Analysing the Reliability of a Self-reconfigurable Modular Robotic System

Lachlan Murray[1], Wenguo Liu[2], Alan Winfield[2]
Jon Timmis[1,3], and Andy Tyrrell[1]

[1] Department of Electronics, University of York, UK
{ljm505,jt517,amt}@ohm.york.ac.uk
[2] Bristol Robotics Laboratory, University of the West of England, UK
[3] Department of Computer Science, University of York, UK

Abstract. In this paper, the reliability of a collective robotic system is analysed using two different techniques from the field of reliability engineering. The techniques, Failure Mode and Effect Analysis (FMEA) and Fault Tree Analysis (FTA), are used to analyse and compare two variants of a previously developed 'autonomous morphogenesis' controller. The reliability of the controller is discussed and areas where improvements could be made are suggested. The usefulness of FMEA and FTA as aids to the design of fault tolerant collective robotic systems, and the comparative effectiveness of the two approaches, is also discussed.

1 Introduction

The SYMBRION project [5] is currently in the progress of developing a large scale heterogeneous collective robotic system. Combining both the swarm and self-reconfigurable modular robotics paradigms, robots may act independently in so-called 'swarm mode' or collectively assemble to form large scale 'organisms'.

A major goal of the SYMBRION project is the long-term survival of collective robotic systems. To which end, the consortium have proposed the '100 Robots 100 Days' grand challenge [4]. As the challenge outlines, for 100 robots to survive without any human interaction for 100 days, they will need to exhibit both high degrees of adaptivity to changes in their environment and extreme tolerance to the presence of faulty individuals. The work here focusses on the latter.

By applying Failure Mode and Effect Analysis (FMEA) to the controller of a swarm robotic system, Winfield and Nembrini [11] showed that although in many cases such systems do display high degrees of fault tolerance, in certain scenarios, the effects of a single failure may be catastrophic. Building upon this work, we apply the same technique to the collective robotic system of SYMBRION. We also apply a second contrasting technique from the field of reliability engineering, Fault Tree Analysis (FTA). Two variations of the morphogenesis controller originally developed in [7] are analysed. The controller is introduced in section 2. In sections 3 and 4 FMEA and FTA are performed. In section 5 the outcomes of the analysis are discussed and the FMEA and FTA procedures are compared. Conclusions and potential avenues of future work are presented in section 6.

2 Morphogenesis Controller

The morphogenesis controller of [7] was designed to allow a swarm of robots to self-assemble into an artificial organism of predetermined size and shape. Once assembled, at a later stage, the organism may either completely disassemble, returning every module to swarm mode, or partially disassemble, allowing for the efficient transformation into a different shape. Before describing the controller itself, it is necessary to briefly introduce the robots for which it was developed.

SYMBRION, and the associated REPLICATOR project, are currently developing three unique, yet complimentary, robotic platforms [4]. Here, we are concerned primarily with the so-called 'Backbone' platform. To aid in the formation of organisms, Backbone robots possess precise omnidirectional 2D locomotion, and four docking sides at which other robots may connect. Each docking side contains a set of infrared (IR) components which allows robots to locate and align with each other, a docking element that allows them to physically connect, and an interface through which they may transfer both energy and data.

The docking elements contain a mechanism which physically locks the connection between two robots in place. For a reliable connection to be established between two robots, at least one of the pair must have their element locked.

The IR subsystems that help robots to align with each other are critical to the performance of the morphogenesis controller. Together, the various IR components serve a variety of different functions. Specifically, on each docking side of a robot there are: two IR sensors, for obstacle and robot proximity detection; three IR LEDs, which allow the robot to act as a beacon or broadcast messages to its neighbours; and a single IR remote receiver, for detecting the messages sent by others. For full details of how these components are integrated see [6].

Every robot in the system runs the same behavioural controller, a finite state machine (FSM) for which is shown in figure 1. The behaviours labelled with an 'O' in figure 1 are executed by modules that are currently in the organism, and the behaviours labelled with an 'S' are executed by robots in swarm mode.

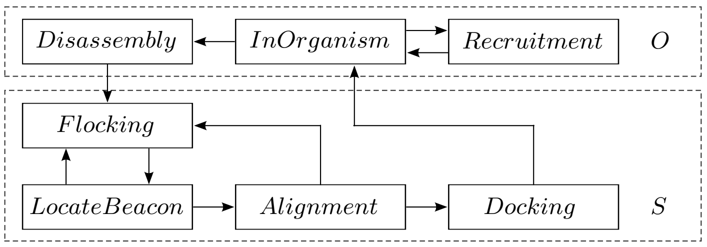


Fig. 1: A finite state machine for the morphogenesis controller of [7].

Figure 1 shows the behaviours of the individual robots. It is also possible to identify 'system level' behaviours which arise from the *interactions* of individual robots. At least three such system level behaviours may be identified: *Ex-*

ploration, *Self-assembly* and *Self-disassembly*. These three behaviours together allow the system to exhibit the property of autonomous morphogenesis.

Exploration is provided by the interactions of the robots in the *Flocking* state. *Flocking* is a place holder for any swarm mode behaviour [7], in this scenario it may more accurately be described as 'obstacle avoidance'. Robots simply move forward and if an obstacle is encountered, avoid it. In an enclosed arena with no internal obstacles, this strategy is sufficient to ensure good coverage.

Self-assembly begins with robots in the *Recruitment* state broadcasting both long-range 'recruitment' messages and short range beacon signals. The detection of 'recruitment' messages causes robots in the *Flocking* state to transition into the *LocateBeacon* state and subsequently, if the beacon signal is detected, into the *Alignment* state. After alignment, in order to establish a reliable connection, a simple docking protocol is executed. At the same time as it locks its own docking element, the docking robot instructs the recruiting robot to do the same by sending a 'docking-ready' message using its IR LEDs. After a short delay, allowing time for the connection to be established, the docking robot transmits a 'docking-complete' message through the wired channel. The recruiting robot responds by transferring information regarding the shape of the current organism.

When a new robot joins the organism it enters the *InOrganism* state. When a module is require to recruit it transitions to *Recruitment*, and when finished transitions back to the *InOrganism* state. The decision as to which robots enter the *Recruitment* state at which point in time, and upon which sides they signal for other robots to join, is determined by a 'recruitment strategy'. Two different recruitment strategies have been developed. The first of which is described in [7] and referred to as '*single entry recruitment*'. In this strategy only one robot may occupy the recruitment state at any moment in time, and may only recruit upon one docking side. The strategy relies on modules propagating messages whenever a new robot joins, so that every robot is aware of the state of completion, and hence when it is their turn to recruit. The second strategy, currently in press [8], is referred to as '*multiple entries recruitment*'. This strategy removes the restrictions upon the number of robots that may recruit simultaneously, and the number of sides at which they may do so. As soon as a robot joins the organism, if it is required to recruit, it immediately transitions to the *Recruitment* state.

The controller also incorporates precautions against the interference that may occur as multiple robots attempt to align with the same recruiting module. For example, whilst aligning, robots broadcast 'expelling' messages, which, if detected by another robot, cause that robot to switch back to *Flocking*.

Self-disassembly begins with modules propagating messages throughout the organism. As soon as an individual is aware that it is time to disassemble it moves into the *Disassembly* state and unlocks its docking element. Robots that are only connected to one other module then remove themselves from the organism by reversing away, and so the process repeats until all of the robots have left. We assume that disassembly begins only once the 3D organism has reconfigured itself into a 2D planar organism, how the organism arrives in this arrangement is considered to be outside of the current scope.

3 Failure Mode and Effect Analysis

Used widely throughout the manufacturing industry, Failure Mode and Effect Analysis (FMEA) is a well established procedure for analysing the safety and reliability of a product or process [9]. For every component in the system under study, the analyst will derive a list of specific 'failure modes' and then attempt to identify all of the effects that these failures may have on the system, in the process building up a general overview of the system's reliability. Because of this progression from specific failures to general effects, FMEA may be described as an *inductive* approach to system analysis. For further details refer to [2] and [9].

To perform FMEA on the morphogenesis controller we follow a similar approach to Winfield and Nembrini [11]. The individual components of the system are considered to be the system level behaviours introduced in section 2. The failure modes or 'hazards', meanwhile, correspond to the complete or partial failure of an individual robot. Like [11], we consider only internal hazards. The six hazards identified by [11] (reproduced in table 1a) form the basis of this work.

Hazard	Description
H_1	Motor failure
H_2	Communications failure
H_3	Avoidance sensor(s) failure
H_4	Beacon sensor failure
H_5	Control systems failure
H_6	Total systems failure

(a)

Hazard	Description
H_M	Motor failure
H_R	IR remote receiver failure
H_S	IR sensor failure
H_L	IR LED failure
H_T	Total systems failure
H_D	Docking element failure
H_W	Wired communication failure

(b)

Table 1: Hazards investigated by [11] (a) and those analysed in this study (b).

The majority of the hazards considered in [11] correspond to the failure of an independent subsystem. In this study, the subsystems responsible for communications, avoidance sensing and beacon sensing are all inter-linked. As shown in table 1b, because of this lack of independence, hazards corresponding to the low level components that make up each subsystem, rather than the subsystem itself, are considered. Motor and total systems failures are kept without alteration.

In section 2 three system level behaviours were identified: exploration, self-assembly and self-disassembly. When an individual robot suffers a failure, the effect that it has on the system will differ depending upon which of the system level behaviours the robot was contributing to when the failure occurred. During self-assembly, the choice of recruitment strategy will influence the effects of any failures. Because we are interested in comparing the two recruitment strategies, during analysis we extract the recruitment strategy decision process from the self-assembly behaviour and consider it as a separate system level behaviour.

Whilst performing FMEA, four different effects were identified, two of which are described as serious and two of which are described as non-serious. All four are listed below, with uppercase lettering used to denote the serious effects:

e_1 - *reduction in the number of capable robots*
e_2 - *delay in the formation of an organism*
E_1 - *stall in the formation of an organism*
E_2 - *stall in the disassembly of an organism*

The various scenarios in which these effects occur are now outlined. The full analysis of all seven hazards is too long to reproduce in full. Consequently, although summaries are provided, details of the effects of hazards H_L and H_R, which were identified as being the least detrimental to the system, are omitted.

H_M - Motor failure A robot that has suffered a motor failure will remain stationary. It will not, however, be prevented from communicating with others.

The effect that a motor failure has on the system depends largely upon where the robot was located when the failure occurred. The failure of a robot that is located outside of communication range of the organism, thus contributing only to exploration, will have very little effect on the system. Since there is no explicit communication between members of the swarm during exploration, a failed robot will not interfere significantly with others. The only effect will be a reduction in the number of capable robots, e_1.

Robots in swarm mode which are contributing to the self-assembly behaviour will be located closer to the organism. The effect of a motor failure in this instance will be far more severe. A robot that stops moving whilst in the *Alignment* state, for example, will not only fail to join the organism itself, but by transmitting 'expelling' messages and physically blocking the path of others, may actively prevent further robots from doing so. If robots are prevented from joining the organism at one or more recruitment sites, the formation of the organism will be permanently stalled, effect E_1.

Analogously, during self-disassembly, a robot that has suffered a motor failure may act as a physical obstacle, preventing itself and others from leaving the organism, resulting in a stall in the disassembly of an organism, or effect E_2.

H_S - Infrared sensor failure IR sensors can fail in a variety of different ways. In this study, no single type of failure is considered, but rather the focus is on the general effects of a breakdown in the correlation between true and sensed values. IR sensors are used for both proximity detection and beacon detection. The failure of one or more IR sensors will reduce the robot's ability to perform both of these functions. Since the recruitment strategies do not require these functions they will not be affected, the other behaviours however, will be.

Without the ability to effectively perform proximity detection, a robot is likely to collide with obstacles or its neighbours. This is not a problem if the robot is contributing only to exploration, in which the effect will simply be a reduction in the number of capable robots, effect e_1. If a robot is contributing to self-assembly or self-disassembly, however, the inability to perform proximity detection may lead to the robot colliding with the organism. By physically blocking the path of other robots, in the worst case, an IR sensor failure may cause both the assembly and disassembly processes to stall, effects E_1 and E_2.

Without the ability to perform beacon detection, robots will be unable to align with the recruiting module, this may simply lead to the robot returning to the *Flocking* state and the exploration behaviour, but in the worst case it may lead to the robot colliding with the organism, again effect E_1.

H_T - Total systems failure A total systems failure —which may be considered equivalent to a robot running out of energy— will result in the shutdown of all of a robot's subsystems. A robot that suffers a total systems failure will be immobilised and unable to communicate with other robots.

During exploration the effects of a total systems failure are negligible. Any robot contributing to exploration that suffers a total systems failure will simply act as a static obstacle to other members of the swarm. The only effect will be a reduction in the number of active robots, e_1.

For similar reasons as to when a motor failure occurs, a robot that suffers a total systems failure whilst blocking the path to or from an organism, may cause both the assembly and disassembly of the organism to stall, effects E_1 and E_2. Note, however, that unlike the effects of H_M there will be no interference due to the robot sending 'expelling' messages. A total systems failure will also prevent recruiting modules from attracting new robots and disassembling modules from unlocking their docking mechanisms, further precursors of effects E_1 and E_2.

Robots which fail whilst executing the recruitment strategy will be unable to transition from the *InOrganism* state to the *Recruitment* state when required. Furthermore, in the case of single entry recruitment, will be unable to propagate messages when a new robot joins the organism. Again leading to effect E_1.

H_D - Docking element failure The failure of a docking element will force the mechanism to remain in the state that it currently occupies. Docking elements do not change state during exploration, or during the execution of either recruitment strategy, their failure in these behaviours, therefore, will have no effect.

Assuming that all robots start with their docking elements unlocked and remembering that the docking protocol introduced in section 2 ensures that only one locked docking element is required for a reliable connection to be established, if a single element fails in the unlocked state, self-assembly will not be affected.

If the docking element of an existing member of the organism fails-locked, however, it will prevent the affected robot (and any connected neighbours) from un-docking. Thus leading to effect E_2, a stall in the disassembly of the organism

H_W - Wired communication failure The failure of wired communications will prevent the affected robot from sending or receiving messages. Furthermore, the failed robot will be unable to act as a node in a wired network. Not only will the failure have a local effect on the failed robot, but any two robots in the organism whose sole path of communication passes through the failed module will also be prevented from exchanging messages. Wired communications are not utilised during exploration, nor are they used during the decision process of the multiple entries strategy, in these scenarios such a failure will have no effect.

During self-assembly, the failure of wired communications in either a recruiting robot or a newly docked robot, will prevent the new module from receiving information about the shape of the organism being constructed. Thus, the assembly of the organism will stall, effect E_1.

During self-disassembly, if a module is unable to propagate messages when it is time to disassemble, sections of the organism may be unaware that it is time to do so. Resulting in a stall in the disassembly of the organism, or effect E_2.

Similarly, in order to determine when a robot should enter the *Recruitment* state, the single entry recruitment strategy requires that messages are propagated throughout the organism when a new module joins. If this is not possible then the formation of the organism will stall with effect E_1.

The effects that all seven of the hazards may have on the performance of the system are summarised in table 2. The three system level behaviours and the two recruitment strategies are considered separately.

Behaviour	H_M	H_R	H_S	H_L	H_T	H_D	H_W
Exploration	e_1	e_1	e_1	-	e_1	-	-
Self-assembly	E_1	e_2	E_1	E_1	E_1	-	E_1
Self-disassembly	E_2	-	E_2	-	E_2	E_2	E_2
Recruitment strategy							
Single entry	-	-	-	-	E_1	-	E_1
Multiple entries	-	-	-	-	E_1	-	-

Table 2: FMEA Summary.

As shown in table 2, of the 21 possible combinations of hazard and behaviour (ignoring for now the recruitment strategies), all but 6 will have some effect on the system. Of the 15 which have an effect, 10 of the effects are described as serious. The most detrimental of the seven hazards is a total systems failure, which has a negative effect on the system in every behaviour, as well as during the execution of both recruitment strategies. The failure of an infrared remote receiver is the least harmful, in no scenario will this hazard have a serious effect on the system. LED failures and docking element failures are the next least critical, each only leading to a single serious effect, in a single behaviour.

In comparing the two recruitment strategies, the only difference that arises is the effect of a wired communications failure. Systems employing both strategies are reliant upon wired communications to ensure that newly docked robots receive information about the shape of the organism being constructed. After a robot has joined the organism, however, only the single entry strategy uses wired communications for determining which robots should enter the *Recruitment* state. With the single entry strategy the failure any robot that prevents messages being propagated to the required modules will cause the formation of the organism to stall. With the multiple entries strategy, meanwhile, only a failure during the initial exchange of information will stall the formation of an organism.

4 Fault Tree Analysis

In contrast to FMEA, Fault Tree Analysis (FTA) [10] is a *deductive* approach to failure analysis. Beginning with a general system failure, the analyst attempts to identify all of the potential causes of the event and the logical sequence of secondary events leading up to it, these together make up the fault tree.

The tree for a particular system failure is constructed by repeatedly breaking the event down into more specific intermediate events. The relationships between events at different levels in the tree are specified using Boolean logic, where the outputs of functions at lower levels serve as the inputs to those at higher levels.

Following the construction of a fault tree, and relying on the fact that the underlying structure of the tree may be described in terms of Boolean algebra, the analyst may perform both qualitative and quantitative analysis (this again differentiates FTA from FMEA, in which only qualitative analysis is possible). Qualitatively, the analyst is able to identify the various combinations of component failures which may cause a system failure, as well as their relative importance. Quantitatively, if the failure rates of the individual components are known, it is possible to calculate the probability of a system failure.

In the remainder of this section, a simple fault tree is constructed for the morphogenesis controller. The tree is used to analyse some of the potential causes of a stall in the formation of an organism. As we are limited by space, however, only a small part of the entire fault tree for this event is presented. Fault trees are represented graphically using an extension of standard logic gate notation. In this work, only a subset of the available notation is used, with symbols introduced as and when they are required, for a full list see [10]. Following construction, the tree is analysed qualitatively. The combinations of component failures which may lead to a stall in the assembly process are identified and their relative importance discussed. Comments are made upon the levels of fault tolerance in the current system and areas in which this could be improved are identified.

Construction Before beginning construction we must state more precisely what is meant by a "stall in the formation of an organism". A stall in the formation of the organism can be said to have occurred if at least one robot has been unable to recruit the modules that it was required to (within some acceptable time limit). This means that one or more robots have either become stuck in, or failed to transfer to, the *Recruitment* state. When investigating the multiple entries recruitment strategy we need only to consider the scenario in which a robot becomes stuck in the *Recruitment* state, since the time spent in the *InOrganism* state (before a robot has completed its recruiting duties) is negligible. With the single entry strategy we must consider both cases. Here we examine only the first scenario, that a robot has become stuck in the *Recruitment* state.

The only condition that will cause a robot to leave the *Recruitment* state is the receipt of confirmation (through wired communications) that another robot has docked with it. As shown in figure 2, where $R1$ is the robot that is awaiting confirmation, this statement constitutes the topmost event of our fault tree. In fault tree notation, events are represented using rectangular boxes.

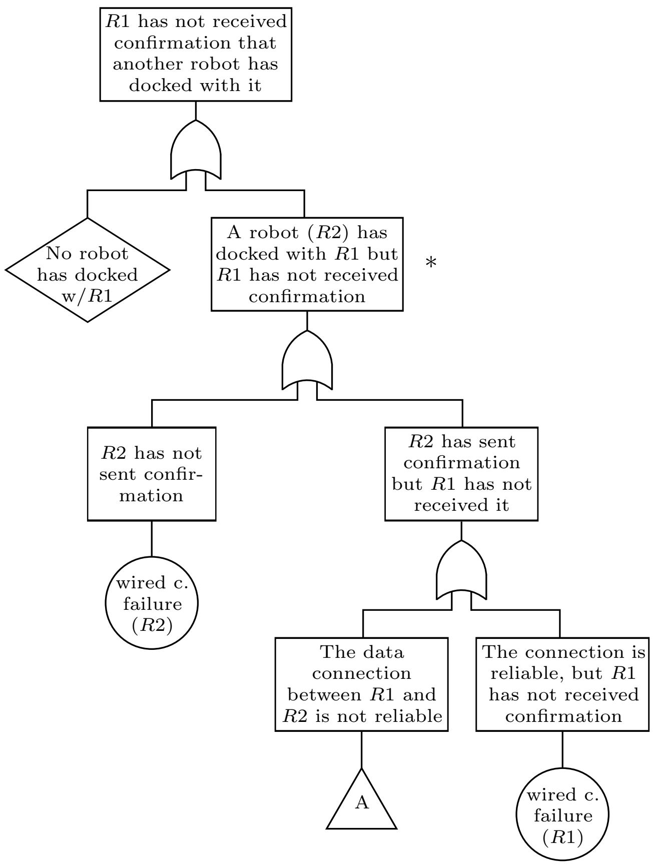

$R1$ has not received confirmation that another robot has docked with it
No robot has docked w/$R1$
A robot ($R2$) has docked with $R1$ but $R1$ has not received confirmation
*
$R2$ has not sent confirmation
$R2$ has sent confirmation but $R1$ has not received it
wired c. failure ($R2$)
The data connection between $R1$ and $R2$ is not reliable
The connection is reliable, but $R1$ has not received confirmation
A
wired c. failure ($R1$)

Fig. 2: Fault tree examining the causes of a stall in the formation of an organism.

We now consider the reasons why $R1$ has not received confirmation. There are two options, either no robot has docked with $R1$, or a robot ($R2$) has docked with $R1$ but for some other reason $R1$ has not received confirmation. These options are separated using an OR gate, representing the fact that the output of this gate will occur if at least one of the inputs occurs. The scenario in which no robot has docked with $R1$ is not pursued further, and as such this 'undeveloped event' is presented in a diamond shaped box. We now focus on the case in which another robot has docked with $R1$. In this context, we consider two robots to have 'docked' once they have aligned and moved into the docking position, communication between the pair and the engagement of either docking mechanism are not considered necessary for docking to be said to have taken place. There are two immediate reasons why $R1$ would not receive confirmation of docking from $R2$. The first is that $R2$ has not sent confirmation, the second is that $R2$ has sent confirmation but $R1$ has not received it. Considering the first leads to the first basic cause of a stall in the assembly of an organism, that

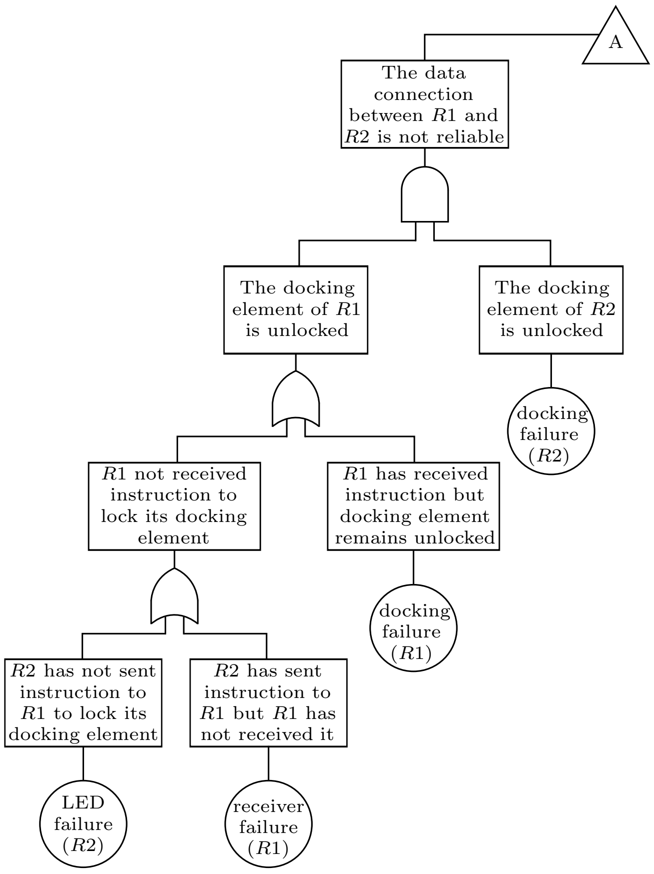

A
The data connection between $R1$ and $R2$ is not reliable
The docking element of $R1$ is unlocked
The docking element of $R2$ is unlocked
docking failure ($R2$)
$R1$ not received instruction to lock its docking element
$R1$ has received instruction but docking element remains unlocked
docking failure ($R1$)
$R2$ has not sent instruction to $R1$ to lock its docking element
$R2$ has sent instruction to $R1$ but $R1$ has not received it
LED failure ($R2$)
receiver failure ($R1$)

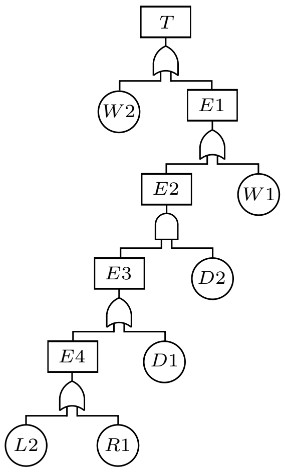
Fig. 3: Fault tree examining the causes of a stall in the formation of an organism.

$R2$ has suffered a wired communications failure. Such ‘basic fault events’ are represented by circles. The second case, that $R2$ has sent confirmation but $R1$ has not received it, requires further development.

There are two possibilities why, although $R2$ has sent confirmation, $R1$ has not received it. The first is that the data connection between the two robots is unreliable. The second is that although the data connection is reliable, $R1$ has still not received confirmation. In this second scenario, given that confirmation was successfully sent by $R2$, the problem must lie with the wired communications of robot $R1$. The case of an unreliable data connection is developed further in figure 3, triangular ‘transfer symbols’ are used to connect the two diagrams.

Using an AND gate, the output of which will occur only if both inputs do, the first part of figure 3 shows that for an unreliable connection to be formed between $R1$ and $R2$, the docking elements of both robots must be unlocked. Remembering the docking protocol introduced in section 2. Since $R2$ is not reliant on any communication with $R1$ in order to lock its docking element, the only reason that it would remain unlocked is a failure of the device itself. $R1$ on the other hand will only lock its element once it receives instruction from $R2$ through an IR receiver, or when a wired connection is already established. Since we are assuming that a reliable wired connection has not been established, there are two reasons why $R1$'s docking element remains unlocked, either $R1$ has not received instruction to lock it, or it has received instruction, but in attempting to lock it has been prevented by the presence of a docking element failure.

The scenario in which $R1$ has not received instruction to close its docking element leads to the final two basic causes of a stall in assembly, and to the end of the fault trees construction. For $R2$ to instruct $R1$ to close its docking element both the LED of $R2$ used to send the message and the IR receiver of $R1$ used to receive it must be functioning. If either of these components have failed $R1$ will not receive the instruction to close its docking element.



$$\begin{aligned}T &= W2 + E1\\ E1 &= E2 + W1\\ E2 &= E3 \bullet D2\\ E3 &= E4 + D1\\ E4 &= L2 + R1\end{aligned}$$

Fig. 4: A minimal version of figures 2 and 3, with equivalent boolean equations.

Analysis The first step of analysing a fault tree involves representing it in terms of boolean equations. Ignoring the undeveloped event, figures 2 and 3 can be minimally represented as figure 4. Where the basic fault events WX, DX, LX and RX correspond respectively to wired communications, docking element, LED, and receiver failures of robot 'X'. The equivalent boolean equations of the minimal tree are shown along side it in figure 4.

With the tree converted into Boolean equations, the next task is to obtain the "minimal cut sets". A minimal cut set is any combination of basic events which, if they all occur, will cause the top event to occur. The minimal cut sets can be obtained simply by reducing the equations of the fault tree until only basic fault events are left. The equations in figure 4 are easily converted into:

$$T = (D1 \bullet D2) + (D2 \bullet L2) + (D2 \bullet R1) + W1 + W2$$

The minimal cut sets then, are simply: $\{D1, D2\}$, $\{D2, L2\}$, $\{D2, R1\}$, $\{W1\}$ and $\{W2\}$. Where, for example, $\{D1, D2\}$ represents the scenario in which both a recruiting module and a newly docked robot both suffer docking element failures.

For this simple example, deriving the minimal cut sets was trivial. However, it should be remembered that this is only a subsection of a much larger fault tree. To determine the minimal cut sets of more complex fault trees requires greater effort. Fortunately, software exists to aid the analyst during both the construction and analysis of fault trees. One example of FTA software which was used to help develop the fault trees in this work is OpenFTA [1].

Even though we have only considered a subsection of the overall tree, it is still possible to make some interesting qualitative observations. From the point of view of reliability, single component minimal cut sets are undesirable, simply because they imply the failure of a single component may cause the entire system to fail. Even without reducing figure 4 to the minimal cut sets of its Boolean equations it is obvious from the lack of AND gates that the system will contain single component cut sets. In this case, those cut sets are $\{W1\}$ and $\{W2\}$, corresponding to the scenarios in which either a recruiting robot or a newly docked robot has suffered a wired communications failure. Figure 4 does contain one AND gate, and its presence gives rise to the three remaining two-component cut sets, all of which it may be noted, contain $D2$. The design of the hardware and the simple docking protocol, in this situation, ensure that the single failure of one of the involved components does not lead to a stall in the assembly process.

As a general rule, it is desirable to have AND gates positioned as high up the fault tree as possible. With this in mind it is possible to suggest some improvements to the system. A weak point in the fault tree of figure 2 may be identified at the '$*$'. Although recruiting modules are provided with information about the presence of newly docked robots in the form of both wired and wirelessly transmitted messages, as well as the values returned by their IR proximity sensors, they are critically dependent upon the wired channel to determine their behaviour. Recall that the only time at which a robot will leave the *Recruitment* state is when it receives confirmation, through wired communications, that another robot has docked with it. It is suggested, therefore, that rather than relying solely on wired communications to dictate their behaviour, recruiting modules should make better use of the data from their IR receivers and sensors. If robots react appropriately to the information (or lack thereof) that all three channels provide, the chances of a robot becoming stuck in the recruitment state, following the docking of another robot, may be severely reduced. For example, detecting the presence of another robot without receiving confirmation through wired communications may signal to the recruiting robot that there is something wrong. Thus providing the robot with another way out of the recruiting state by, for example, initiating the processes of repairing or replacing either itself or the docking robot. The fault tree of a controller adapted in this manner, no longer being singularly reliant on wired communications, would be expected to contain more AND gates, located further up the tree. The system, therefore, would exhibit greater tolerance to the failure of individual components.

5 Discussion

Whereas FMEA is an inductive approach to failure analysis, FTA is deductive. When performing FMEA, the analyst begins with a list of specific hazards and attempts to identify what general effects they may have on the system. Conversely, with FTA, the analyst begins with a general effect and gradually traces it back to its root causes, the specific hazards. In practical terms, this means that by employing FTA, the analyst is forced to focus on identifying increasingly specific causes. Resulting in a very complete understanding of the chain of events that lead from a component failure to a system failure.

One advantage of the precise knowledge that the FTA procedure provides, is that it may help the analyst to identify specific weak points in the system. This was shown in section 4 where the over reliance of recruiting robots on wired communications was identified as a weakness. What the FMEA procedure lacks in terms of its depth of detail, it makes up for with the breadth of information that it provides. In section 3 the FMEA procedure covered three different effects of five different single component failures. In a similar amount of space, with significantly greater effort, the FTA procedure covered only a subsection of a single effect, and revealed only one single component cause.

What the FTA procedure does reveal with ease, which may be less obvious when performing FMEA, are the different *combinations* of component failures which may lead to a system failure. Furthermore, although in general, FMEA is better suited to the task of exhaustively listing possible hazards, the two approaches both have the potential to identify hazards overlooked by the other.

Another advantage of FTA, which was not exploited in this work, is the ability to perform quantitative analysis. One of the reasons why quantitative analysis was not carried out is because it is reliant upon failure rate data, which was not available. Another more pressing reason, however, indicative of a much larger problem with the application of FTA to collective robotic systems, is that the systems themselves may be too complex to model in sufficient detail. When constructing the fault trees here, assumptions were made about the state of the system, in particular with regard to the interactions between robots and their environment. The tree presented in section 4 examined the system in a relatively static state, considering the interactions between only two robots. Other, more dynamic parts of the system require increasingly limiting assumptions. Allowing for a more complete understanding, but at the expense of detail in the model. Without modelling the interactions between robots accurately, it is not possible to produce accurate quantitative results. The dynamic nature of these systems, as well as the large numbers of agents and the locational dependency of their interactions make such systems difficult to model using standard fault trees.

The desire to model the reliability of complex fault tolerant systems is widespread. The difficulties that arise in accounting for the high levels of redundancy, fault recovery mechanisms and sequentially dependent failures that such systems possess are not unique to robotic systems. Efforts to help solve these problems have led to the development of Dynamic Fault Trees (DFT) [3]. In constructing DFTs, additional ‘dynamic’ gates are used to account for the extra complexity. Analysing DFTs then essentially involves combining the standard fault tree approach with Markov Chain models. Thus, the simplicity of the standard approach is augmented by the flexibility of Markov models [3].

The analysis carried out in section 3 revealed that 15 of the 21 possible combinations of hazard and behaviour will have some effect on the system, 10 of which will be serious. This picture is stark in contrast to the results of the study by Winfield and Nembrini [11], in which there were observed to be only 6 serious effects out of a possible 30. While performing FMEA, a pessimistic view was taken in assuming that all components of a single subsystem will fail simultaneously. Whilst in some cases, the effect of a failure will depend upon how many and which components of a subsystem fail, in others, the failure of a single component may be equally as detrimental as the failure of multiple. The poor outlook, therefore, cannot be attributed solely to pessimism. Nor should we be quick to denounce the long held belief that swarm robotic systems inherently provide fault tolerance, even if in certain scenarios, as shown by [11], its foundations are questionable. We are not dealing with a pure swarm robotic system, but with a combined swarm and modular robotic system. The modular aspect brings with it far greater dependence between robots. If we examine the system closer, we see that it is during the interactions with modules already in the organism where the effects of a failure are most severe. Note that in table 2 there are no serious effects when a failure occurs during exploration, the only behaviour that involves robots operating exclusively in swarm mode.

To endow a modular robotic system with levels of fault tolerance similar to those found in swarm robotic systems, greater plasticity in the conformation of the system is required. Whereas in swarm robotic systems, faulty robots may simply be ‘left behind’, in modular robotic systems, more explicit methods of replacing or repairing modules are required. The SYMBRION robotic platforms were designed with this in mind, supporting the ability of robots to recharge or directly power their connected neighbours. As a result, in our pursuit to design fault tolerant modular robotic systems, we may be encouraged by the fact that what appears to be the most detrimental hazard, a total systems failure, is also the easiest to repair (if assumed to be caused by a robot running out of energy).

In this analysis we did not consider the effects of either transient or external hazards. If transient hazards are considered, or if it is possible to repair or replace failed modules, the multiple entries strategy possesses a significant advantage. With the multiple entries strategy, even though the assembly of an organism may stall at one point, it may continue at others. A system utilising the multiple entries strategy is therefore able to recruit and repair simultaneously. In considering external hazards, such as the interference observed near recruitment sites, the single entries recruitment strategy possesses the advantage. However, with the multiple entries strategy, the increased level of interference is offset by the fact that multiple robots may recruit simultaneously. The time saved by the multiple entry strategy directly translates to a saving in energy, which may be critical to the long-term survival of the system, and consequently, of great importance to the completion of the ‘100 Robots 100 Days’ grand challenge.

6 Conclusions and Future Work

FMEA and FTA have been applied to the morphogenesis controller of a combined swarm and self-reconfigurable modular robotic system. The analysis revealed several scenarios in which even a single failure may have serious consequences, the worst case being the occurrence of a total systems failure. FMEA and FTA were each identified to have their own advantages. Whereas FMEA provides a good general overview, FTA reveals specific details about the chain of events that may lead to a system failure. A combined approach, as demonstrated here, is advocated during the design of fault tolerant collective robotic systems.

Using the knowledge gained in this study, future work will investigate ways of improving the fault tolerance of the morphogenesis controller. The focus will primarily be on addressing total systems failures, particularly within the scope of energy management. Further analysis of the both the current controller and any improved controllers, may also be performed. For which purpose, DFTs and Markov chain models have been identified as a promising future prospects.

Acknowledgments. The SYMBRION project is funded by the European Commission, within the 7th Framework Programme. Project No. FP7-ICT-2007.8.2

References

1. Auvation. OpenFTA, 2011. [Computer software] http://www.openfta.com.
2. Kenneth W. Dailey. *The FMEA Pocket Handbook*. DW Publishing Co., 2004.
3. J. Dugan, S. Bavuso, and M. Boyd. Dynamic fault-tree models for fault-tolerant computer systems. *Reliability, IEEE Transactions on*, 41(3):363–377, 1992.
4. S. Kernbach, O. Scholz, K. Harada, S. Popesku, J. Liedke, H. Raja, W. Liu, F. Caparrelli, J. Jemai, J. Havlik, E. Meister, and P. Levi. Multi-Robot Organisms: State of the Art. In *ICRA10, workshop on "Modular Robots: State of the Art", Anchorage*, pages 1–10, 2010.
5. P. Levi and S. Kernbach. *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*. Cognitive Systems Monographs. Springer, 2010.
6. Wenguo Liu and Alan Winfield. Implementation of an IR approach for autonomous docking in a self-configurable robotics system. In T. Kyriacou, U. Nehmzow, C. Melhuish, and M. Witkowski, editors, *Proceedings of Towards Autonomous Robotic Systems*, pages 251 – 258, September 2009.
7. Wenguo Liu and Alan F. T. Winfield. Autonomous morphogenesis in self-assembling robots using IR-based sensing and local communications. In *Proceedings of the 7th international conference on Swarm intelligence*, ANTS'10, pages 107–118, Berlin, Heidelberg, 2010. Springer-Verlag.
8. Wenguo Liu and Alan F.T. Winfield. Distributed autonomous morphogenesis in a self-assembling robotic system. In R. Doursat, H.Sayama, and O. Michel, editors, *Morphogenetic Engineering: Toward Programmable Complex Systems, to appear.*
9. R.E. McDermott, R.J. Mikulak, and M.R. Beauregard. *The Basics of FMEA*. Productivity Press, 2008.
10. W.E. Vesely and N.H. Roberts. *Fault Tree Handbook*. U.S. NRC, 1981.
11. A. Winfield and J. Nembrini. Safety in numbers: fault-tolerance in robot swarms. *International Journal of Modelling, Identification and Control*, 1(1):30–37, 2006.