Modelling for Systems with Holistic Fault Tolerance

Rem Gensh, Ashur Rafiev, Fei Xia, Alexander Romanovsky and Alex Yakovlev

Newcastle University, Newcastle upon Tyne, UK
{r.gensh, ashur.rafiev, fei.xia,
alexander.romanovsky, alex.yakovlev}@newcastle.ac.uk

Abstract. Trade-offs between extra-functional properties, such as performance, reliability and resource utilisation, have been recognised as crucial in system design. The concept of Holistic Fault Tolerance (HFT) is aimed at targeting these trade-offs in run-time system control. Previous work has shown that HFT systems can have significant complexity, which may require sophisticated modelling at the design stage. This paper presents a novel HFT design methodology based on hierarchical modelling and stochastic simulations. The former caters to system complexity and the latter estimates extra-functional properties in the trade-offs. The method is demonstrated with an application example of number plate recognition software.

Keywords: Modelling, Holistic Fault Tolerance, Order Graphs, Stochastic Activity Networks, Extra-functional properties.

1 Introduction

System modelling aims to create an abstract representation of a designed system. This process assists in better understanding of the system and gives a possibility to find and eliminate potential problems at early stages of system development. However, for complex systems, the system model can also be complex and difficult to use. Therefore, it is necessary to ensure that only important parts of the system are studied during the modelling to reduce comprehension complexity.

A well-accepted method of controlling model complexity is the use of hierarchical models. Different models may be constructed for the same system or subsystem at different levels of abstraction. High-level models of high degrees of abstraction tend to be small and easy to analyse, but also include few details and may provide low representative resolution or precision for the quantities or parameters being studied [1]. On the other hand, low-level models of less abstraction may offer finer grain representation of system details and provide higher resolution for studied parameters. However, they may have high degrees of complexity and difficult to work with. Hierarchical modelling provides designers with a means of trading off modelling quality with model usability.

Another popular method of dealing with the model complexity issue and at the same time handling run-time unpredictability is stochastic modelling. Quantities and parameters under study are assumed to be stochastic and models of manageable size

can be used to estimate such quantities without precise knowledge of all the contributing factors such as run-time eventualities [2].

Aspects of study during system design and analysis include functional behaviour and extra-functional parameters. Functional correctness is important, but extra-functional parameters can also be significant contributors of the success or failure of a design. The most interesting extra-functional parameters attracting the attention of system designers include performance, energy consumption and reliability.

In previous studies, we introduced the notion of Holistic Fault Tolerance (HFT) [3] and showed that the HFT architecture can be applied to implement the system, taking into account extra-functional properties, such as reliability, performance and resource utilisation [4]. And finally, maintainability evaluation of the HFT architecture was provided in [5].

In these investigations, it was demonstrated that the HFT approach provides better maintainability of fault tolerance mechanisms. The HFT architecture includes system components, an HFT controller and a number of agents which supports interactions between the components and the HFT controller. The HFT run-time is implemented through control loops that manage the extra-functional parameters through component configuration. However, there remain challenges faced by the HFT developer during the design stage. It is not always clear how to choose the system components, which will be involved in the interaction with the elements of the HFT architecture (essentially the number of control loops). If the designer chooses to involve all system components in the interaction with the HFT elements, i.e. have the maximum number of all possible control loops included, the system would be extremely complex for modelling, implementation and maintenance. On the other hand, unguided control loop reduction would rarely result in optimal system designs.

This study focuses on modelling that supports design-time and run-time system optimisation through the (re)configuration of system components and the efficient use of control loops. At the same time the model should not be very complex for understanding. Iterative top-down design and stochastic representation of extra-functional parameters offer promising solutions.

In this paper we propose a general design method supporting HFT systems. This method makes use of a hierarchical model language, known as order graphs (OGs) [6], which has good representations of horizontality and verticality issues and good support for having different levels of abstraction for different parts of a system model. Also included is an established stochastic model language, known as stochastic activity networks (SANs) [2], which provides facilities such as state-space analysis and simulation engines.

The proposed design workflow is based on the following key points:

- The characterisation of system components leading to SANs models. These SANs models can be used to provide estimates of the extra-functional parameters under study (usually reliability, system utilisation and/or performance) and generate importance costs for potential control loops in the HFT control.
- The concept of controllability is applied to minimise the number of control loops.

- The development of a hierarchical model of the HFT system based on OGs. This model can be used to validate the existence of control loop paths at all levels of model abstraction.

This paper is organised as follows. Section 2 provides the background describing SANs, OGs and HFT. Section 3 explains our modelling methodology. Section 4 describes an HFT case study. Section 5 concludes the paper with discussions.

2 Background

2.1 SANs and Stochastic modelling

SANs is an extension to general stochastic Petri nets (GSPNs) which are based on Petri nets (PNs) [7]. It inherits the general attributes of PNs including a distributed representation of system states, making it easy to represent parts of a system directly as local subsystems, and more straightforward representations of such important issues as concurrency and synchronisation. A well-established method, it is supported by the mature software tool-kit: Möbius [8].

SANs are capable of representing both deterministic and stochastic events, and event durations in time. The elements used in this work include a) transitions whose firing speeds (rates) are specified as stochastic, following given distributions, b) transitions with multiple firing cases with specific probabilities for each case, and c) input and output gates with predicates and implications specified through logic functions.

The Möbius tool, used in this paper, incorporates a set of solvers including both Monte-Carlo simulation and statespace related solvers. Numerical Markovian solutions can be done for steady-state or time averaged interval rewards, but limited to models with exponentially distributed firing rates. The tool's concept of "rewards" can be easily extended to physical parameters, such as power. In this work we use rewards to evaluate system's extra-functional properties including performance, reliability (defined as success rate), and resource utilisation.

2.2 Order Graphs and resource modelling

Hierarchical representations have been used for modelling complex systems for a long time. The idea of separating the "vertical" relation between the layers of abstraction from the "horizontal" knowledge of the system at each particular layer of abstraction has been hinted in [9] and then formally defined in Zoom structures [10] as the concepts of verticality and horizontality. Zoom structures are based on partial orders and are very permissive. In contrast, OGs put a number of constraints on the modelling, which guarantee consistency between the abstraction layers.

An OG is a graph with nodes representing various system resources arranged in tree hierarchies. The hierarchies can be built from the knowledge of the system structure and by similarities of its constituents. The distance from the root relates to the level of abstraction. The formal definition and properties can be found in [6].

The modelling using OGs is an iterative top-down process, starting from the most abstract representation of the system and gradually adding more details, when moving to lower levels. The dependencies between the system's components at the same level of abstraction are represented with "horizontal" arcs in the graph, hence the horizontal paths represent transitive dependencies between the elements in the system. The rigorous definition of OGs provides a built-in capability of consistency checking by preserving the resource dependency paths at each level of abstraction.

OG contains the static knowledge of the system and needs to be paired with a dynamic model to capture the system behaviour (in our case: SANs). The nodes in OG that are included in this model relation form a cut. If the cut goes through different depths in the hierarchy (layers of abstraction), it is called a cross-layer cut. The cut containing all leaves relates to the most concrete (detail) model of the system. Moving up in the abstraction hierarchy, thus grouping multiple nodes into one, represents grouping the corresponding elements in SANs into a single entity by averaging/totalling their parameters (known as black-boxing). This reduces the size of a model, but also introduces inaccuracy. The trade-off between the model complexity and accuracy can be achieved from manipulating cross-layer cuts. This method, called selective abstraction, has been explored in details in [1].

2.3 Holistic Fault Tolerance

The idea of Holistic Fault Tolerance was introduced and developed in [3, 4]. There are two goals of the HFT architecture. The first goal is to provide a method that allows the developer to design and implement computer systems that are efficient in terms of performance and resource utilisation. The second goal is to improve the software maintainability of fault tolerance mechanisms in the system.

A computer system implemented in accordance with the HFT architecture includes functional components that are responsible for main system tasks and the HFT part. The HFT part controls the functional components and ensures reliable and efficient system operation. This part is built around the HFT controller, which is responsible for distribution of computation resources in the application and provides an overarching control over the extra-functional properties, such as system performance. HFT controller also performs a task of re-configuring the system in run-time in case it finds a better operating point.

The HFT controller interacts with the system components through a set of public interfaces. Additionally, the controller is assisted with the HFT agents – auxiliary objects aimed at decreasing the complexity of the HFT architecture. Each HFT agent is responsible for certain extra-functional property of the system. The HFT agents monitor and, when it is required, intervene in the control flow of the critical components. The typical HFT agent can be responsible for one of the following activities: performance monitoring, error handling and gathering of diagnostics information. The structure of an agent can depend on the components it works with, the idea is that the agent incorporates component-dependent code in order to keep HFT as flexible as possible. The data gathered by the HFT agents are translated into a component-independent format and transferred to the HFT controller for dynamic analysis. In

case of error handling, the HFT agent requests the HFT controller for a suitable handling action.

It is advised to implement the link between an HFT agent and a functional component implicitly for the component. This approach significantly reduces the dependence of the component on the HFT agent, thus the component would focus on implementing only its functional task without tangling with non-functional activities. In our previous works [4, 5], we used Aspect Oriented Programming [11] in order to improve the development cycle and reduce maintainability effort. In general, the decisions on the structure of HFT agent heavily depend on the software design and the tasks of the system, however the aim of this work is to address the design decisions in a methodological way.

3 Modelling methodology

In this section, we consider the goal of the modelling, define extra-functional properties of the system, and the context in which these properties are analysed. We also introduce the workflow of the modelling approach.

The goal of the modelling is to provide a method that allows the developer to design and implement the system based on the HFT architecture. It is necessary to guarantee that the system will be efficient with regards to extra-functional properties, such as reliability, performance and resource utilisation. The modelling assists in defining efficient points of the interplay between these extra-functional properties. An *efficient design* allows the developer to implement such a system, which will be efficient in terms of this interplay.

Performance is considered as the amount of work completed per unit time. Faster operation typically requires more resources or can be achieved by reducing the quality of computation. The work performed by the system is measured in *work units*. The processing of each work unit can be finished successfully or unsuccessfully.

Reliability is represented with the success rate, which is defined as the ratio of successfully finished work units to the total amount of work units.

Resource utilisation is the amount of computer resources required to process a certain number of work units. In this context, we define a resource as any facility that enables computation, which may include CPU cores, application threads, memory, energy, etc.

As mentioned in Section 2, the system contains functional components that implement the computation. The HFT control for the extra-functional properties is realised using *knobs and monitors*. The knobs are provided by system components as configuration points, and the monitors are instrumentation that provides readings of extra-functional properties at the component level. The system-wide set of knob states is called a *system configuration*. During the system operation, the HFT controller dynamically choses the most suitable system configuration, depending on the history of monitor data.

An instance of such interaction between the HFT elements and the functional system components is defined as a *control loop*. It can be considered as a special inter-

face between the components and the HFT part. The control loop is managed only by the HFT part and is implicit to the system components.

3.1 Workflow of the modelling approach

The workflow of the HFT system modelling approach is described in Fig. 1. Each of the steps is described in a subsequent subsection. Note that order graph modelling happens in parallel to the right hand main branch of the workflow.

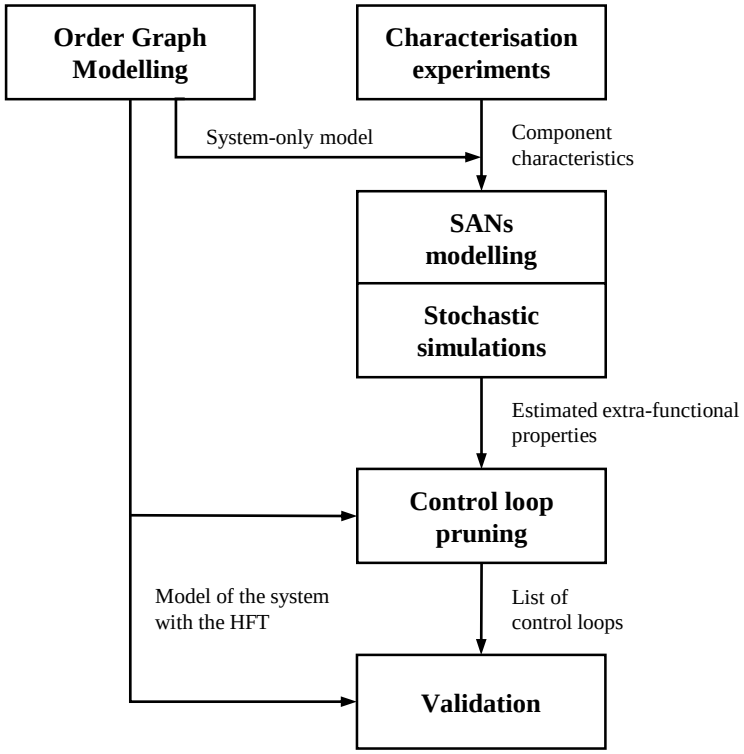

Fig. 1. Workflow diagram

3.2 Characterisation of the system's extra-functional components

The designer should characterise the extra-functional properties of each individual component. If the component supports multiple configurations or algorithms of processing, the characterisation should be done for each individual configuration. The full result of a characterisation pertaining to some component and some extra-functional property describes the value of that property when executing that component. Characterisation is not done beyond component level.

3.3 Building and simulating the SANs model of the system

In this step the SAN model of the system is built using component characterisations from the previous step and the system-only OG model. The granularity of the SAN model for any part of the system is determined by the OG modelling step (Fig. 1) and the parameter values are obtained from the characterisation step. The characterisation step usually pertains to the SAN model of the finest detail, because there is no point of developing a SAN model at a finer level of detail than the existing characterisation data. If the OG step suggests a higher level of abstraction, it is possible to derive SAN models of less detail than the characterisation data, for instance by running simulations at the characterisation level of detail then abstracting from the results.

From characterisation to the final SAN model for simulations the approach is bottom-up, but the OG step is usually top-down. There is no conflict because in order to determine the granularity of the final SAN model the entire OG model covering all levels of abstraction needs to have been established. In a way discovering the SAN model is a process of raising the level of abstraction from the bottom traversing the OG until a satisfactory SAN has been found.

The preferred tool for working with SANs is Möbius [8]. The main point of this step is that the SAN system model, assembled from component models, supports system-wide analysis of the modelled extra-functional properties from component-level characterisation data. The most practical analysis method for SAN models of HFT is simulation, as other forms of analysis such as state space studies tend to be restricted to very small models. However, Möbius does provide non-simulation solvers if and when they can and need to be used.

3.4 Control loop pruning

The estimated values of system-wide extra-functional properties, obtained from the previous step, can be used to reduce the complexity of the HFT controller, by eliminating unnecessary control loops.

The method is based on the problem of preserving controllability [16] while reducing the number of knobs. It assumes that the number of monitors is both sufficient and necessary to represent the extra-functional properties under study. The monitor values are considered state variables.

We use simulations to build system transfer function [16] relating knobs to monitors. This is achieved by analysing differentials in the estimated monitor values from simulations. Ideally, this requires an exhaustive set of simulation covering all combinations of knob values. However, it is possible to apply known optimisation methods, such as Monte-Carlo [17], to improve the usability of the method.

From this database of state relations, it is possible to determine the smallest set of knobs that maintains controllability.

Although in this paper we deal only with deciding what control loops to include in an HFT system, the off-line design flow described here can yield valuable quantitative data that may be helpful for the detailed design of run-time control. For instance, the

SAN models may provide a set of reference points which may be used in the designs of individual control loops.

3.5 Validation using OG hierarchy

As mentioned in Section 2, OG modelling provides a top-down workflow that helps the developer to incrementally add the details in the system design. In the proposed workload, the dependencies in the graph represent interactions between the element of the system and provides paths for the control loops. A rigorous path consistency checking between the layers of abstraction guarantees that the designed HFT controller is consistent with the control loops established in the previous steps of the workflow.

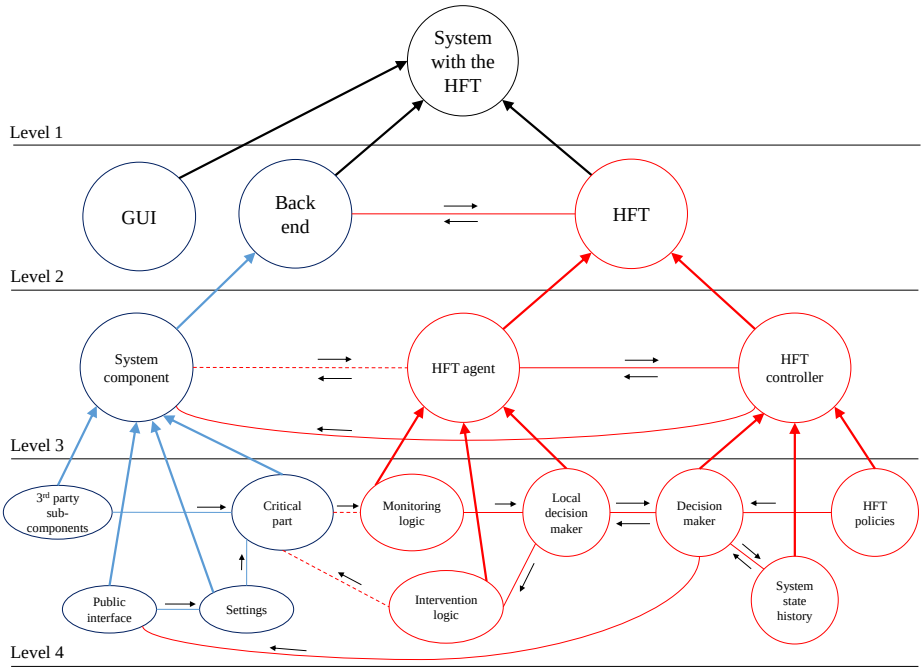

Fig. 2. A general template for HFT Order Graph model

Fig. 2 illustrates the hierarchical model of the system with the HFT architecture in three levels of detail. At the top level, there is only the system with the HFT architecture. The second level contains graphic user interface, backend or functional part of the system and HFT part. Information flow between the backend and the HFT is shown in both directions. The next level represents the backend is decomposed to the system components and the HFT part decomposed to the HFT controller and HFT agents. For simplicity, the figure shows only one component and only one agent. At this level, the control loops between the system components and the HFT elements should start to appear.

The most detailed level of the hierarchical model considers the inner structure of the system components and the HFT elements. The system component may include third-party subcomponents, public interfaces, component settings and critical parts. A possible internal structure of an HFT agent consists of monitoring logic, intervention logic and local decision maker. The HFT controller includes the decision maker, the system state history (or dynamic HFT data) and the HFT policies (or static HFT data).

Connections represented by the dashed lines assume that for better maintainability it is preferable to implement this link in such a way that the system component was not aware of implementation details of monitoring and intervention logic in the HFT agent. HFT agents do not directly provide performance of reliability benefits. They were introduced to simplify the developing and improve understanding of the systems with the HFT. It was shown [5] that such configuration supports maintainability of the system. This is the reason why we consider decomposition of the HFT architecture to the HFT controller and the HFT agents.

4 Use case

4.1 Case study application

As a use case, we have chosen the application for the recognition of the UK number plates [4, 5]. The input of the application is a set of images. As an output, the application links each image with recognition results that include the contour of the number plate, recognised string and the probability of correct recognition.

The functional part of the application consists of several components. The Graphical User Interface (GUI) component is the frontend of the application, which allows the user to upload the images. These images are sent to the Initial Image Processing (IIP) component. At this stage, every image undergoes an initial processing, which includes various filters, searching of the number plate on the image, cropping of the number plate from the image and elimination of the perspective skews of the number plate cutout. Two algorithms for number plate search can be applied: OpenCV-based rectangle detection and HAAR cascade [12] trained to recognise the area with the UK number plate. If the number plate is found and cropped it is put to Number Plates Queue (NPQ). When the NPQ is not empty, the Optical Character Recognition (OCR) component takes available number plate cutout and performs the text recognition on the cutout. There are two OCR algorithms in the OCR component: Tesseract [13] and number plate recognition algorithm described in [14]. If the OCR recognises the text on the cutout, this text is checked by the Result Checker (RC) component to ensure compliance of the car number with a national format. These additional algorithms are introduced to provide redundancy and increase reliability of the application.

The UML diagram of the application is shown in Fig. 3. GUI does not participate in the HFT scheme and it should not be considered in details in the model. Interfaces between the functional components (IIP and OCR) and the HFT controller are omitted to make the diagram clearer.

In both IIP and OCR components the images are processed concurrently. The HFT controller specifies the most suitable number of working threads for each component.

The Performance agent monitors the execution time of the IIP and OCR components. The Error Handling agent is responsible for handling the errors in the IIP and OCR component. An error implies a deviation from the correct service [15] and it is not necessarily exception only. Impossibility to find the number plate or low probability of the recognition is considered as an error as well. At the same time, not all exceptions are regarded as errors. If the error is detected by Error Handling agent, it requests the HFT controller for a suitable error recovery action, which could vary depending on current system operation.

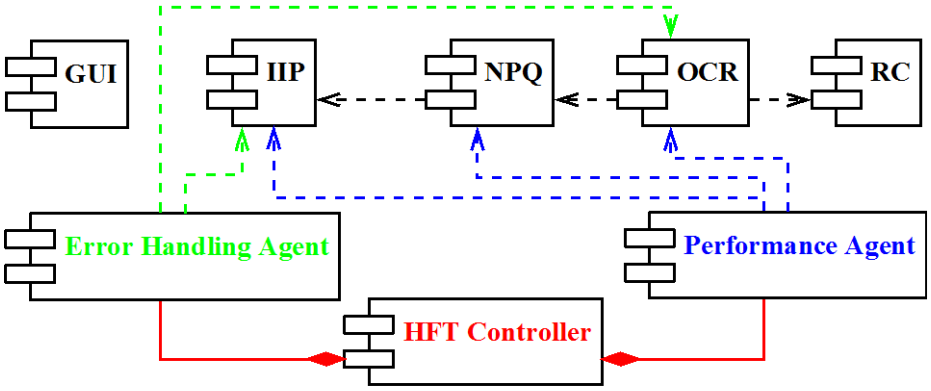

Fig. 3. UML diagram of the use case application

4.2 Characterisation of the components

For the characterisation, we have chosen the IIP and the OCR components, since they are the most critical components of the application. Characterisation data is presented in Table 1 and Table 2. The input data varies significantly for the given application, hence we have chosen three groups of images distinguished by size: small, medium and large. Time and reliability of the image processing significantly depends on the image size.

Table 1. Characterisation of the IIP component

Original image size		Number plate detection algorithm			
		Rectangle detection		HAAR cascade	
		Average time	Average reliability	Average time	Average reliability
Small	< 200 KB	20 ms	85%	9.3 ms	77%
Medium	200 KB – 1MB	85 ms	80%	76 ms	85%
Large	1 MB – 7 MB	143 ms	72%	328 ms	86%

Table 2. Characterisation of the OCR component

Original image size		Optical Character Recognition algorithm			
		OpenCV implementation		Tesseract	
		Average time	Average reliability	Average time	Average reliability
Small	< 200 KB	23 ms	70%	33 ms	75%
Medium	200 KB – 1MB	29 ms	73%	37 ms	78%
Large	1 MB – 7 MB	45 ms	48%	50 ms	62%

4.3 SAN modelling and simulations of the system

With this characterisation data, we can build the SAN models in Möbius. A detailed SAN model for the two components IIP and OCR, each in three versions small, medium and large is shown in Fig. 4. The fundamental states for each component version are working and idle. Working means that this component version is in execution and idle means that it is not in execution. The model is simplified to put all idles together. This means that for, e.g. IIP, the IIP_idle place is initialised with the with the number of threads given to the IIP component. This may be known as the IIP capacity of the system. Each completion of an IIP component version puts a token back to this idle place. Each IIP component version has a probability of success P_s and a probability of failure 1-P_s and this is represented by the stochastic timed transitions IIP_finish. The OCR component models have the same structure. Between the IIP and OCR blocks, three queues are modelled with the standard SAN representation for queues or buffers. The IIP_start transition on the left generates input images stochastically according to probability functions and rates that can be set in the model.

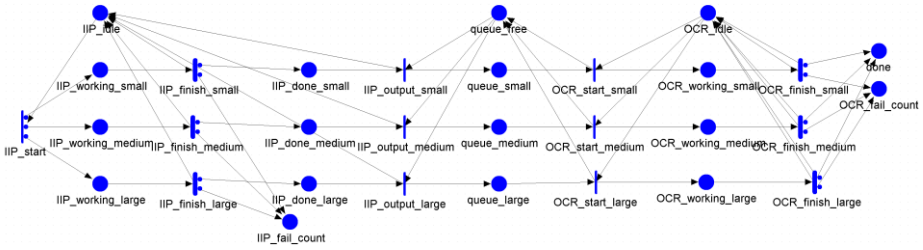

Fig. 4. Detailed SANs model of the use case application in Möbius

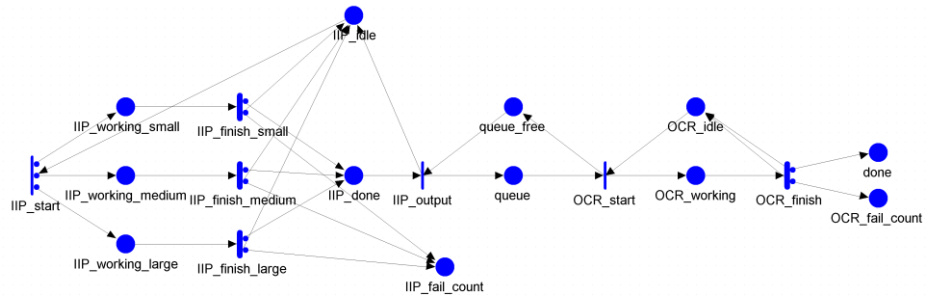

Fig. 5. Reduced SANs model of the use case application in Möbius

The occurrences of failure are tracked by the markings of the failure places and the overall number of successful recognitions is recorded in the final done place at the right end of the net. Running simulations with this model produces success and failure rates, resource utilisation (e.g. the average number of threads being active) and overall performance.

This model turns out to require somewhat significant time (more than a few minutes) to simulate. As a result, by making OG analysis and studying the characterisation data, we decided to derive a reduced model, which is shown in Fig. 5.

The reduced model only has a single OCR component version by combining the three different versions in the detailed model into one using the average behaviour. The reason behind this is that the version pertaining to large size is significantly slower than the others, which are very similar. Intuitively, component capacities are used more on the faster processing versions as they tend to grab the token from the idle place more frequently.

The reduced model required simulation times that are an order of magnitude shorter than the detailed model, and they produced very close results with differences within 5% on all the extra-functional properties being studied. Some simulation results are shown in Table 3.

Table 3. Simulation results

Configuration				Estimates		
IIP algorithm	IIP threads	OCR algorithm	OCR threads	Core allocation	Success rate	Image time
Rectangle	4	OpenCV	4	4.55	0.543	44.89
Rectangle	2	OpenCV	6	2.40	0.550	55.91
Rectangle	6	OpenCV	2	6.63	0.548	40.77
Rectangle	1	OpenCV	7	1.24	0.528	87.11
Rectangle	7	OpenCV	1	7.49	0.559	40.68
Rectangle	1	OpenCV	1	1.22	0.529	88.26
Rectangle	3	OpenCV	1	3.40	0.553	48.86
HAAR	4	Tesseract	4	4.53	0.574	90.54
HAAR	2	Tesseract	6	2.36	0.577	116.89
HAAR	6	Tesseract	2	6.57	0.568	77.46

Rectangle	4	Tesseract	4	5.42	0.561	46.23
Rectangle	2	Tesseract	6	2.86	0.564	56.53
Rectangle	6	Tesseract	2	7.27	0.551	41.91
HAAR	4	OpenCV	4	4.26	0.561	90.01
HAAR	2	OpenCV	6	2.20	0.589	117.79
HAAR	6	OpenCV	2	6.31	0.557	76.98

In these particular simulations, we wanted to find out if the relative numbers of IIP and OCR components executed affect the execution time, resource utilisation and reliability. It was found that the reliability stays about the same, but running more IIP components than OCR components improved the overall execution time and resource utilisation (more components get executed simultaneously, pressing more cores and reducing idle time and queue length).

In case if the observed change in reliability is considered insignificant, the reduction of control loops leads to removal of all knobs except the number of IIP threads. This remaining knob provides the control over resource utilisation and performance. On the other hand, if the reliability difference is considered significant, all knobs contribute to controlling the system properties.

4.4 Hierarchical model of the system

A hierarchical model of the system is built following the general template (Fig. 2) and is shown in Fig. 6.

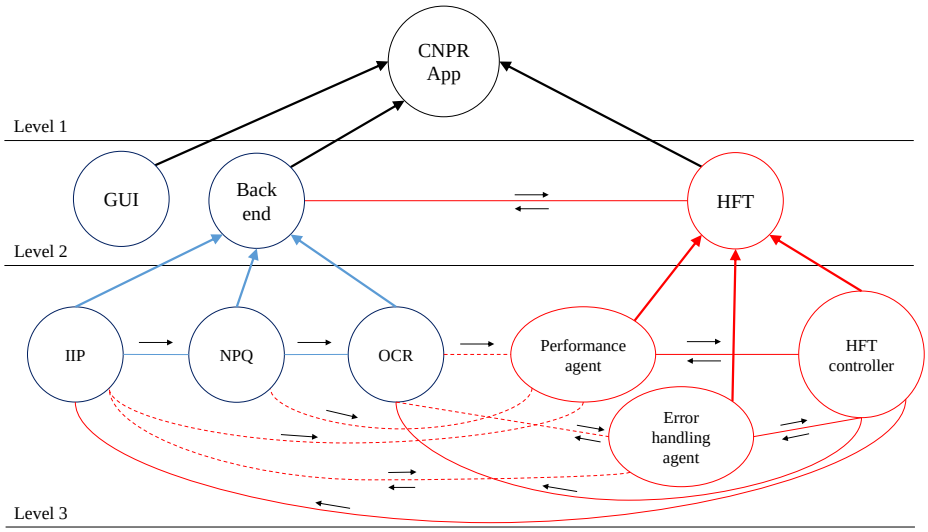

Fig. 6. Hierarchical model of the system

At Level 1 of the system Order Graph there is only one node “Car Number Plate Recognition Application”. Level 2 distinguishes between the HFT part of the system

and functional part, which is comprised of the GUI and system backend. At Level 3 all crucial components of the system and the HFT part are illustrated. We do not model GUI behaviour, therefore we stop at Level 2 for GUI. We have chosen to decompose the backend to IIP, NPQ and OCR components because it follows the UML structure of the application. The HFT part is decomposed to the HFT controller, Performance Agent and Error Handling Agent. At Level 4 there is further decomposition to the inner structure of the functional components and the HFT elements. Level 4 is not illustrated here due to the number of elements at this level.

There is uni-directional information flow from the IIP, the NPQ and the OCR components to the Performance agent, because this agent only monitors the components, but it does not affect the control flow of the components. In contrast, the Error Handling agent has bi-directional information flow, since it intervenes in the control flow of IIP and OCR components in order to handle the errors. The interfaces between the agents and components are represented by dashed lines because they are implicit for the components. The HFT controller, in turn, utilises public interfaces of the IIP and OCR components to reconfigure the components and performs fault handling of the application. In addition, there are information flows between the HFT controller and the HFT agents. It can be seen that all control loops mentioned in Section 4.4 exist in this Order Graph, which validates the correctness of the selected HFT architecture.

5 Conclusion

In this study, we elaborated the general method for modelling computer systems with the HFT at the early stages of the system design. The given method simplifies the modelling process and allows the developer to adjust the system at the early stage to achieve efficient operation after the implementation.

As a part of the workflow, we build the SANs model of the system using Möbius tool. After that we obtain the list of interfaces for the HFT control representing the control loops between system components and extra-functional properties of the system. At the same time, we create a hierarchical model of the system with the HFT using Order Graphs. The method has been demonstrated with a use case application of UK number plate recognition.

Currently we are working on the evaluation of the efficiency of the HFT architecture in terms of performance and resource utilisation. The evaluation is based on the comparison of these properties in two functionally identical systems. One system is implemented with the HFT architecture and another system uses the standard approach to fault tolerance. The presented modelling method is expected to significantly assist in adjusting the system to prepare it for the evaluation.

As a future work we are planning to ensure scalability of the HFT approach, and show that the HFT architecture can be applied for large-scale systems. We propose to introduce the idea of adaptive holistic fault tolerance, that will be able to control the HFT agents in run-time depending on the current system state. To do this the present-

ed modelling approach should be extended to include the reconfiguration of the HFT architecture elements.

References

1. Rafiev, A., Xia, F., Iliasov, A., Gensh, R., Aalsaud, A., Romanovsky, A., Yakovlev, A.: Selective abstraction and stochastic methods for scalable power modelling of heterogeneous systems. In: 2016 Forum on Specification and Design Languages (FDL), pp. 1-7. (2016)
2. Sanders, W.H., Meyer, J.F.: Stochastic activity networks: formal definitions and concepts. In: Brinksma, E., Hermanns, H., Katoen, J.-P. (eds.) Lectures on formal methods and performance analysis, pp. 315-343. Springer-Verlag New York, Inc. (2002)
3. Gensh, R., Romanovsky, A., Yakovlev, A.: On structuring holistic fault tolerance. Proceedings of the 15th International Conference on Modularity (MODULARITY 2016). ACM, Málaga, Spain (2016)
4. Gensh, R., Rafiev, A., Garcia, A., Xia, F., Romanovsky, A., Yakovlev, A.: Architecting Holistic Fault Tolerance. In: 2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE), pp. 5-8. (2017)
5. Gensh, R., Garcia, A., Romanovsky, A.: Experience Report: Evaluation of Holistic Fault Tolerance. School of Computing Science Technical Report Series. School of Computing Science, Newcastle University (2017)
6. Rafiev, A., Xia, F., Iliasov, A., Gensh, R., Aalsaud, A., Romanovsky, A., Yakovlev, A.: Order Graphs and Cross-Layer Parametric Significance-Driven Modelling. In: 2015 15th International Conference on Application of Concurrency to System Design, pp. 110-119. (2015)
7. Peterson, J.L.: Petri Net Theory and the Modeling of Systems. Prentice Hall PTR (1981)
8. The Möbius modelling tool. https://www.mobius.illinois.edu.
9. Zurcher, F.W., Randell, B.: Iterative Multi-Level Modeling - A Methodology for Computer System Design. In: Proceedings IFIP Congress 68, pp. 138-142. Press, (1968)
10. Ehrenfeucht, A., Rozenberg, G.: Zoom structures and reaction systems yield exploration systems. International Journal of Foundations of Computer Science. 25, 275-305 (2014).
11. Laddad, R.: AspectJ in Action: Practical Aspect-Oriented Programming. Greenwich, CT, USA: Manning Publications Co. (2003)
12. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
13. Smith, R.: An Overview of the Tesseract OCR Engine. In: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), pp. 629-633. (2007)
14. Baggio, D.L., Emami, S., Escrivá, D.M., Ievgen, K., Mahmood, N., Saragih, J., Shilkrot, R.: Mastering OpenCV with Practical Computer Vision Projects. Birmingham: Packt Publishing Ltd (2012)
15. Avizienis, A., Laprie, J.-C., Randell, B., Landwehr, C.: Basic Concepts and Taxonomy of Dependable and Secure Computing. Transactions on Dependable and Secure Computing. 1, 11-33 (2004)
16. Bubnicki, Z.: Modern Control Theory. Springer-Verlag Berlin Heidelberg (2005)
17. Metropolis, N., Ulam, S.: The Monte Carlo Method. Journal of the American Statistical Association (1949)