Applications of Self-Organizing Systems in Maritime Environments

Martijn Neef, TNO Physics and Electronics Laboratory,
The Hague/The Netherlands, neef@fel.tno.nl
Anthonie van Lieburg, TNO Physics and Electronics Laboratory,
The Hague/The Netherlands, lieburg@fel.tno.nl

Abstract

Maritime platforms are becoming increasingly complex to maintain and operate, due to rising demands in terms of operational capabilities, functional robustness and cost efficiency. Moreover, with an ongoing demand for reduced manning the need for more capable and clever control systems is obvious. Conventional automation approaches typically fall short on maintainability, robustness and scalability, and becoming inadequate to capture the complexity of modern maritime environments, with their abundance of data and dynamic nature. To resolve some of these concerns, the Command & Control group of TNO-FEL is researching the application of self-organizing systems theory for various types of maritime decision support systems. These biologically inspired systems are typically distributed, fast and reactive, computationally very simple, and above all, possess autonomous self-organizing properties that we can employ to build new platform control systems. The most typical example of self-organization are the so-called 'ant-based algorithms', which are a class of algorithms that mimic cooperative behavior of real ant behavior to achieve complex computations. The structural simplicity of these systems, together with their self-organizing capabilities make them a very interesting approach for certain types of computational challenges. We will demonstrate their use in damage control systems, platform management systems and decision support systems, and will describe how we envision their use in future control systems.

1. Introduction

As is the case with many domains, the maritime domain is becoming very complex and hard to deal with from an automation point of view. Maritime platforms are becoming increasingly complex to maintain and operate, due to rising demands in terms of operational capabilities, functional robustness and cost efficiency. Moreover, with an ongoing demand for reduced manning the need for more capable and clever automation is obvious. Conventional automation approaches typically fall short on maintainability, robustness and scalability, and are becoming inadequate to capture the complexity of modern maritime environments with their abundance of data, their dynamic nature and the complex inter-system relationships that characterize modern maritime platforms. To resolve some of these concerns, the Command & Control group of TNO-FEL is researching the application of self-organizing systems theory for various types of maritime support systems. This paper shortly introduces the basic principles of self-organization, and describes how they can be applied in practice. To this end, we introduce two case studies in which we utilized the concept of self-organization and elaborate on the general benefits of the approach.

TNO Physics and Electronics Laboratory (TNO-FEL) has a longstanding relationship with the Royal Netherlands Navy, and may be considered as being one of its chief research and development centers. TNO-FEL has contributed to numerous innovations on sensor technology, platform system control and weaponry, and also provides advisory services on Command, Control, Communication and Information (C3I) issues. TNO-FEL also has strong ties with commercial maritime industries, with whom in close cooperation innovative solutions are being developed in the field of decision support systems and distributed control.

2. Self-organizing systems

This section gives a short introduction to self-organizing systems and their applicability as computational optimization algorithms.

2.1 Introduction

Self-organization is a very much used and misinterpreted term in computer science nowadays. Researchers and software developers are eager to label their products as 'self-organizing' because it adds a certain mystical flavor to the functionality of a system: that of being able to 'magically' configure and manage itself without any human intervention. In reality, engineers usually refer to the capability of a system to manage itself under fairly known circumstances, such as the capability to detect certain anomalies and apply behaviors to correct these failures. This is not self-organization, but rather smart programming. To be precise, self-organization is not a technology, but primarily a way of thinking about dynamic, adaptive systems, which applies to biological, social, chemical and physical systems. Self-organization is about system structure appearing without explicit steering from outside of the system. It is about emergent global system behavior that stems from local interactions between the different entities that form the whole system, without explicit representation of these global patterns on the level of the individual components. The main characteristic of all these systems is their ability to achieve complex collective tasks and (organizational) structures with relatively simple individual behaviors, without the need for central control or hierarchy, without central models or internal representations of the environments are situated in.

An often used example to illustrate self-organizing systems is the behavior of ant-colonies, *Dorigo and Stützle (1996).* Although single ants presumably lack knowledge of their social organization, they jointly display extraordinary complex behavior, such as their food finding behavior. They achieve their global food searching behavior through local interactions, by means of deploying *pheromone* trails. Ants release pheromone, an aromatic substance, on their way to food. Initially ants have no idea of where food is in the environment, so they wander randomly, but meanwhile leave a pheromone trail. If the ant finds food, it will wander back over its own trail, and again release pheromone. Consequently, the trail will increase in intensity. However, pheromone will evaporate over time. So, the longer the path to food, the weaker the trail will be. Ants are sensitive to pheromone, and will follow existing pheromone trails if they come across them, with a tendency to follow the strongest trail. So, if an ant finds a good, nearby food source, his trail will be picked up by other ants (since it has still has a high pheromone intensity because of its proximity), who will also release pheromone on this trail towards the food. As more ants use a particular trail, the pheromone concentration on it increases hence attracting more and more ants. Conversely, less preferable paths will not be replenished and will evaporate. This process is executed in parallel by the population of the ant colony, resulting in a swift and efficient exploration of the surroundings of the colony.

The ant colony behavior is a very illustrative example of self-organization: the appearance of structure or pattern without an external agent imposing it, *Heyligen (2001).* There is no central controller telling the ants where to go, or what to do. Moreover, none of the ants have a notion of the state of other ants, and their interaction is indirect, by means of the environment. The ant colony has an innate capability to organize *itself,* apparently without being capable of maintaining any internal state, or any other cognitive capabilities. Ant colonies exhibit a form of *swarm intelligence.* Swarm intelligence is the general term which is used to describe the form of intelligent behavior that natural systems exhibit. Swarm intelligence takes advantage of the synergistic effects caused by the interactions between the actors (i.e. the elements of which a systems is composed), their environment, and the traces the actors leave in their environment. The environment essentially serves as the memory for the systems, in combination with the physical characteristics of the actors. This characteristic does not deny such systems any ability to learn, but it characterizes the way these system achieve global goals, i.e. by efficiently using the environment they are situated in. Many more example of swarm intelligence exists in nature, such as the building of complex structures by swarm of insects (like bridges, nests, and chains) and display of many complex tasks (like defense, foraging, social care-taking, labor allocation and so on).

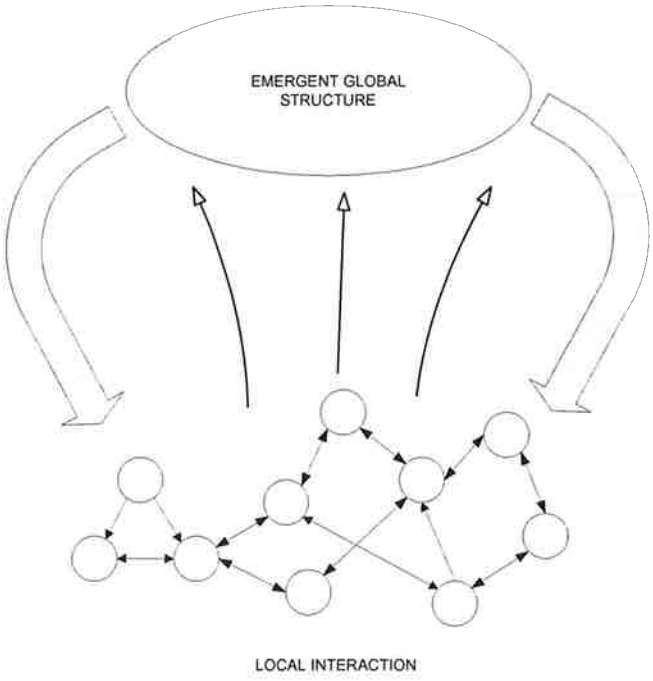

Fig.1: Emergent structure coming from local interactions (after Langton in *Lewin (1992)*)

In a broader context, self-organization is a fundamental principle of most natural systems. The main scientific theory related to self-organization is *complexity theory*, which states that 'critically interacting components self-organize to form potentially evolving structures exhibiting a hierarchy of emergent system properties', *Lucas (2003)*. In other words, complex systems (i.e. systems that are made of several closely connected parts) can, under certain circumstances, spontaneously evolve from one state to another. In terms of non-linear dynamics: any system that moves to a fixed structure (i.e. a stable pattern) can be said to be drawn to an attractor. A complex system can have many attractors and these can alter with changes to the system interconnections (mutations) or parameters. In effect, researching self-organization is equivalent to investigating the attractors of a system, their form and dynamics. Changes to a system can instigate self-organizing behavior by allowing the exploration of new state space positions. In short: if we perturb a complex self-organizing system (by for instance changing its structure or its surroundings) then the system as a whole will tend to find new stable states, and move away from unstable states along certain trajectories. Although complex systems theory is too extensive and outside the scope of this paper to describe further in this paper, it is important to note that the self-organizing properties of the applications we will describe further on have their theoretic roots in complexity theory, and in order to fully understand their potential knowledge of this line of research is essential. *Bak et al. (1988), Bak and Chen (1991), Lewin (1992), Kauffman (1993), Holland (1998)* and *Johnson (2001)* provide concise overviews to the domain of self-organization and complexity theory.

2.2 Self-organization as a Computational Approach

Many new technologies, such as evolutionary algorithms, multi-agent systems and neural networks have their roots in theoretical biology, and self-organization is no exception. It might not come as a surprise that many have quickly found their way into the domain of computer science. Their ability to tackle complex problems with only very basic rules has inspired many researchers. The use of biologically inspired computational models is twofold: they may help us in understanding biological phenomena, and they may enable new computational approaches to tackle complex problems. One of the main reasons why biological models of self-organization have attracted so much interest from the engineering community is their inherent robustness and adaptive capabilities. Anyone can confirm the robustness and adaptive nature of insect colonies: the disruption of a part of colony will most likely not be enough to get rid of them. Put an object in the middle of an ant trail, and not before long the

ants will travel around that object, and continue on their way. These are capabilities that are very much desirable in engineered systems, and hard to obtain using conventional model-based approaches. Moreover, coordination and control have always been complex issues to resolve in distributed systems, and as the world is seeing more and more networked applications and environments, these issues are gradually becoming truly obtrusive.

Parunak and Brueckner (2004) hands us five domain characteristics which might indicate when self-organizing behavior (or rather swarm intelligence) might be appropriate for engineering purposes: *discreteness*, *deprivation*, *distribution*, *decentralization*, and *dynamism*. Thus a swarming approach may yield benefits in domains that are discrete (i.e. domains that consist of discrete elements), that have strict constraints on resources (such as lack or limitation of communication means or available computing power), that need computational entities to be distributed (such as in sensor networks or telecommunication networks), that are not bound to centralized control requirements (and thus may benefit from distribution of system functionality), and that are dynamic in nature (i.e. may change over time). Not surprisingly, most successful applications of self-organization have been in domains that adhere to these characteristics. The most successful real-world application of swarm intelligence has been in network routing, where the dynamic and distributed nature of the swarming approach is particularly helpful. Ant colony based routing methods have been shown to outperform conventional routing methods in various routing domains, such as telecommunication networks, *Schoonderwoerd et al. (1997), Bonabeau et al. (2000),* and route planning (see for instance ant-based optimization of the Travelling Salesman Problem by *Colorni et al. (1992)*. Other domains where self-organizational principles have been successful and actively being research include coordination mechanisms for multi-robots systems and other agent-based systems, *Kube and Zhang (1993), Pagello et al. (1998), Parunak and Brueckner (2001)* and many others, adaptive task allocation and resource allocation, e.g. *Bonabeau et al. (1998), Cicirello and Smith (2001), Eymann et al. (2003).*

2.3 Self-organizing Control Systems

We have briefly introduced the fundaments of self-organization and their use as a computational approach. There are many forms in which biological self-organization principles can be applied from a computational point of view. Aside from the well-known ant-optimization algorithms, there are many more example of biological self-organization models making the transition to computational sciences. For instance, neural networks and genetic algorithms are prime example of artificial systems capable of self-organization. Also, many characteristics of self-organization can be found recognized in contemporary multi-agent systems and distributed AI research. However, this paper is geared towards the use of a specific type of self-organization, and its use for a specific type of application. The stereotypical name we have adopted for our purposes is that of *self-organizing control systems (SOCS):* support systems that have control over and have responsibility for other, often physically embodied, systems and which make use of self-organizing properties to achieve that task. SOCS systems have the following characteristics:

- A SOCS can be defined as a group of interacting components that is functioning as a whole and distinguishable from its surroundings by its behavior.
- In a SOCS the various parts are laid out as to promote a specific function. The organizational layout of a SOCS is a critical attributing factor to the functioning of the system.
- In a SOCS self-organization is the evolution of the system into an organized form *in the absence of an external supervisor.*
- SOCS are tolerant to partial failures. Both the dynamic structure and distributed functioning enable graceful degradation and fault-tolerance.
- In a SOCS the functioning of the system as a whole is not explicitly programmed though the behavior of the entities that form the systems is defined by simple rules. Global behavior emerges from local interactions of these entities.
- SOCS are dynamic in structure. System parts may be removed from the system (e.g. in case of failure) or added to the system (e.g. ad-hoc replacements) at any time. Being open and

dynamically structured, the system does not need to be explicitly configured to accommodate for system changes.
- SOCS components rely on interaction (communication), and tend to make use of the simplest form of communication that allow the system to fulfill its aims.

In our research we primarily use the *cellular automaton* paradigm. A cellular automaton is a system made up of many discrete cells, each of which may be in one of a finite number of states. A cell or automaton may change state only at fixed, regular intervals, and only in accordance with fixed rules that depend on cells own values and the values of neighbors within a certain proximity. A classic example of a cellular automaton is Conway's Game of Life, *Gardner (1970).* The Game of Life is a set of simple rules that governs the behavior of a two-dimensional cell pattern, that have an amazing potential to generate complex and interesting new patterns depending on the initial pattern. Each generation switches cells on or off depending on the state of the cells that surround it. This indirect interaction (the status of a cell depending on the status of its neighbor) is a typical example of local interaction causing global patterns without any central coordination, thus of self-organization. The following case studies illustrate the SOCS concept in more detail.

3. Self-organizing Systems in Maritime Environments

The Command and Control group of TNO-FEL is applying the notion of self-organization in various domains such as traffic management, decision support and platform control systems. This section described two of these efforts which are situated in the maritime domain: an example of a platform control system, and an example of a damage control system. Additionally, we illustrate the use of self-organizing systems as an integral part of a control system.

3.1 Distributed Platform System Control

Major issues in ship control system development are achieving *robustness* and *flexibility,* the former implying the ability of a control system to sustain a certain amount of damage without losing operational capabilities, and the latter implying the ease with alterations in the platform system can be accommodated in the control software. In order to optimize robustness and flexibility, we have developed a fully distributed, autonomous control systems approach, which employs self-organizing properties to sustain maximum platform system performance. As a case study for our research we have used a generalized model of the chilled water system (CWS) aboard a frigate. A chilled water system provides cooling to several frigate systems by transporting chilled water throughout the ship by means of pipes. The water within the CWS itself is cooled by seawater. A typical chilled water system consists of little more than pumps, valves and pipes. Aboard modern ships these individual components are controlled remotely from a damage control center. When an incident occurs, various alarms are sent out by sensors and monitoring equipment mounted on systems and the CWS itself.

For a chilled water system the most crucial function is proper distribution of cool water amongst various instruments. To fulfill this goal it should maintain its physical integrity as well as it can. The only means available to the control system is the opening and closing of valves, which enable the rerouting of water and cooling liquids if required. When an incident occurs that causes physical damage to the system, it is paramount to act as responsively and effectively as possible to limit the damage by prompt *isolation* of the affected sections and fast consequent *rerouting* of the water flow. In contrast to the conventional approach, where human operators (semi-)manually control the status of valves in case of an emergency, we have relocated the control of the chilled water system to an autonomously functioning architectural layer that is directly interfaced to the physical platform system. This layer, simply denoted as the *reactive* layer, is in charge of actually configuring the platform system (i.e. utilizing the actuators, such as valves, pumps, etc.) and sensing the status of the platform system (i.e. reading sensor values). An important aspect of this layer is that it is fully distributed, not only functionally, but geographically as well. Each functional entity in the platform system is represented in the reactive layer by a separate agent. This collection of agents is responsible for carrying out a kind of *reflexive* behavior: an immediate corrective reaction in response to failures

or damage of the platform system. It contains logic that enables it to quickly and autonomously change the system configuration as to minimize damage.

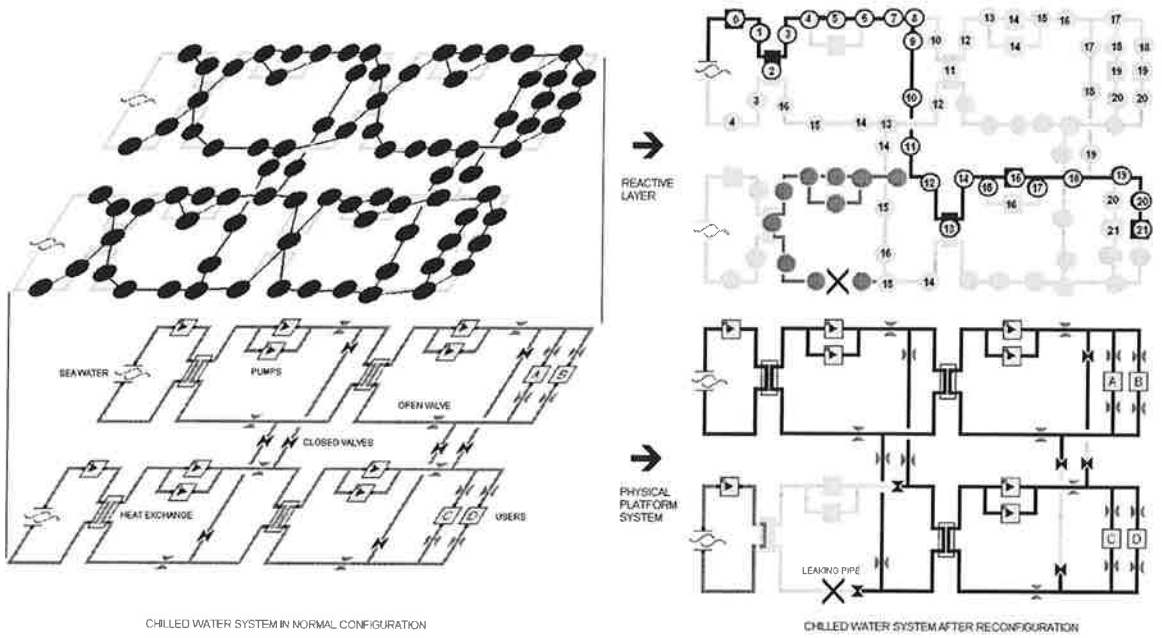

Fig.2: Illustration of the hybrid gradient routing method. On the left the agent layout with respect to the platform system, on the right the functioning of the routing algorithm.

Detecting leakage may involve knowledge of water pressure and flow in the pipes. Declining values for both values indicates that the configuration has changed, possibly because of a leak. To detect a leakage a pressure or flow meter should be placed at each few meters of pipe, covering the entire system and all of its components. Once a leakage has been detected, the first thing to do is close off the leaking part of the circuit. For this we use the following rules:

```
1.  WHEN PRESSURE OR FLOW FALLS
        SET ALARM FLAG
        SEND ALARM TO NEIGHBOR
2.  WHEN RECEIVED ALARM FROM NEIGHBOR
        SET ALARM FLAG
3.  WHEN ALARM FLAG IS SET:
        SHUT DOWN EQUIPMENT IF CAPABLE
        CLOSE VALVE IF CAPABLE
        IF INCAPABLE OF CLOSING PIPE, THEN SEND ALARM TO NEIGHBOR
```

All agents in the reactive layer behave to these rules. The agent closest to leaking pipe will detect the first fall of pressure. An alarm will be dispersed along the circuit away from the leak. The alarm stops where it meets the valves that border the circuit. These valves will be closed. The leaking circuit is now separated from the rest of the network and equipment on it has shut down.

Now that the network has changed it is necessary to reconfigure the rest of the valves to guarantee the cooling of other systems. Is it is clear that reconfiguration is an urgent task and ideally should be triggered as a reflex action. One might wonder whether the reconfiguration function can do without complete knowledge of the network. We agree on this, but we think that this knowledge should available in fully distributed form (i.e. there is no central location where there is complete knowledge about the state of the network). To implement the reconfiguration process as a reflex action, it needs to be broken down to simple procedures for which only local knowledge of the network is required. In this form the function can be distributed over the network. We achieved this by using a hybrid distributed gradient descent method.

After a leakage has been detected, the valves that shut off the leaking circuit will send an reconfiguration request to their neighbors by using the following behavioral rules:

```
4. WHEN CAPABLE OF CLOSING A VALVE
        AND ALARM IS SET
        AND RECONFIGURATION FLAG IS NOT SET
                SEND RECONFIGURATION REQUEST
                TO ALL NEIGHBORS
                SET RECONFIGURATION FLAG

5. WHEN RECEIVED A RECONFIGURATION REQUEST
        AND ALARM IS NOT SET
        AND RECONFIGURATION FLAG IS NOT SET
                SEND RECONFIGURATION REQUEST TO ALL NEIGHBORS
                SET RECONFIGURATION FLAG
```

As a result the request is spread out over the network and all agents in the network will know that a reconfiguration will take place. To find a path from the users to the seawater we will use a hybrid distributed gradient descent method. This is implemented by the following rules:

```
6. WHEN CONNECTED TO SEAWATER INLET
        AND RECONFIGURATION FLAG IS SET
                SEND GRADIENT VALUE "0" TO OUR NEIGHBOR

7. WHEN ALARM FLAG IS NOT SET
        AND RECEIVED A GRADIENT VALUE
        AND OUR GRADIENT VALUE IS HIGHER
                SET OUR VALUE TO RECEIVED VALUE
                SEND GRADIENT VALUE "OUR VALUE + 1" TO OUR NEIGHBOR'S UPSTREAM
```

Each message holds a counter. The receiver will increase this counter before it forwards the message to its other neighbors, unless it has received a copy holding a lower counter earlier on. In this way the message is distributed over all functional components in the network, which creates a 'gradient' from the seawater to the users (the devices that depend on the delivery of chilled water, such as electronic equipment). Once the gradient reaches a user, it will send back a chilled water request. This request will follow the gradient downhill towards the seawater inlet:

```
8. WHEN CONNECTED TO SEAWATER INLET AND RECEIVED A GRADIENT VALUE
        SEND DELIVERY REQUEST TO THE NEIGHBORS HOLDING THE LOWEST GRADIENT VALUE

9. WHEN RECEIVED A DELIVERY REQUEST
        SEND DELIVERY REQUEST TO THE NEIGHBORS HOLDING THE LOWEST GRADIENT VALUE
        SET "ON PATH" FLAG
```

The seawater inlet will receive the delivery request and confirm the request. All agents that receive the confirmation and are on a delivery path will open their valve, while all others close.

```
10. WHEN RECEIVED A DELIVERY CONFIRMENT AND "ON PATH" FLAG IS SET
        OPEN VALVE IF CAPABLE

11. WHEN RECEIVED A DELIVERY CONFIRMENT AND "ON PATH" FLAG IS NOT SET
        CLOSE VALVE IF CAPABLE
```

This small set behavioral rules results in prompt reconfiguration of the CWS in such a way that, if there is an alternative route which the chilled water can follow (e.g. through cross-over pipes between chilled water zones) to reach isolated users, the network will configure itself to do so.

So, what is the use of implementing a distributed and highly decentralized system design when a centralized processing unit (like a diagnosis module) is still considered to be necessary? When a processing unit (e.g. an agent on the reactive level) cannot communicate with its neighbors, it can still handle local problems because it only needs local information to do so. What would happen when a

crewmember detects a leaking valve? That person would most likely try to fix the valve or even bypass the leak with hoses. In the latter case, the CWS will remain operational, but its architecture has changed and the affected valve is no longer part of the CWS model. This is where a distributed architecture has a clear advantage over a centralized system. The knowledge of a centralized system would need to be updated to capture the new condition, where a distributed architecture does not require a new model of the system because it never had one in the first place. Self-organizing control systems dynamically react to changes of the environment they are part of. Therefore, the approach is robust, highly scalable and adaptive.

The above approach has been demonstrated in both software and hardware demonstration, has shown that it can provide valuable support in emergency situations. Further developments include the embedding of such a self-organizing layer into a full control system setting.

3.2 Adaptive Evacuation Routing

Efficient evacuation procedures are a major issue aboard modern vessels. Especially with rising personnel and passenger capacities aboard modern ships, the increasing criticality of proper evacuation procedures is obvious, and several severe incidents over the past two decades have shown that this topic still deserves all the attention it can get. This section describes one of the efforts TNO-FEL is undertaking to increase evacuation performance: *adaptive evacuation routing.*

The main procedure of evacuation is to guide people away from an affected region into safe surroundings. The overall goal of evacuation procedures is to minimize the time it takes for all passengers and crew on board to move from the location where they were when the evacuation alarm was issued to safety (such as inside lifeboats or on a safe location on the vessel). On vessels all kinds of incidents could trigger evacuation procedures, such as fires, chemical incidents, malfunctioning systems or even impact damage on military vessels, and they can initiate at many locations around the vessel, such as engine rooms, cabins, kitchens or workplaces. Therefore, evacuation is a dynamical process: the actual progress of an evacuation is, aside from general, abstract evacuation plans, directed by the specifics of an incident, the ship geometry and ship orientation, and other contextual constraints. Consequently, optimization of the evacuation procedure is a complicated matter due to the amount of dynamic parameters involved. Fires may block main evacuation routes, or may interfere with the attack routes of damage control teams, routes may be blocked by fire, smoke or flooding or may be too small to carry the number of evacuees.

In this research we propose a fully distributed approach to evacuation routing. Our approach is in sharp contrast to most conventional approaches to evacuation routing optimization, which often use a centralized model-based approach, i.e. optimize evacuation routing using a complete model of the area. These model-based approaches cannot uphold in modern complex ships, and they are gradually becoming too complex to sustain and apply in practice. Also, we want the actual evacuation support towards the evacuees to reflect the dynamic nature of the process. We aim to create an adaptive evacuation routing system that guides evacuees to safety using dynamic signs and alarms. To achieve this behavior, we envision a network of low-cost smart devices that control sensors (e.g. smoke sensors) and actuators (e.g. direction signs or sound devices). Fig.3 illustrates the *evacuation support network* concept. In this example each section contains three types of devices: smoke detectors, direction sign and door status monitors. These devices are controlled by 'smart' network entities, that effectively represent the devices. The network layout mimics the actual corridor layout and gives us implicit knowledge about the layout of platform we are designing evacuation support systems for. We utilize this effect in our routing optimization procedure.

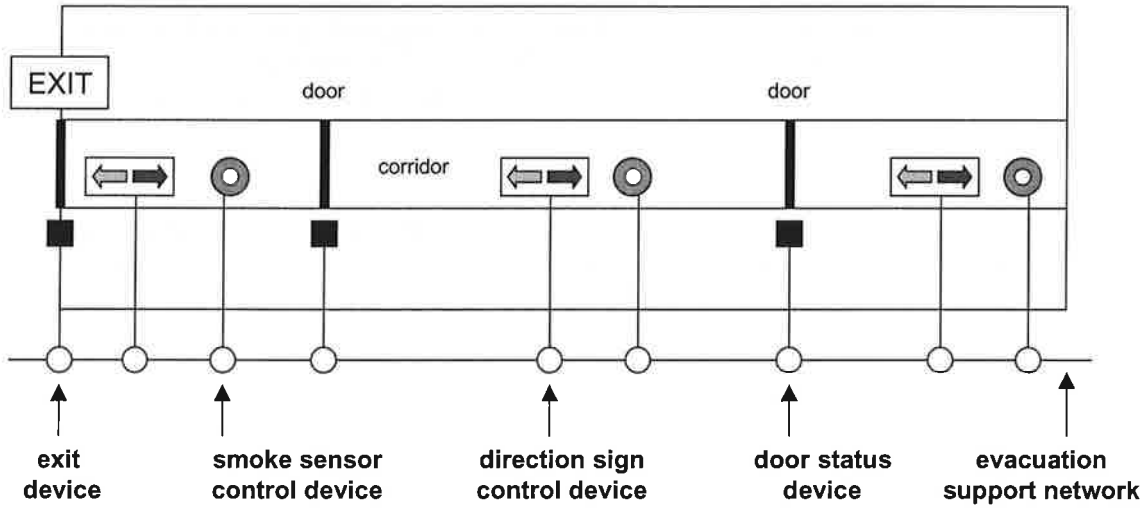

Fig.3: Illustration of the evacuation support network concept. Various sensors and actuators are controlled by networked devices to jointly provide evacuation support.

The evacuation routing process has two aims: provide optimal evacuation routes for passengers and provide efficient attack routes for damage control teams. Evacuation should have the highest priority, and should not interfere with damage control routes. The support system should inform personnel on location to where the nearest exit is, but taking into account that routes may be blocked by smoke, fire or any other intrusive circumstances. The routing procedure must also take capacity constraints for corridors and doors into account, and prevent congestion as much as possible. The nearest exit may therefore not be the most preferable exit. Our initial approach is to use a *gradient based* approach in which information propagates throughout the aforementioned network of sensors and actuators. On top of the gradient based approach we apply an ant-based optimization algorithm to deal with capacity effects of corridors and to initiate the exploration of alternative routes from any given location to an exit.

Let us examine in more detail how this would work. The evacuation process may be initiated by a alarm trigger. This might, for instance, be a smoke sensor going off. This alarm trigger subsequently sends out an evacuation initiation signal to the network. Each device in the network passes this signal on to its neighbor. As soon as exits receive this signal, they will start sending out their own signal. This signal is essentially a counter. Each device again passes on the signal, but increases the counter. In case a device already received a signal (and a counter), it only accepts the new counter if it has a lower value than the one received earlier. In effect the network becomes a kind of potential field, in which at any location the nearest exit can be found by following the gradient. This very simple algorithm is in essence enough to let direction signs point towards the closest exits, even if sections have become inaccessible due to hazardous conditions. For instance, if smoke sensors detect very high levels of smoke, their networked counterparts may block signal, and effectively prevent hazardous sections to be included in the evacuation routing. Add other types of sensors (such as camera devices or movement detection) and even more intricate constraints can be included in the evacuation routing process. As soon as the counter propagation process has finished (as may be signaled by another control signal), the direction signs light up towards the strongest signal (i.e. towards the neighboring network device with the highest counter).

The gradient-based method is good enough to handle routing, but is in this form not capable of capacity-based planning, which is essential if we deal with high numbers of evacuees. Including capacity constraints adds a dynamic aspect to routing, since we might need to disperse evacuees over multiple routes. The gradient always points towards a single exit. To this end, an ant-based optimization algorithm may be applicable in addition to the gradient based approach. We assume that it is possible to obtain information about the amount of evacuees around the vessel. This information might be obtained by using sensors, central services or even by using estimates based on general

information (e.g. time-bound information, locations of cabins and work areas, etc.). We also assume a new type of non-sensor bound network device capable of relaying information about the capacity of the corridor or section it represents. The idea of ant-based optimization (or rather ant-based exploration) is that multiple entities explore their neighborhood in parallel, and lay out gradually decaying trails while doing so, as we discussed earlier. At a certain location, perhaps a location where multiple routes to exists start, we initiate the ant-optimization process. A device sends outs a number of signals onto the network, where the number of signals is relative to the amount of evacuees that need to be guided towards exists. The signals represent 'virtual ants', and the network is the neighborhood they need to explore. The network itself keeps track on how long it takes for an ant-signal to reach an exit by means of updating the signal. As they explore the network (i.e. being passed around by the network devices) they encounter the aforementioned corridor-capacity devices. These devices impose an artificial delay on the transmission of the ant-signals if too many signals arrive at once, essentially simulating a congestion. Consequently, if a signal passes through a congested corridor, its 'time to exit' increases. This information is used to make the route less attractive to newly arriving ants by lowering the gradient on that particular path, implicitly guiding the ant colony towards alternative exists. In conjunction with adaptive control procedures and signaling devices that are capable of informing evacuees about multiple routes to exists, this approach would lead to a capacity-based adaptive evacuation routing system.

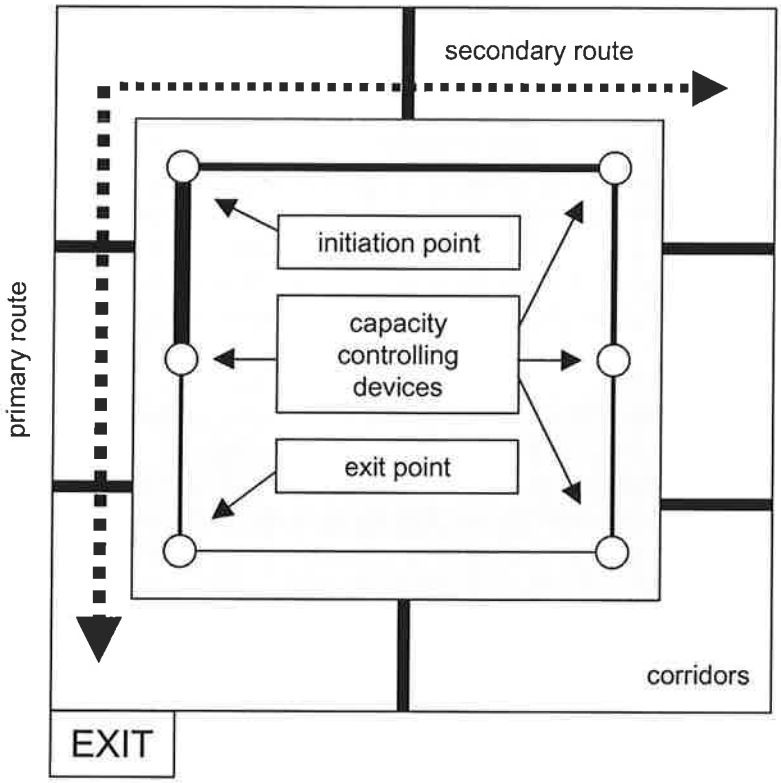

Fig.4: Network layout for a capacity-based evacuation routing system. From the initiation point 'virtual ants' explore the area, with a predisposition for timewise shorter routes. Capacity controlling devices may deliberately add a delay to the travel time of the ants, in case of crowding, leading to the activation of secondary routes

At the moment, the gradient based routing approach has successfully been simulated using an actual floor plan of a frigate. The ant-based optimization method we briefly discussed, that would enable capacity constraints, is being implemented at the moment for further examination. Nonetheless, the conceptual approach we have taken seems very worthwhile, both from a optimization point of view, as well as from a practical point of view. Even though there are very good routing optimization algorithms, we are looking for Ockam's Razor: of approaches, all other things being equal, the simpler one is to be preferred. Our approach does not require any global world-model, is robust, flexible and

adaptive. Furthermore, from a implementation and installation point of view, our approach yields even more benefits. The only requirement is that sensors and actuators that play a role in the evacuation routing process are equipped with a small (embedded) networked device. These devices are commercially available and low-cost. The procedural knowledge that these devices need to be equipped with is very simple, and in practice consist of a small set of signal-handling rules. Finally, the actual network between the devices might be tailored to the cost, quality or any criterion, as long as there is a form of communication. A very appealing line of development might be the development of an all-in-one evacuation support device, that might be composed of various sensors, a signaling device, a network devices and wireless communication capabilities to communicate with other support devices. No matter how an actual physical implementation would look like, the conceptual approach is effective and worthwhile to investigate in other domain.

3.3 Embedding Self-organizing Systems in Control Systems

Both examples in the previous section illustrate the use of self-organizing properties in operational settings. Both could be deployed as fully autonomous support systems (i.e. a fully autonomous chilled water control system, and a fully autonomous evacuation routing system). However, there is a distinct need to integrate such systems into *human*-operated control environments. No matter how much automation will be incorporated in future vessels, humans will always be responsible for the state of the vessels in the end. Especially when it comes to operations that affect the inhabitants of the vessel, human operators should be able to monitor the performance of the control systems and, if necessary, should be able to take over. However, there is a catch here. As the number of components in a platform systems is rising and the overall complexity of managing a platform system is increasing, control systems are becoming harder to design and operate. Even though command centers are equipped with state of the art control systems, trained personnel and well defined control procedures, maintaining proper control during large scale incidents still is a difficult task. Operators nowadays are faced with a multitude of information about the platform system and being hard-pressed for quick system recovery in times of critical incidents. Additionally, future command centers will also need to function with less personnel than currently, due to the ongoing demand for reduced manning. So, there is a definite need for novel approaches to platform system control in general, and the self-organizing properties we discussed earlier may play a part in achieving that ambition. This section describes how self-organizing system may be integrated into conventional control system architectures.

During normal conditions operators can deal very well with failures of today's platform systems. Small failures can be handled accurately and solved without too many problems. The real challenging situations occur during extensive incidents. In these situations operators might be overburdened, occupied with other tasks or unavailable at all, which can lead to costly loss of time before the actual damage control procedures are initiated. Quick response to an emergency is essential. In our vision, future control systems should serve as the first line of defense in case of an emergency: the control system should operate on its own to remedy the situation until the operator is able to take over, even if this would imply that the control system defines a sub-optimal solution. The control system should be able to manage itself in such a way such that at least a minimal set of operational demands are met, such as quick isolation of damaged areas, the immediate initiation of evacuation procedures or promptly turning off equipment to prevent further damage. It thus implements a *reflexive* behavior that is, to some extent, comparable to those found in biological systems (like the withdrawal reflex when we burn ourselves at a hot plate). These reflexes could help to retain the system from harmful conditions and maintain its functional integrity as long as possible. More effective and elaborate recovery plans can be formulated when the operator is back in the loop and the situation has become less critical. This kind of behavior is desirable, since it adds to the overall capabilities of the operator: the control system as a cooperating actor, instead of a mere control panel. Reflexive behavior helps to recover platform system operation as quickly as possible in case of a crisis. Our aforementioned self-organizing control system concept is a concrete architecture capable of providing such reflexive behavior. However, this kind of behavior only pays off if the control system is able to exert the proper control commands on the platform system in case of an incident. In a centralized, monolithic control

system sensors and actuators obtain their control data from a central control location, from which all decisions are made and fed back to the actuators. Of course, this is not a good approach if we aim for a robust arrangement: in case of an extensive emergency communication lines may be cut, leaving the platform system unattended for. To ensure robustness of the control system itself we need to dispose of the centralized design paradigm: we need an arrangement in which the control intelligence is distributed. The components of the control system that enable the reflexive behavior need to be close to the equipment they are controlling. Furthermore, the behavior of these components should be devised in such a way that the system is adaptive. Adaptivity is necessary because in advance it is unknown which parts of the system could be damaged. The architectural concepts we introduced in the two case studies adhere to this notion: agent-based networks that are both geographically and functionally close to the physical systems they control. However, an equally important question is how a such control system should interact with and aid human operators. In most current platform control systems the amount of support from the control system to the operator is. With larger number of actors and devices in control systems it becomes harder to provide the operator with an accurate and complete picture of the current situation without overwhelming the operator. Therefore, control systems need to display some level of intelligent cooperative behavior: the control system should actively participate in the control task. By letting the control system take over elementary tasks (such as putting the platform system in a certain configuration or gathering information), the operator could spend more time on knowledge intensive tasks such as detailed diagnosis and recovery planning. In such a collaborative setup, the purpose of the control system would be to amplify strengths and alleviate weaknesses of the operator by providing more or new resources, such as time or performance, *Chalmers (1998), Terveen (1995).*

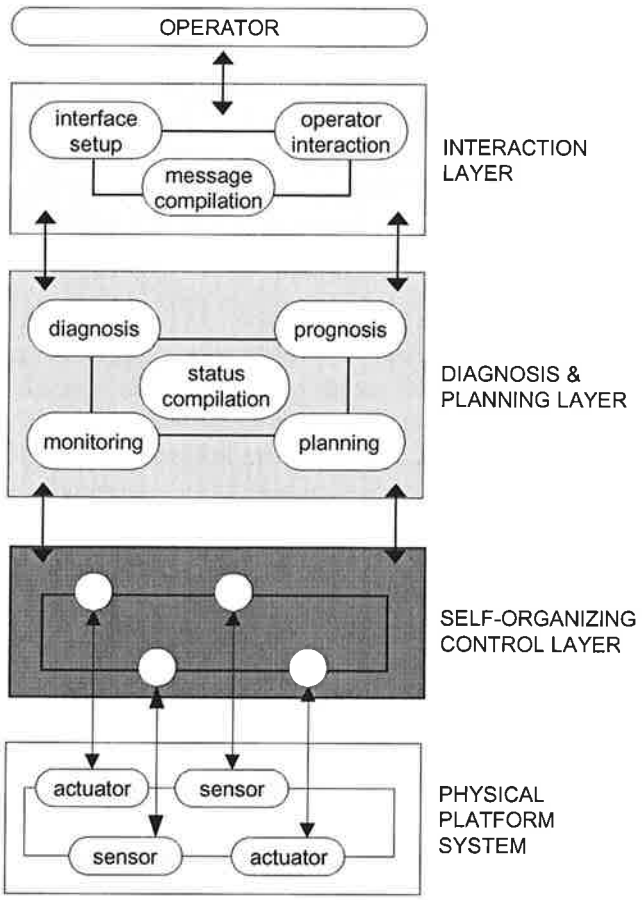

Fig.5: A layered approach to platform system control with an embedded self-organizing control layer

The main objective for our approach is threefold: the architecture must display *robustness* (be as fail-safe as possible), easily *scalable* (easy to extend when the platform system is modified) and must *facilitate* control tasks for the operator. (i.e. prevent alarm flooding, actively support the operator and provide new ways to exert control). Above all, we advocate distributed control. This implies that control tasks are performed by different parts of the control system. To arrive at a functional architecture we have defined behavioral layers for each of the functions that we want to automate and arranged them hierarchically in terms of level of abstraction. Translating data into meaning information takes time. If we're aiming for fast response, we should not be dependent on the completeness and availability of information. The less information needs to be gathered and interpreted the more responsive the action can be. Therefore the most responsive actions (i.e. the reflexive behavior) should be placed at the lowest level of abstraction. Functions that depend on more detailed information such as analysis, diagnosis and scheduling are placed at a higher level.

The proposed architecture exists of three functional layers, Fig.4: a self-organizing control layer, a diagnosis and planning layer and an interaction layer. These layers are connected in a subsumption-like manner, *Brooks (1991),* although other schemes than priority-based selection might fit as well, in which the self-organizing control layer can function on its own and the diagnosis and planning layer can function without the interaction layer. The *diagnosis and planning layer* is responsible for diagnosis, prognosis, condition monitoring and action scheduling. This layer takes on information from the self-organizing control layer and can issue control commands back to the agents in that layer. Just like the self-organizing control layer, this layer is built up using a distributed architecture, i.e. a collection of agents. Conceptually, all functions that one would expect from a decision support system can be placed here. Depending on functional requirements of the control system one may leave out certain aspects without doing harm to the overall concept. The *interaction layer* deals with all operator interaction. It takes care of displaying relevant information in the right form, aids the operator in information gathering, provides feedback if desired, filters irrelevant data and so on. This level is also responsible for proper processing of operator commands. It could, for example, allow the operator to issue a general configuration statement without having to deal with the precise actuator settings required to achieve that particular configuration. The functional meaning of this level would be alike a smart 'personal buddy' for the operator. The information about the system that this assistant would need to have to provide support comes from the subordinate diagnosis and planning layer. Characteristic examples for the intended agent type in this layer can be found in *Maes (1994)* and *Sycara and Zeng (1996).*

In essence, the self-organizing control system becomes a pro-active functional entity within the platform control system. It may be authorized to perform certain functions, under certain circumstances on its own, and also provides indirect access to the physical system, and thus forms a natural extensions to most conventional control systems.

4. Conclusions

The topic of this paper was the application of self-organization principles in maritime environments. We have given a short introduction to some basic concepts and have demonstrated their use in two practical case studies: the control of a chilled water system and evacuation routing. It is important to note that the approach advocated in this paper is a specific type of self-organization, and that there are many more ways in which self-organizational principles can be applied to deal with computational challenges. The most important message this paper aims to convey is the ideology of using self-organizing system models: the use of very simple structures to solve complex problems. From an engineering point of view, self-organization is not just another technology that one can simply adopt. It requires a shift in thinking about systems engineering: in order to obtain more system capabilities, we need to let systems control themselves, and make smart use of the surrounding that the systems are in (e.g. utilize their *embodiment* and *stigmergetic* capabilities). As for opportunities for maritime environments, there are many more than the two examples we have discussed. In general any type of computational problem that deals with routing or autonomous systems coordination can be approached with a self-organizing systems tactic. Especially in the field of multi-agent system design

(be they software agents or embodied agents such as autonomous robots) there are ample opportunities to employ such techniques. And with inherent adaptiveness, robustness and scalability as major benefits, this line of research should appeal a wide audience within the maritime domain.

References

BAK, P.; TANG, C.; WEISENFELD, K. (1988), *Self-organized criticality*, Physical Review A 38, pp.364-374

BAK, P.; CHEN, K. (1991), *Self-organized criticality*, Scientific American, January 1991, pp.46-53

BONABEAU, B.; HENAUX, F.; GUERIN, S.; SNYERS, D.; KUNTZ, P.; THERAULAZ, G. (2000), *Routing in telecommunications networks with 'smart' ant-like agents*, (Eds. Albayrak and Garijo), Intelligent Agents for Telecommunication Applications, Springer, pp.60-71

BONABEAU, E.; SOBKOWSKI, A.; THERAULAZ, G.; DENEUBOURG, J.L. (1997), *Adaptive task allocation inspired by a model of division of labour in social insects*, Lundh et al. (Eds.), Bio-Computing and Emergent Computation, World Scientific, pp.36-45

BROOKS, R.A. (1991), *Intelligence without representation*, Artificial Intelligence 47, pp.139-159

CHALMERS, B.A. (1998), *On the design of computer-based C2 decision support for a modern frigate*, 4th Int. Command and Control Research and Technology Symp., U.S. Department of Defense, C4ISR Cooperative Program, Nasby Park

CICIRELLO, V.A.; SMITH, S.F. (2001), *Insect societies and manufacturing*, IJCAI-01 Workshop on Artificial Intelligence and Manufacturing: New AI Paradigms for Manufacturing, pp.33-38

COLORNI, A.; DORIGO, M.; MANIEZZO, V. (1992), *Distributed optimization by ant colonies*, 1st European Conf. on Artificial Life, pp.134-142

DORIGO, M.; STÜTZLE, T. (2002), *The ant colony optimization metaheuristic: Algorithms, applications and advances*, (Eds. Glover and Kochenberger), Handbook of Metaheuristics, Kluwer Academic Publ.

EYMANN, T.; REINICKE, M.; ARDAIZ, O.; ARTIGAS, P.; FREITAG, F.; NAVARRO, L. (2003), *Self-organizing resource allocation for autonomic networks*, DEXA Workshops 2003, pp. 656-660

GARDNER, M. (1970), *Mathematical games: The fantastic combinations of John Conway's new solitaire game 'Life'*, Scientific American 223, pp.120-123

HEYLIGHEN, F. (2001), *The science of self-organization and adaptivity*, The Encyclopedia of Life Support Systems, EOLSS Publishers

HOLLAND, J.H. (1998), *Emergence: from chaos to order*, Helix Books

JOHNSON, S. (2001), *Emergence: The connected lives of ants, brains, cities, and software,* Scribner

KAUFFMAN, S. (1993), *The origins of order*, Oxford University Press

KUBE, C.R.; ZHANG, H. (1993), *Collective robotics: From social insects to robots*, Adaptive Behaviour 2/2, pp. 189-219

LEWIN, R. (1992), *Complexity: Life at the edge of chaos*, Macmillan

LUCAS, C. (2003), *Self-organizing systems*, in FAQ for Usenet group comp.theory.self-org-sys, (The Complexity & Artificial Life Research Concept for Self-Organizing Systems), http://www.calresco.org/sos/sosfaq.htm

MAES, P. (1994), *Social interface agents: Acquiring competence by learning from users and other agents*, Working Notes of the AAAI Spring Symp.on Software Agents, Stanford, pp. 71-78

MacGREGOR SMITH, J. (2001), *Evacuation networks,* Encyclopedia of Optimization, Vol. 2, C.A. Floudas and P.M. Pardalos (Eds.), Kluwer Academic Publ., pp.36-44

PAGELLO, E.; MONTESELLO, F.; DANGELO, A.; GARELLO F.; FERRARI, C. (1999), *Cooperative behaviors in multi-robot systems through implicit communication,* Robotics and Autonomous Systems 29, pp.65-77

PARUNAK, H.V.D.; BRUECKNER, S. (2001), *Entropy and Self-Organization in Multi-Agent Systems*, 5th Int. Conf. Autonomous Agents (Agents 2001), Montreal, ACM, pp.124-130

PARUNAK, H.V.D.; BRUECKNER, S. (2004), *Engineering Swarming Systems,* (Eds. Bergenti et al.), Methodologies and Software Engineering for Agent Systems, Kluwer

SCHOONDERWOERD, R.; HOLLAND, O.; BRUTEN, J.; ROTHKRANTZ, L. (1997), *Ant-based load balancing in telecommunications networks*, Adaptive Behavior 5/2, pp.169-207

SYCARA, K.; ZENG, D. (1996), *Multi-agent integration of information gathering and decision support*, 12th European Conf. on Artificial Intelligence (ECAI '96), Budapest, pp.549-556

TERVEEN, L. (1995), *An overview of human-computer collaboration*, Knowledge-Based Systems 8/2-3, pp.67-81

Computer Integrated Planning and Resource Management in Shipbuilding

Christian Massow, Logimatic Software, Bremen/Germany, chm@logimatic.dk
Ivan Siksne-Pedersen, Logimatic Software, Aalborg/Denmark, isp@logimatic.dk

Abstract

High complexity of product and processes, parallel processing, high customer influence are the most important characteristics of order processing in today's shipbuilding industry. This requires special, dedicated methods and tools for planning and resource management. Integration of these "best-of-class" solutions in CIM environments is the challenge to work effectively on basis of common information in all areas of the shipyard. Organization and software systems to support an integrated planning and control approach are discussed. An example illustrates a practical integrated IT solution for shipbuilding taking into account aspects of integrated planning and resource management.

1 Introduction

Due to global competition the shipbuilding industry is under pressure to improve its manufacturing efficiency. The economical situation of the shipbuilding industry is influenced by an increase in demand to improve integrated information and control concepts. The efficiency of production/project management and coordinating activities within complex production environments is expected to increase by integrating operations on various levels of planning and control. These levels include both intra-organizational operations as found in autonomous, de-centralized areas as well as operations to be coordinated between multi-site production facilities such as between manufacturer, supplier, and sub-supplier.

In this situation adoption of well-known production management concepts is not useful since the shipbuilding process, as an good example of one-of-a-kind production, shows significant differences compared to repetitive production.

As these differences, concerning customer order processing and the production process, necessitate specific requirements for a pursued concept of production management, an individual approach for the shipbuilding process is required.

2 Planning and Control in Shipbuilding

The special features of customer order processing in shipbuilding lie in the individuality of the product and the required production processes together with their corresponding dependencies. The main characteristics influencing the required production planning and control concept are:

- Due to the short lead times (relative to product and process complexity), the large-scale orders/projects require parallel production activities. This includes the simultaneity of design, procurement, operations planning, manufacturing and assembly.
- Offer planning is extremely important since every offer involves such a large outlay of financial and production resources that the existence of the organization is at risk. Moreover, the relationship between orders acquired and offers to be prepared has deteriorated over the past years and the offers must be worked up under increasing time pressure.
- Customer influence does not end with the signing the contract. In fact, the customer can permanently control product specification as far as actual product delivery. Process specification in turn is also affected of course.
- Product complexity and decreasing manufacturing penetration in terms of lean production leads to inter-organizational production. This inter-organizational production necessitates additional efforts for production planning and control.

3rd International Conference on

Computer and IT Applications in the Maritime Industries

COMPIT'04

Siguënza, 9-12 May 2004

Volker Bertram, Manuel Armada (Ed.)

Sponsored by

This work relates to Department of the Navy Grant N00014-04-1-1055 issued by the Office of Naval Research Global. The United States Government has a royalty-free license throughout the world in all copyrightable material contained herein.

Index

Volker Bertram 5
Towards Intelligent Simulation-based Ship Design

Marco Ferrando, Andrea Lommi, Antonio Traverso 17
Use of Genetic Algorithms in Propeller Foil Design

Jörn Hinnenthal[1], Stefan Harries 27
A Systematic Study on Posing and Solving the Problem of Pareto Optimal Ship Routing

Otto Millbourn 36
Concurrent Engineering through the Supply Chain from Ship Owner to Supplier

Hans Wuttke, Andreas Schwill, Frank Domeyer 42
An E-Commerce Platform for Engineering Services in Shipbuilding

Tobias Haack, Stefan Krüger 52
Propulsion Plant Models for Nautical Manoeuvre Simulations

Lawrence Henesey[2], Paul Davidsson, Jan A. Persson 61
Using Simulation in Evaluating Berth Allocation at a Container Terminal

Eberhard Blümel, Leonid Novitski 73
Introduction to BALTPORTS-IT: Applications of Simulation and IT-Solutions in the Baltic Port Areas

Peter Weiss[1], Yann Le Page, Frédéric Schom, Alain Fidani 81
Robot Applications in the Field of Shipbuilding

Vedeesh Sahajpal 86
An intelligent GA based AIS FATDMA scheduler

Jonathan M. Ross 98
A Practical Approach for Ship Construction Cost Estimating

Robert Bronsart, Steffen Gau, Diane Luckau, Wolfgang Sucharowski 111
Communication in ship design networks

Torsten Fischer, Hermann Gehring 122
A Multi-Agent based Approach for Solving the Vehicle Transshipment Problem

Meelis Mäesalu, Jerzy Matusiak 131
Visualisation of Ship Motions in Waves and during Grounding for Simulators

Peter Weiss[1], Fivos Andritsos, Frédéric Schom, Alain Fidani 140
Innovative Robotic Solutions for the Survey and Certification of Ships and Mobile Offshore Units

Yuichi Sasaki, Hiroyuki Yamato, Shoichi Enomoto 148
Research on Industrial Engineering System by using Wearable PC

[1] Partially supported by SENER
[2] Partially sponsored by Tribon Solutions