Towards Dependable Swarms and a New Discipline of Swarm Engineering

Alan F.T. Winfield, Christopher J. Harper, and Julien Nembrini

Intelligent Autonomous Systems Laboratory,
UWE Bristol, Coldharbour Lane, Bristol BS16 1QY, UK
Alan.Winfield@uwe.ac.uk
http://www.ias.uwe.ac.uk/

Abstract. This review paper sets out to explore the question of how future complex engineered systems based upon the swarm intelligence paradigm could be assured for dependability. The paper introduces the new concept of 'swarm engineering': a fusion of dependable systems engineering and swarm intelligence. The paper reviews the disciplines and processes conventionally employed to assure the dependability of conventional complex (and safety critical) systems in the light of swarm intelligence research and in so doing tries to map processes of analysis, design and test for safety-critical systems against relevant research in swarm intelligence. A case study of a swarm robotic system is used to illustrate this mapping. The paper concludes that while some of the tools needed to assure a swarm for dependability exist, many do not, and hence much work needs to be done before dependable swarms become a reality.

1 Vision

From an engineering standpoint the design of complex distributed systems based upon swarm intelligence is compellingly attractive but problematical. A distinguishing characteristic of distributed systems based upon swarm intelligence is that they have no hierarchical command and control structure, and hence no common mode failure point or vulnerability. Typically, individual agents make decisions autonomously, based upon local sensing and communications [5, 6]. Systems with these characteristics could, potentially, exhibit very high levels of robustness, in the sense of tolerance to failure of individual agents; much higher levels of robustness than in complex distributed systems based on traditional design approaches. However, that robustness comes at a price. Complex systems with swarm intelligence might be very difficult to control or mediate if they started to exhibit unexpected behaviours. Such systems would therefore need to be designed and validated for a high level of assurance that they exhibit intended behaviours and *equally importantly* do not exhibit unintended behaviours. It seems reasonable to assert that future engineered systems based on the swarm intelligence paradigm would need to be subject to processes of design, analysis and test no less demanding that those we expect for current complex systems.

E. Şahin and W.M. Spears (Eds.): Swarm Robotics WS 2004, LNCS 3342, pp. 126–142, 2005.

Some might argue that a 'dependable swarm' is an oxymoron; that the swarm intelligence paradigm is intrinsically unsuitable for application in engineered systems that require a high level of integrity. The idea that overall desired swarm behaviours are not explicitly coded anywhere in the system, but are instead an emergent consequence of the interaction of individual agents with each other and their environment, might appear to be especially problematical from a dependability perspective. This paper suggests that this is not so: that systems which employ emergence should, in principle, be no more difficult to validate than conventional complex systems and, indeed, that some characteristics of swarm intelligence are highly desirable from a dependability perspective.

The aim of this paper is to explore the question of how future engineered systems based on the swarm intelligence paradigm might be designed, analysed and tested for dependability. The paper attempts to do this by the juxtaposition of two hitherto disconnected disciplines: dependable systems engineering and the design of multi-agent systems based on the swarm intelligence paradigm (which we shall term 'swarm engineering'). This is a big question, a complete answer to which is well beyond the scope of this paper. The paper instead tries to set out the important questions for the ongoing study of dependable swarms.

In order to illustrate the questions raised by this paper an example of a robotic swarm is presented as a case study. The case study is incomplete, since the tools and disciplines needed to fully validate the system in question do not exist: that is of course the point of this paper. The case study does, however, help us to think about the rather abstract issues of dependable systems engineering with reference to a robotic swarm that could see real-world application within the near future. This paper proceeds as follows. Section 2 introduces the case study that will be used throughout the rest of the paper. Section 3 is a review of current best practice in the field of dependable systems engineering. While outlining and referencing the processes and methodologies of analysis, design and test, this section will reflect on what these might mean in practice, for swarm engineering, with reference to the case study. Section 4 then concludes with a discussion and outlook, setting out a roadmap of the work that needs to be done before real-world swarm engineering can become a reality.

2 Case Study: Swarm Containment

As a case study let us consider a swarm robotics approach to physical containment or encapsulation, as illustrated in figure 1.

Potential applications for such an approach might include a swarm of marine robots that find and then contain oil pollution or *in-vivo* nano-bots that seek and isolate harmful cells in the blood stream (a kind of artificial phagocyte). The latter application is not so far-fetched when one considers the rate of progress in the engineering of genetic circuits, see Yokobayashi et al [27].

The emergent encapsulation behaviour of figure 1 is one of a number of emergent properties of a class of algorithms that we have developed, which make use of local wireless connectivity information alone to achieve swarm aggregation; see Nembrini et al. [18]. Wireless connectivity (what Støy termed *situated com-*

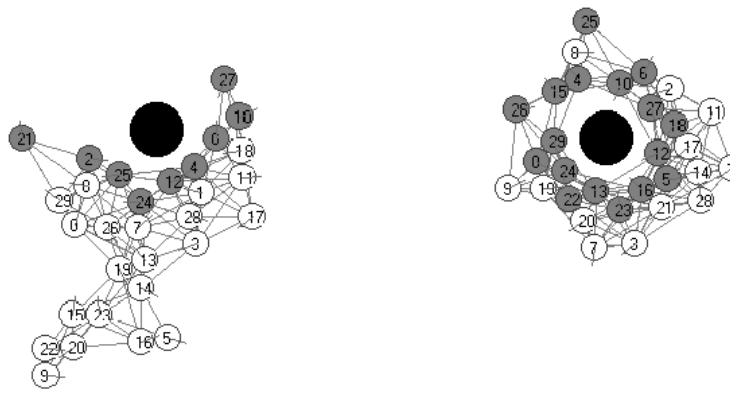

Fig. 1. Emergent encapsulation; (left) encapsulation in progress and (right) encapsulation complete.

munication [24]) is linked to robot motion so that robots within the swarm are wirelessly 'glued' together. This approach has several advantages: firstly the robots need neither absolute or relative positional information; secondly the swarm is able to maintain its coherence (i.e. stay together) even in unbounded space, and thirdly, the connectivity needed for and generated by the algorithm means that the swarm naturally forms an ad-hoc communications network. Such a network would be a requirement in many swarm robotics applications. The algorithm requires that connectivity information is transmitted only a single hop. Each robot broadcasts its ID and the IDs of its immediate neighbours only, and since the maximum number of neighbours a real robot can have is physically constrained and the same for a swarm of 100 or 10,000 robots, the algorithm scales linearly for increasing swarm size. The algorithm thus meets the criteria for swarm robotics, articulated by Sahin [21] and Beni [3]. We have a highly *robust* and *scalable* swarm of homogeneous and relatively incapable robots with only local sensing and communication capabilities, in which the required swarm behaviours are truly emergent. Furthermore we observe *flexibility* to its environment in that our wireless connected swarm demonstrates emergent taxis towards a beacon (which, in this case, is the object to be contained), emergent obstacle avoidance and emergent beacon encapsulation.

Our algorithms for coherent swarming of wireless networked mobile robots have been tested extensively in simulation and, rather less extensively, using a fleet of physical laboratory robots. A group of these robots ('Linuxbots') are shown in figure 2. The real robot implementation does not, however, constitute a real-world application. It is instead an 'embodied' simulation, whose main purpose is to verify that algorithms tested in computer simulation will transfer to the real world of non-ideal and noisy sensors and actuators.

3 Dependable Swarm Engineering

Current best practice in assuring the dependability of complex systems requires that a set of processes and disciplines are transparently applied during system

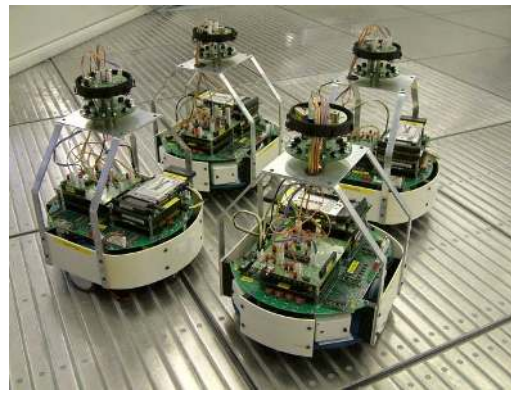

Fig. 2. The Linuxbots, used for embodied simulations.

analysis, *design* and *test*, see Anderson et al [1]. This paper now considers the approaches that would typically need to be applied to safety-critical systems in the context of swarm engineering, under these three headings. Note that best practice requires that the processes of analysis, design and test are applied concurrently and iteratively, so the ordering of the following sections should not be taken to imply sequence.

3.1 Analysis

From a dependability perspective, analysis is concerned with trying to establish two properties of a system: 'liveness' and 'safety'. Liveness is defined as the property of exhibiting desirable behaviours (doing the right thing) and safety is defined as the property of not exhibiting undesirable behaviours (not doing the wrong thing). While these properties are clearly somewhat complementary proof of one does not imply proof of the other, by inversion. A system that is provably safe could, for example, do the wrong thing safely. Although it may appear counter-intuitive, the methods needed to verify these two properties are not the same.

Verification of Liveness. Verification of 'liveness' requires that we formally prove that a system exhibits desirable behaviours. Conventionally this requires analytical or mathematical modelling. In the safety systems community the use of testing alone to prove liveness is now deprecated on the grounds that systems are becoming too complex to allow anything like acceptably complete test coverage, or even to allow complete test specifications to be written. Simulation is similarly regarded as unacceptable as an analysis tool (an interesting observation given the widespread use of simulation within swarm intelligence research[1]). Simulation is nevertheless accepted as a useful tool in prototyping,

[1] For a valuable discussion of the role of simulation in embodied systems research see Ziemke [29].

to for instance refine the system specification and to understand the design or parameter space.

Complete verification of the liveness of a swarm system thus requires mathematical modelling at two levels: the individual agent, and the swarm as a whole.

Let Us First Consider the Individual Agent. Often, single artificial agents within swarms are designed using the behaviour-based control paradigm [7]. Behaviour-based control is appropriate given that such agents are typically reactive finite-state machines with relatively few states. We have developed an approach, based on a second order extension of Lyapunov stability theorems, proving both marginal and asymptotic stability [11]. The significance of second order stability is that position control in mobile agents can generally only be achieved through actuators that generate forces which govern acceleration; the second derivative of position. Of particular significance is that these new stability theorems provide an explicit mathematical representation of subsumption. Based on this observation Harper has developed a design methodology called 'Direct Lyapunov Design' which leads from analysis directly to a colony-style behaviour based controller which is provably stable (in the sense of Lyapunov), and exhibits the liveness property [12]. In a fixed-priority behaviour-based architecture such as the colony-style subsumption architecture [9], the transfer functions of behaviour modules must be *partial functions*, i.e. which do not generate outputs continuously, in order that lower priority behaviour modules will have a chance to drive the system. Direct Lyapunov Design allows the construction of behaviour modules as partial functions and hence their integration into a colony-style subsumption controller. This approach thus advantageously encompasses both analysis and design.

Case Study: Figure 3 shows the colony-style subsumption controller for a single robot in the coherent swarm described in section 2. For simplicity only the bottom three layers are shown; the beacon-taxis layer is omitted. Notice that the local neighbourhood connectivity information can be treated as, in effect, sensory input to the coherence layer. In fact, the coherence layer makes use of memory to store neighbourhood connectivity, but in modelling the controller as a subsumption architecture we can treat the memory as part of the connectiv-

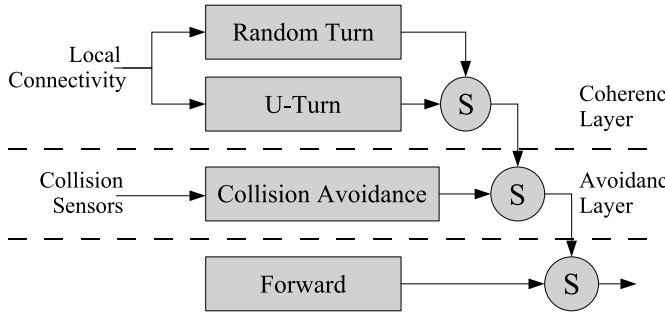

Fig. 3. Case study: single robot control architecture.

ity 'sensor'. The two behaviours in the coherence layer are 'U-turn', executed when the connectivity-sensor estimates that the robot is leaving the swarm; and 'Random turn', executed when the connectivity-sensor estimates that the robot has regained the swarm (for a description of how these estimates are made refer to [18]).

As a demonstration of the application of the second order stability theorems consider the analysis of the avoidance layer module. If we describe the state-space vector for the avoidance layer as $\underline{x}_A(t)$, and assume the existence of a candidate Lyapunov function[2] $V_A(\underline{x})$, then the value of that Lyapunov function along the state trajectories of the avoidance behaviour can be defined as function $W_A(t)$ where

$$W_A(t) \equiv V(\underline{x}_A(t)) \tag{1}$$

The principle of the method is based on the observation that the first order asymptotic stability theorem subsumes the second order theorem whenever the system motion is stable in the first order sense, i.e. whenever the motion is naturally convergent on the desired goal and $\dot{W}_A(t) < 0$. The second order asymptotic stability theorem can be used to design stable behaviour even if $\dot{W}_A(t) \geq 0$ within limits, as long as the second derivative $\ddot{W}_A(t)$ is negative and the motion is decelerating, i.e.

$$0 \leq \dot{W}_A(t) < \dot{W}_{max} \wedge \ddot{W}_A(t) < \ddot{W}_{max} < 0 \tag{2}$$

In order to achieve stable collision avoidance behaviour the transfer function of the collision avoidance behaviour module needs only to be defined for states where $\dot{W}_A(t) \geq 0$, generating outputs (actions) which ensure that $\ddot{W}_A(t) < 0$ and therefore it is a partial function over the state space of the collision avoidance behavioural domain. Since it is a partial function it can be included within a fixed-priority subsumption architecture. The same argument would apply to the coherence layer and the value of the Lyapunov function along the state trajectory for the coherence behaviour, $W_C(\underline{x})$. Thus, we are able to move toward verification of the liveness property for a single robot within our case study.

Now Consider the Mathematical Modelling of the Whole Swarm. There has been relatively little work in this direction, but one very promising approach is the probabilistic model developed by Martinoli et al. [19]. In this approach the interactions of agents with each other and their environment are modelled as a series of stochastic events, with probabilities determined by simple geometrical analysis. By modelling several series together, one for each agent, the overall behaviour of the swarm can be studied. The approach of Martinoli et al may be thought of as bottom up (or microscopic as they describe it). A top down (or macroscopic) approach has been developed by Lerman and Galystan [14]. Like Martinoli, Lerman and Galystan regard the behaviour of each agent as inherently probabilistic and Markovian, because their next state is a function only of

[2] Which could be as straightforward as the Euclidian distance between $\underline{x}(t)$ and the goal state $\underline{x}_g(t)$.

their current state. However, they develop an overall model of the system using the stochastic Master Equation (from stochastic dynamical systems), then derive rate (differential) equations from it, which describe how the average macroscopic system properties change over time. A review of modelling and analysis methods for swarm robotic systems is given in Lerman et al. [15].

Case Study: Analysis suggests that mathematical modelling of our swarm containment case will not yield to the method proposed by Martinoli, et al [19]. We are able to model the single robot controller as a state transition diagram (see figure 4), however, because our swarm operates in an unbounded space then geometrical analysis cannot be used to develop expressions for the state transition probabilities. In particular transitions between the forward state and the U-turn or Random-turn states in the coherence behaviour depend on local network topology. Similarly, we are unable to use the macroscopic approach of Lerman et al [14] because the individual agents in our case cannot be modelled as simple Markovian processes; they have memory and their next state may depend on the recent history of the local network topology.

Verification of Safety. To verify 'safety' we need to prove that a system does not exhibit undesirable behaviours. In order to attempt such a proof first requires that we identify and articulate all possible undesirable behaviours. This is called 'hazard analysis' and is problematical with conventional complex systems; and there is no reason to suppose that identifying the hazards in swarm engineered systems will be any different. Hazards analysis is problematical because there are no formal methods for identifying hazards. It simply has to be done by inspection (typically by 'extreme brainstorming' to try and list all possible hazards no matter how seemingly implausible or improbable).

Given a reasonably well understood operational environment there are two reasons for undesirable behaviours: random errors, or systematic (design) errors. Random errors are those due to hardware or component faults, and these are typically analysed using techniques such as Failure Mode and Effects Analysis

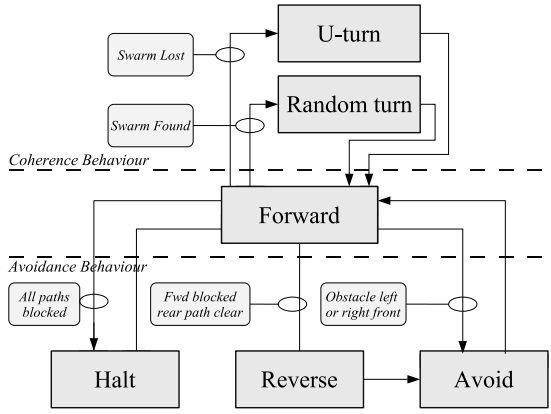

Fig. 4. Case study: single robot state transition diagram.

(FMEA). The likelihood that random errors cause undesirable behaviours can be reduced, in the first instance, by employing high reliability components. But systems that require high dependability will typically also need to be fault tolerant, through redundancy for example. This is an important point since swarm engineered systems should, in this respect, offer significant advantages over conventional complex systems. Two characteristics of swarms work in our favour here. Firstly, simple agents with relatively few rules lend themselves to FMEA, and their simplicity facilitates design for reliability. Secondly, swarms consist of multiple agents and hence, by definition, exhibit high levels of redundancy and tolerance to failure of individual agents. Indeed, swarms may go far beyond conventional notions of fault tolerance by exhibiting tolerance to individuals who actively thwart the overall desired swarm behaviour.

Systematic errors are those aspects of the design that could allow the system to exhibit undesirable behaviours. For swarm engineered systems analysis of systematic errors clearly needs to take place at two levels: in the individual agent and for the swarm as a whole. Analysis of systematic errors in the individual agent should be helped by the relative simplicity of the agents, but is not trivial. In general terms we would need to prove that an agent's state-space trajectory is always 'away from' the hazard states. Following the discussion of section 3.1 we conjecture that the 2nd order Lyapunov approach could be extended to cover the analysis of hazard states as well as goal states, thus offering the possibility of verifying liveness and safety with a single analysis, see Appendix A. Analysis of systematic errors for the swarm as a whole is much more problematical, particularly if the desired behaviours are emergent. Proof of safety for the overall swarm would appear to require that we prove that there are no undesired emergent behaviours. How to prove this to an acceptable level of confidence is by no means clear.

Case Study: A valuable measure of the 'coherence' of our swarm is network connectivity. Within the coherence layer of our single robot controller comparison of local network connectivity against a threshold determines the estimate of 'swarm lost' and hence triggers the U-turn behaviour. Adjusting this threshold value for the whole swarm controls the network connectivity, and hence area coverage; a low value of threshold generates a low density swarm with relatively few wireless connections between individual robots, whereas a high threshold value generates a dense and highly connected swarm. We have developed, from graph theory, upper and lower bounds on the area coverage of the swarm, for given threshold values and swarm sizes. While these bounds are rather loose, they nevertheless provide valuable confidence that the swarm will not exceed a given area coverage. Of course, the swarm exceeding a given area is only one possible 'hazard', so our upper bounds analysis provides proof of swarm safety for just this one identified hazard.

3.2 Design

The design of systems based on the swarm intelligence paradigm is challenging, not least because there are no principled design approaches for determining the

behaviours required of the individual agents in order to give the desired emergent overall swarm behaviour. Indeed, some would argue that a principled approach to the design of emergence is impossible. This paper is however concerned with dependability, and there is no reason to suppose that emergent behaviours cannot form part of a dependable system.

Most complex systems are designed top-down from an overall functional design specification (FDS), by functional decomposition: breaking down the overall system into smaller and smaller components, then defining each of those components and the interfaces between them. What differentiates design for dependable, or safety critical, systems is that it will typically use a structured design methodology to provide a framework for capturing and documenting the design as it progresses, top down. The Yourdon structured design methodology, for instance, is based upon the dataflow paradigm. It starts at the top level by describing the overall system and its interfaces with its operational environment as a 'context diagram': this is level 0. The context diagram is then decomposed into level 1 'processes' and the dataflows between them, expressed in a data flow diagram (DFD). Each process in level 1 is then further decomposed into lower level DFDs, and so on, see Yourdon [28]. The structured design may well be applied within the discipline of a document driven approach [13], together with code inspection [10].

If we consider the applicability, and utility, of the Yourdon structured design methodology to swarm engineered systems it is clear that, at the top level, we can express the single swarm and its interfaces with the environment as a context diagram (level 0). Equally well, we could describe the internal processes of an individual agent with a data flow diagram (level 2). What is interesting, however, is how we might express the intermediate level 1 as a DFD. If we assume that single agents are (a) mobile, and (b) able to sense only their immediate neighbours [18, 25], then the level 1 DFD will reflect the instantaneous topography of the swarm. After the mobile agents have moved, the DFD must change to reflect the new swarm topography. This interestingly suggests an extension of the DFD which we could term the 'dynamic data flow diagram'.

Case Study: As discussed above we can express the design of our case study swarm robotic system graphically, as a hierarchy of data flow diagrams. Figure 5 shows the level 1 DFD but, in a departure from standard DFD notation, the data flows between level 1 processes - which happen to be robots - will change dynamically as the robots move. The DFD in figure 5 is thus a snapshot of the relationship between processes, rather than a static map. However, since every level 1 process (robot) and every dataflow between level 1 processes is identical then the DFD in figure 5 is simpler than it appears. The value of this approach is that we can make use of the full structured formalism of Yourdon to capture the design at both swarm and single robot level.

Figure 6 shows the DFD for a single level 1 process (robot), and its decomposition into level 2 processes. The 'behaviour-based control process' shown in figure 6 is described as a subsumption architecture in figure 3, and a state transition diagram in figure 4. When we add specifications for interfaces between

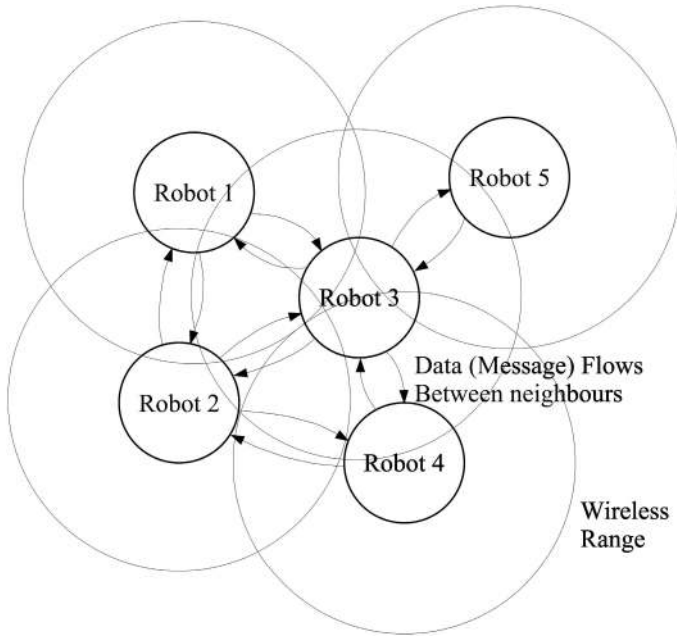

Fig. 5. Case study: swarm dynamic data flow diagram.

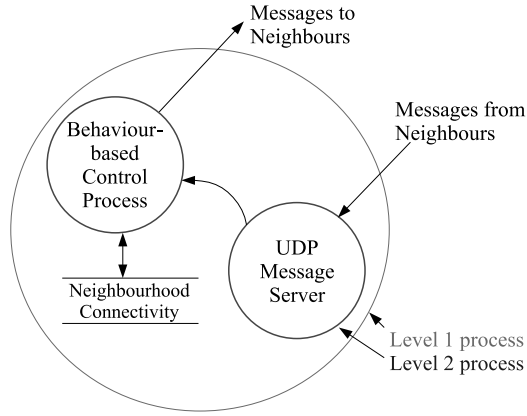

Fig. 6. Case study: single robot data flow diagram.

processes (dataflows) and data structures then we have a complete description of the design specification for the swarm and its robots. The Direct Lyapunov Design methodology [12] introduced in section 3.1 provides a design procedure for formally deriving implementations of individual robot behaviours, as 'motor schema', from the 2nd order Lyapunov stability analysis. The advantage of motor schema [2] is that they are simple piecewise mapping functions relating sensor inputs to actuator outputs which could be realised as gate arrays for very reliable controller hardware.

3.3 Test

Within the safety critical systems community there is general agreement that testing, whilst essential, can only provide a limited measure of confidence in the liveness and safety properties of a system [8]. There are two problems. Firstly, to write a complete test specification for a complex system is very difficult, and secondly to achieve 100% test coverage (which means exercising every possible execution path through control code or state machines under controlled conditions), whilst not technically impossible, is infeasibly time consuming for even moderately complex systems. Thus even the most safety critical systems in use today, such as aircraft flight management systems, will have been put through demanding but ultimately incomplete testing [16]. This is the reason that testing needs to go hand in hand with mathematical modelling, as discussed in 3.1 above.

Typically, a test regime for safety critical systems is split into two parts: system level functional testing and component level testing. System level testing is primarily concerned with liveness, and treats the overall system as a black box, testing only for correct behaviour of the system as a complete entity against a system test specification. Component level testing breaks the system into its sub-systems and tests each one individually. Thus component level testing is the equivalent of system level white box testing.

At component level, sub-systems need to be tested functionally. This normally requires that test harnesses are created to enable components to be tested in isolation from the rest of the system. A test harness will set up input conditions for a component that might be extremely difficult to create by treating the system as an integrated whole. Test coverage can be measured directly in a process called dynamic analysis, which 'instruments' code such that each time it is executed a tally is kept of the number of times every possible execution path has been exercised. Dynamic analysis is an iterative (and cumulative) process in which ever more ingenious new tests are devised (typically by inspection of the code), in order to exercise those parts of the code revealed to have been not executed by the testing so far. The process continues until the target level of test coverage has been achieved. Needless to say dynamic analysis is a difficult and time consuming process. For completeness static analysis should also be mentioned since it often goes hand in hand with dynamic analysis. Static analysis measures code without actually executing it against coding standards including, typically, the McCabe complexity measure to assess the 'spaghetti-ness' of code [20].

If we now consider swarm engineered systems in the light of the discussion above, it is clear that system level testing needs to apply to the swarm as a whole, operating in its intended environment, and component level testing applies, in effect, to an individual agent. The fact that individual agents are often identical in swarm systems, and relatively simple in functional terms, suggests that component level testing should not be intractable. This view is, however, probably illusory, since the 'environment' for a single agent is the sum total of the other (presumably) neighbouring agents and the environment. Complete

testing of a single agent would require that every possible configuration of neighbours and environment is specified, and repeatable tests devised (the neighbours plus environment becomes in effect the test harness). There has been little work in mobile robotics to quantitatively assess the effect of its environment on an individual robot, but see Schöner et al. [22]; Smithers [23]. The recent paper of Nehmzow and Walker [17] suggests methods based on dynamical systems theory, time series analysis and deterministic chaos theory.

The question of how to write a swarm test specification (STS) for the swarm as a whole might appear to be problematical given that the internal structure of the swarm is typically highly dynamic and chaotic. However, if we discipline ourselves to treating the swarm as a single entity then it should be possible to develop tests for the desired swarm behaviours. These will almost certainly be statistical, measuring for instance the frequency with which a given behaviour reaches a quantitative threshold condition of achievement within a given time frame, over repeated test runs. Thus, developing an STS for a swarm engineered system is likely to require careful attention to defining criteria for the achievement of swarm behaviours, including metrics for swarm properties such as mean swarm velocity, or mean area coverage.

Case Study: Providing the means to repeatably test a real robotic swarm could well, depending upon the size and form of the robots, present a significant engineering challenge. This requires, in the first instance, an instrumented test arena in which a representative operational environment can be created so that the performance of the swarm in achieving its desired behaviours can be observed and measured. This is itself not straightforward. Of course ultimately the swarm would also need to be tested in its real operational environment and that could be an even greater challenge, so let us confine ourselves here to thinking about the controlled test environment. Figure 7 shows two successive frames from a test run of our case study swarm. It must be stressed that these robots are not the real robots of a real-world application of our case study; the setup shown here is an embodied simulation aimed at providing proof-of-concept confirmation of the basic algorithms. Nevertheless, it will serve to illustrate the tools and techniques that will be required to test the real swarm.

Figure 7 shows a test of our embodied simulation in progress. Here we are trying to experimentally verify that the swarm maintains coherence, i.e. stays together. Note that the seven robots in figure 7 are grouped together by virtue of their wireless connectivity, not the physical bounds of the experimental arena; it follows that in our case the arena needs to be large enough with respect to the coherent swarm to provide it with an effectively unbounded space. The tests of figure 7 are concerned with, firstly, testing when, how and with what probability robots become detached from the swarm and, secondly, measuring the swarm area coverage. Area coverage is indicated by the bounded polygon of figure 7. The swarm test arena needs to provide the means to (a) motion capture test runs, (b) track and label individual robots and (c) process the captured test sequences to identify lost robot events and measure area coverage. The fact that our robots are equipped with wireless LAN [26] is a distinct advantage here, since it provides

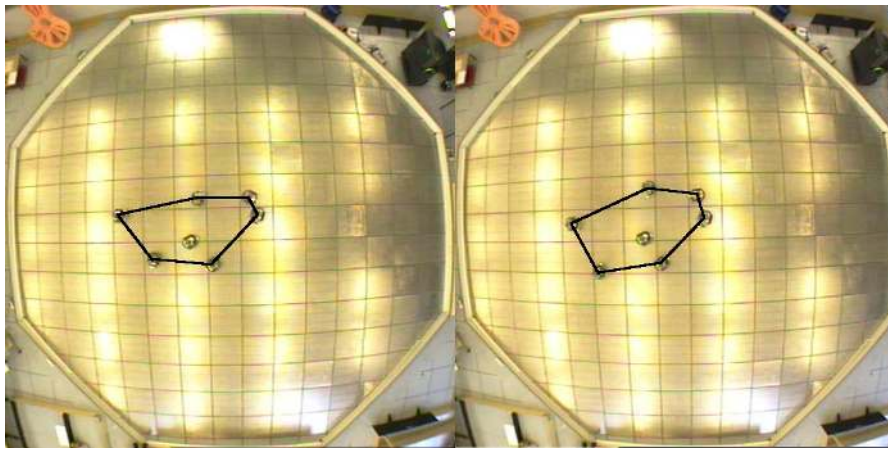

Fig. 7. Case study: two successive frames from a system level test run (captured by the overhead camera in the laboratory test arena).

us with, firstly, the ability to be able to command the robots to a given starting position to initialise each test run and, secondly, the means to obtain continuous telemetry on each robot's internal state, connectivity, and odometry. Recording and time synchronising this data against the motion capture is important since it provides us with the information to be able to conduct deep analysis of the progress of test runs. In fact, a test script which automatically initialises each test run then, at the end of the run, halts the robots and re-initialises them for the next run, allows us to automate the whole of the swarm test sequence, from the STS to plots of swarm performance metrics.

If we now consider the problem of conducting component level tests, i.e. tests on a single robot, we can see that the experimental test environment described here provides us with the means to verify the correct operation of a single robot under a very wide range of 'environmental' conditions (recall that the test environment for a single robot is the sum total of its neighbouring robots plus the external (to the swarm) environment). By collecting data on internal state, connectivity and odometry for every robot, we can track the progress of a single robot through the swarm and - for a wide range of local conditions (proximity and connectivity) - confirm that the control action actually taken by the robot is the action that would be expected for those particular conditions. The dynamicity of the swarm provides us naturally with a very wide range of 'test' conditions for an individual robot, and by running a simple simulation of a single robot controller we can automate the process of verifying actual against expected control actions. Thus, in a sense, the single Robot Test Specification (RTS) does not need to be written (in that every possible test condition does not need to be written down), nor does it need to be manually executed. The system level test provides both the tests, and test environment, for the single robot.

4 Discussion and Outlook

This paper has proposed a framework for a new discipline of 'Swarm Engineering'. The paper has attempted a juxtaposition of dependable systems engineering with swarm intelligence and in so doing has tried to map processes of analysis, design and test for safety-critical systems against relevant work in swarm intelligence research. Perhaps not surprisingly, there is not a great deal of overlap between the two fields. To the authors' knowledge there has not been, to date, a single real-world application of swarm engineering with real physical agents. Thus no-one has yet had to face the challenge of assuring the dependability of such a system.

In respect of *analysis*, this paper has shown that promising mathematical modelling approaches are emerging for establishing the *liveness* property, for both the overall swarm and its constituent robots. These approaches are at present limited; for the overall swarm, to swarms in which individual robots can be treated as stochastic Markov processes; and for individual robots in which the controller can be modelled as a colony-style subsumption architecture. The more serious weakness, from a dependability perspective, is that no tools exist for establishing the *safety* property, that is to determine that a robotic swarm cannot exhibit undesirable behaviours. How to do this is by no means clear, although this paper has suggested two possible approaches: an extension of the Lyapunov stability approach for the individual robot, and a 'bounding' approach for the overall swarm.

From a *design* perspective, this paper has shown that the Yourdon structured design methodology might be usefully employed to describe the design of a robotic swarm; the approach has the merit of consistency when moving from the description of the overall swarm to its constituent robots. However, this approach is largely a description tool. Ideally, we require a formal, provable approach to the design of individuals within the swarm, and to the design of overall swarm behaviours. This paper has indicated one possible approach to the former with the technique we term Direct Lyapunov Design. Overall swarm design is problematical because there are at present no principled approaches to the design of emergent behaviours: finding the set of 'atomic' behaviours for the individuals in the swarm that will result in the overall desired emergent behaviour is at present more a process of discovery than design. This paper has, however, argued that this is not necessarily a problem for dependability providing that the emergent swarm behaviours can be assured for liveness and safety.

Finally, in respect of *test*, this paper has highlighted the need to establish robust measures for determining when and how desired swarm behaviours have been achieved, then define (statistical) tests for these measures. The paper has argued that testing, while certainly challenging, is feasible if an appropriate test environment can be created. A surprising conclusion of this paper is that an instrumented test environment for the whole swarm also provides an environment for rigorously testing the swarm at component (i.e. robot) level - for free.

To summarise, from a dependability perspective, future work is needed:

- to extend methods for the mathematical modelling of swarm robotic systems;
- to extend and strengthen formal approaches to provably stable single robot control;
- to start work on 'safety' analysis at both swarm and individual robot levels;
- to develop, if possible, a principled approach to the design of emergence;
- to extend the Direct Lyapunov Design approach to a wider class of behaviour-based controllers, and
- to develop methodologies and practices for the testing of swarm engineered systems.

It is clear that a great deal of work needs to be done before dependable robotic swarms can become an engineering reality.

Acknowledgments

The authors gratefully acknowledge discussions with Chris Melhuish during early preparation of this paper. We are also grateful to the referees for helpful and constructive criticism.

Appendix A

The work of Harper [12], shows that if we have a behaviour-based controller in which behaviours are implemented as motor schema, then we can prove that the Euclidian distance $\|\underline{x} - \underline{x}_g\|$ is a Lyapunov function for that schema and therefore that its behaviour is stable with respect to the goal states $\underline{x}_g$. This represents a formal proof of 'liveness'.

Conjecture: that there is a Lyapunov function $V(\underline{x})$ defined as the ratio of the Euclidian distance of the goal states $\underline{x}_g$ and the hazard states $\underline{x}_h$,

$$V(\underline{x}) = \frac{\|\underline{x} - \underline{x}_g\|}{\|\underline{x} - \underline{x}_h\|} \tag{3}$$

and if the trajectory of $V(\underline{x})$ is negative, i.e. $\dot{V}(\underline{x}) < 0$ then the agent will both seeks its goals and avoid its hazards at the same time. In other words the liveness and safety properties are stable over state space.

References

1. Anderson T, Avizienis A and Carter WC: Dependability: Basic Concepts and Terminology, Series: Dependable Computing and Fault-Tolerant Systems Volume 5, Laprie, J-C (ed), Springer-Verlag, New York (1992)
2. Arkin RC: Motor Schema based Navigation for a Mobile Robot, Proc. IEEE Conf. Robotics and Automation, Raleigh NC (1987) 264–271

3. Beni G: From swarm intelligence to swarm robotics, Proceedings of the SAB'04 Swarm Robotics Workshop, Santa Monica (2004)
4. Bennett P: Software Development for the Channel Tunnel: A Summary, Journal of High Integrity Systems, **1**(2) (1994) 213–220
5. Bonabeau E, Dorigo M, and Theraulaz G: Swarm Intelligence: from natural to artificial systems, Oxford University Press (1999)
6. Bonabeau E and Theraulaz G: Swarm Smarts, Scientific American, March (2000) 72-79
7. Brooks RA: Cambrian Intelligence: the Early History of the New AI, MIT Press (2000)
8. Butler RW and Finelli GB: The infeasibility of quantifying the reliability of life-critical real-time software, IEEE Trans. Software Engineering, **19**(1) (1993) 3–12
9. Connell JH: Minimalist mobile robotics: a colony-style architecture for an artificial creature, Academic Press Professional, San Diego (1990)
10. Fagan ME: Design and Code Inspections to Reduce Errors in Program Development, IBM Systems Journal, **15**(3) (1976)
11. Harper C and Winfield A: Direct Lyapunov Design – A Synthesis Procedure for Motor Schema Using a Second-Order Lyapunov Stability Theorem, Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, October (2002)
12. Harper C: A Rational Methodology for Designing Behaviour Based Systems for Safety Related Applications, PhD Thesis, University of the West of England, Bristol (2004)
13. Institution of Electrical Engineers: Guidelines for the documentation of computer software for real time and interactive systems, IEE London, 2nd Edition (1990)
14. Lerman K and Galstyan A: A General Methodology for Mathematical Analysis of Multi-Agent Systems, USC Information Sciences Technical Report ISI-TR-529, (2001)
15. Lerman K, Martinoli A and Galystan A: A Review of Modeling Methods for Swarm Robotic Systems, Proceedings of the SAB'04 Swarm Robotics Workshop, Santa Monica (2004)
16. Littlewood B and Thomas M: Reasons why Safety-Critical Avionics Software cannot be Adequately Validated, Proc. 1st UK Safety Systems Symposium, Springer-Verlag (1993)
17. Nehmzow U and Walker K: The Behaviour of a Robot is Chaotic, AISB Journal **1**(4) (2003) 373–388
18. Nembrini J, Winfield A and Melhuish C: Minimalist Coherent Swarming of Wireless Connected Autonomous Mobile Robots, Proc. Simulation of Artificial Behaviour '02, Edinburgh, August (2002)
19. Martinoli A, Ijspeert AJ and Gambardella LM: A Probabilistic model for understanding and comparing collective aggregation mechanisms, In Floreano D, Nicoud JD and Mondada F (eds), Proc. 5th European Conference on Advances in Artificial Life (ECAL-99), Vol 1674 of LNAI, Berlin (1999) 575–584
20. McCabe TA: A Cyclomatic Complexity Measure, IEEE Trans. on Software Engineering, **2**(4) (1976)
21. Şahin, E: Swarm Robotics: From Sources of Inspiration to Domains of Application In Şahin, E., Spears, W., eds.: Swarm Robotics: State-of-the-art Survey. Lecture Notes in Computer Science 3342, Springer-Verlag (2005) 10–20
22. Schöner G, Dose M and Engels C: Dynamics of behavior: theory and applications for autonomous robot architectures, Robotics and Autonomous Systems, **16** (1995)

23. Smithers T: On quantitative performance measures of robot architectures, Robotics and Autonomous Systems, **15** (1995) 107–133
24. Støy K: Using situated communication in distributed autonomous robotics, Proc. 7th Scandinavian Conference on Artificial Intelligence (2001)
25. Winfield AFT: Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots, in Parker LE, Bekey G and Barhen J (eds) Distributed Autonomous Robotic Systems 4, Springer-Verlag (2000) 273–282
26. Winfield AFT and Holland OE: The application of wireless local area network technology to the control of mobile robots, Microprocessors and Microsystems, **23**(10) (2000) 597–607
27. Yokobayashi Y, Collins CH, Leadbetter JR, Arnold FH and Weiss R: Evolutionary Design of Genetic Circuits and Cell-Cell Communications, Advances in Complex Systems, **6**(1) (2003) 37–45
28. Yourdon E: Modern Structured Analysis, Prentice-Hall (1989)
29. Ziemke T: On the role of Robot Simulations in Embodied Computer Science, AISB Journal **1**(4) (2003) 389–399