutions. Note how this approach closely resembles the GRAPHSEARCH algorithm proposed in [20]. If no solutions exist, the reasoning service will run out of partial Solutions. In that case, it notifies the masterplan service the Need cannot be achieved within the stated constraints, or that the DCS lacks capabilities.

4.1.3 Masterplan Service

The masterplan service produces the MasterPlan of the DCS. In general, there will be multiple Needs simultane-
the MasterPlan; the design of the masterplan service should be determined by the reasoning service, and manufactures a combined solution, by selecting one Solution for each Need. Some Solutions may be mutual exclusive, yielding conflicts in a combination. Other Solutions may be able to share services, thus reducing the overall cost and eliminating resource conflicts. A combined solution without conflicts is called a valid combination. For each valid combination a cost value can be determined. Different valid combinations may have different cost values. The masterplan service selects the cheapest valid combination to become the MasterPlan.

The cost of a MasterPlan is defined by global criteria imposed on the DCS. Cost can be defined, for example, in terms of resource expenditure or bandwidth consumption. In principle, one can define the global criteria dependent on time or context, thus enabling the possibility to steer the behavior of the DCS according to an actual cost strategy.

The cost of a MasterPlan depends on the Solutions it includes, but in general it is not just the sum of the cost-values of the included Solutions. For example, when the MasterPlan is a combination of two Solutions which share a common functional branch, the cost-value would be the cost of one Solution and the additional cost of the non-shared tasks in the other Solution, as depicted in figure 3. Technology may be derived from results achieved in operations research, to build satisfactory optimization algorithms, for example using constraint satisfaction or linear programming algorithms.

Figure 3: A combination reducing cost

Over time, the set of actual Needs varies. This implies a MasterPlan has a limited life-span. The masterplan service is responsible for the update of the MasterPlan, removing obsolete branches or adding new ones. Three types of event will trigger the masterplan service to determine a new masterplan:

- A change in the set of Needs. Either the appearance or the disappearance (expiry) of a Need will render the current MasterPlan out of date.
- A change in the set of Resources. When new resources are added to the DCS, potentially new Solutions can be found, yielding a possible improvement on the current MasterPlan. When platforms disappear, the masterplan service has to check whether the remaining resources adequately fulfill the remaining Needs.
- A notification from the Quality of Service management service (see section 4.3.3), indicating the current MasterPlan is inadequate.

In general, there will be a cost associated with changing the MasterPlan; the design of the masterplan service should ensure the expenditure due to MasterPlan changes are kept as low as possible.

4.2 Instantiation segment

The Instantiation segment contains a load-management service and an instantiation service. The purpose of the Instantiation segment is to create a run-time collaboration of processes, as defined by the MasterPlan. In the Instantiation segment a separate process gateway service could be identified, which enables migration of processes from one host to another. This process gateway service is used by load-management and (authorized) mobile processes. For the sake of simplicity, in this paper the gateway service is regarded as an extension to the instantiation service. Using the basic start-, stop- and signal- functionality of this service, a protocol to migrate software processes between hosts can be established [25].

4.2.1 Load-management Service

The load-management service carries out the MasterPlan. It examines the requirements of the resources allocated in the MasterPlan, and matches this against the available hardware facilities. It then issues task descriptions to the instantiation service. Each task description contains detailed information about a process or service to be started, stopped or modified. Each process is identified by a global unique ID.

The load-management service is process oriented; it assumes every resource is a software process or is represented by a software process. At the lowest level this process may be a driver or controller-daemon, while (at the highest level) human beings can be seen as represented by their interface processes.

The load-management service bases allocation decisions on static information regarding the capabilities of a host node and the hardware resources attached to it, as well as dynamic information from the Monitoring segment. This dynamic information includes information regarding the node’s health, available bandwidth (processing, memory or communication bandwidth) and so on. If the circumstances allow it to do so, the load-management service can reallocate a process to another host. It will do this when a high load threatens the success of the MasterPlan, or when the load-manager receives a notification from the Quality of Service management service (see section 4.3.3).

4.2.2 Instantiation Service

Each host available to the DCS contains an instantiation server. This server is responsible for starting, stopping and sending (control) messages to software processes. The instantiation service obtains instructions regarding processes to start, stop or send messages to from the load-management service in the form of task descriptions.

4.3 Monitoring segment

The aim of the Monitoring segment is to check whether the results produced meet the required quality. The activities in the Monitoring segment revolve around obtaining performance and progress data, and comparing this data with the requirements in the MasterPlan. When the performance is insufficient, it is the responsibility of the Monitoring segment to decide whether measures should be taken to improve that situation and which service should take them. The Monitoring segment contains a monitoring service, a feedback service and a Quality of Service management service.
4.3.1 Monitoring Service

The monitoring service is composed of a number of monitoring servers, at least one on each host node. A monitoring server frequently samples the status of its host. This sampling includes obtaining data regarding load, available memory and communication bandwidth. The samples are normalized and distributed throughout the rest of the system.

A second task of the monitoring service is to maintain the communication log of the host node. This communication log collects statistics of communication data of the processes allocated on the host node. On request of the Quality of Service management service, it retrieves the data from the communication log, resets the log, and assembles a report for the Quality of Service management service.

4.3.2 Feedback Service

While the monitoring service monitors the performance of the MasterPlan by sampling externally visible characteristics of the system, some (intelligent) processes may be able to express relevant information in much more detail. To make use of this opportunity, intelligent processes can formulate Quality of Service reports to express complaints about their peers or environment. The feedback service collects these Quality of Service reports and presents them to the Quality of Service management service.

4.3.3 Quality of Service Management Service

The management of the Quality of Service (QoS) rendered by the DCS is of paramount importance. It is not sufficient to formulate a MasterPlan for a given set of Needs; also the effectiveness and efficiency of the MasterPlan should be examined and optimized. Every autonomous system must reflect on the situation in its environment, and consider its options. For SMDS that action takes place in the Planning segment. Furthermore it must evaluate the progress and the results of its actions, to enable it to adjust its behavior in order to achieve the desired goals. That evaluation is carried out by the QoS management service.

The QoS management service evaluates the behavior of the managed processes in the MasterPlan. It does this by diagnosing the communication reports it obtains regularly from the monitoring service. From this data it can estimate two important health indicators: the process health and the communication service health. The health of each process is estimated by comparing the amount of data received and sent with the IO-model of the process in the knowledge base. The health of each communication service is estimated by matching the amount of data items sent and received using that service. Furthermore, the QoS management service retrieves and evaluates the Quality of Service reports from the feedback service.

Depending on the outcome of that evaluation the QoS management service decides whether the current MasterPlan and its execution are acceptable. If not, the QoS management service will choose one of two options: it will either notify the load-management service to reallocate processes to improve performance, or notify the masterplan service to create a new MasterPlan.

4.4 Federation segment

The Federation segment contains just a federation service. The federation service is intended to hide boundaries between platforms and to provide easy access to distributed services. A service can be seen as a functionality that is offered by a network of servers. There may be just one server available in the entire system or many. Each server contributing to the service provides the same functionality, although each server may be implemented differently.

Although the servers are distributed, each client should receive identical service. This introduces the problem of providing a uniform, unambiguous access to a service. Common approaches include addressing all known servers in turn with the same request, issuing a 'call for proposal' or broadcast a request (see [12, 24]).

The SMDS federation service deals with this problem by introducing a suite of federation functionality. Equivalent servers, for example name-servers or knowledge base-servers, are interconnected by the federation service, thus forming an ad-hoc service network. Each server obtains federation functionality from a federation library [26]. This federation functionality includes a number of schemes to forward incoming requests over the service network. After forwarding the server processes the request. The responses to the forwarded requests are collected and valued by the forwarding server, after which it assembles its response.

4.4.1 Federation Service

For each relevant service, the federation service connects all servers to establish a service network. An inventory of servers is stored in a map, called a service topology. For each relevant service such a service topology can be established. The federation service notifies equivalent servers to engage in a mutual federation, inherently crossing platform boundaries. However, the federation service should not connect every server with every other server, but construct a network that is robust for link failure, but still has a low degree of connection, by applying appropriate ad-hoc network construction technologies.

5. EXAMPLE

In this section we will explain the SMDS functionality by examining how the SMDS approach would construct a Solution for a Need for an assessment service as defined at the end of section 2 and depicted in figure 1.

Suppose we have an SMDS based management system, which possesses knowledge about constructing assessment solutions, composed of sensors, data-sources, image extractors, recognition services and filters. When a Need for the assessment service is presented to the system, the masterplan service will instruct the reasoning service to construct Solutions for this Need. The reasoning service will find in the knowledge base, that a service provider A exists, that is capable of the requirements, but requires in turn recognized observations as well as aggregated intelligence information. Iteratively, the reasoning service will determine that the required recognized observations are produced by server C, provided it receives filtered observations. Eventually, the reasoning service will produce a number of Solutions, including a Solution which can be represented as in figure 1.

The Solutions are presented to the masterplan service, which will select the best fitting Solution with respect to the rest of the duties of the DCS, and the relative importance of the Need. This selection will have implications for the current MasterPlan, and the masterplan service will distribute an new version of the MasterPlan.
The load-managent service examines the new MasterPlan, and compiles a series of task descriptions, to be carried out by the instantiation service, that will put the new MasterPlan into effect. The task descriptions are sent off to the instantiation servers, who will effectively instantiate and configure the required services, and carry out the other adjustments described in the MasterPlan.

The important issue of this example is that the business activities of the DCS where not influenced, until the masterplan service had determined a new MasterPlan. All planning activities took place 'behind closed curtains'. When finally the coordination activities commence, there is a clear view of the desired new configuration. All processes and services which were not affected by the new Need were left alone, while processes that were affected by it, were re-configured or re-allocated only once, immediately reflecting the new MasterPlan.

6. DISCUSSION

Although the model of the SMDS concept entails a centralistic approach to system’s management, there is no reason for the services in the SMDS segments to be centralistic in their design as well. The SMDS services can in all cases be implemented using a distributed design of networked servers and an adequate data sharing facility. In general, each platform will feature the entire suite of SMDS services.

Decision making in SMDS is based on Boyd’s famous OODA-loop; the same loop underlies MAPE [15]. The main difference between SMDS and the MAPE effort at IBM is that the SMDS management suite is a dynamic sub-system of systems managing a dynamic set of resources. The SMDS concept hence has the explicit notion of federation, applied of systems managing a dynamic set of resources. The SMDS concept is a design. The MAPE effort at IBM is that the SMDS management suite is a dynamic sub-system of systems managing a dynamic set of resources. The SMDS concept hence has the explicit notion of federation, applied of systems managing a dynamic set of resources. The SMDS concept is a sub-system of systems managing a dynamic set of resources.

Decision making in SMDS is based on Boyd’s famous OODA-loop; the same loop underlies MAPE [15]. The main difference between SMDS and the MAPE effort at IBM is that the SMDS management suite is a dynamic sub-system of systems managing a dynamic set of resources. The SMDS concept hence has the explicit notion of federation, applied of systems managing a dynamic set of resources. The SMDS concept is a design. The MAPE effort at IBM is that the SMDS management suite is a dynamic sub-system of systems managing a dynamic set of resources. The SMDS concept hence has the explicit notion of federation, applied of systems managing a dynamic set of resources. The SMDS concept is a sub-system of systems managing a dynamic set of resources.

In the bottom-up approach, the fact that clients have to discover and contract the services they desire, makes the clients spend time on coordination, without a guaranteed successful outcome (I). Furthermore will the entities possess far more capabilities than they actually need. This on the same account excludes entities with a different approach to coordination (V). The SMDS concept avoids spending the time of the resources on coordination activities by allocating the coordination responsibility explicitly to a dedicated set of services. This allocation enables the DCS to keep working on the MasterPlan, while the management suite determines whether and what adjustments are required. This relieves resources from discovery and coordination capabilities, and opens up the DCS to many types of resources. On the other hand, any (intelligent) resource possessing the required skills can participate in all relevant SMDS segments.

In the bottom-up approach, the lack of local overview might cause inferior solutions to be established (II), whereas the lack of global overview in the bottom-up approach can prohibit the selection of solutions that improve the overall DCS performance (III). In contrast, the SMDS concept investigates many Solutions before any budget is spent or resource is activated. The global overview enables the weighing of the relative importance of Needs, enabling an informed decision about allocating or sharing services.

The bottom-up approach suffers from inflexibility to take advantage of the situation where new resources or bandwidth become available (IV). In the SMDS concept, the MasterPlan can be adjusted after introspection. Introspection is triggered on an event basis by changes in the set of Needs, the set of resources or on notification of the Quality of Service management. This yields sufficient flexibility to the self-managing DCS to adjust when the opportunity arises.

Finally, the SMDS management services can detect that the DCS possesses insufficient capabilities or bandwidth to deal with all the current Needs. This information can be relayed to the owner of those Needs for which no sufficient capabilities or bandwidth is available. In other cases, this information can be forwarded to an entity who can decide to allocate more resources or bandwidth to the DCS.

7. CONCLUSIONS AND FUTURE WORK

In this paper we have documented a possible approach to self-configuration and self-management of dynamic collaboration systems, SMDS, composed of four segments. We have shown what functionality the four segments of SMDS contain and how they interact, and we have explained how coordination, instantiation and evaluation are established in the SMDS concept. The SMDS concept is aimed to provide reliable self-management in time- and safety critical dynamic collaboration systems. We have shown how solutions are created in the SMDS concept, and how this differs from conventional approaches. We have argued that a number of drawbacks in current approaches can be avoided using the SMDS concept.

The benefits of the SMDS concept still have to be validated in practice. Therefore, alongside the development of the SMDS concept, we have been implementing a prototype of such a self-configuration and -management system, called COMPASS (Configuration, Organization and Management Prototype for Autonomous Systems of Systems). COMPASS is intended to be a very pure SMDS implementation, in the sense that each functionality documented here is encapsulated in a separate software program. COMPASS is now ready for testing and evaluation at the DECIS Lab.

We are very interested how COMPASS behaves in a truly dynamic environment, where (groups of) platforms enter and leave a DCS. We are also very interested to see how well the COMPASS system scales, and whether encountered limitations are inherent to the SMDS concept or the COMPASS implementation. We will report on results as soon as these become available.

8. ACKNOWLEDGMENTS

The SMDS concept is the result of research conducted at Thales Nederland B.V. Also, the research in this paper is part of the COMBINED Systems project, supported by the Dutch Ministry of Economic Affairs, grant number: TSIT2003, conducted at the DECIS Lab in Delft.

9. REFERENCES


