Dual Graph Error Propagation Model for Mechatronic System Analysis

Andrey Morozov, Klaus Janschek

Institute of Automation, Faculty of Electrical and Computer Engineering, Technische Universität Dresden, 01187 Germany
(e-mails: andrew.morozov@gmail.com, klaus.janschek@tu-dresden.de)

Abstract: Error propagation analysis is an important part of a system development process. This paper addresses a model based analysis of spreading of data errors through mechatronic systems. Error propagation models for such kind of systems must use an abstraction level, which allows the proper mapping of the mutual interaction of heterogeneous system elements such as software, hardware and physical parts. A number of appropriate approaches have been introduced in recent years. The majority of them are based only on a data flow analysis. It is shown in this paper that for a complete picture the system control flow has to be considered as well. A new approach based on probabilistic control flow and data flow graphs is presented. The structures of the graphs can be derived systematically from an UML/SysML model of a system. The knowledge about an operational system profile allows the definition of additional system properties. Initially this model was developed for software errors localization. This paper shows its applicability to the error propagation analysis of an entire mechatronic system. The paper presents the modeling concept, the complete mapping process and application of the model for error localization. A reference robot control example demonstrates the main modeling steps.

Keywords: Control system analysis; Data flow analysis; Error analysis; Markov models.

1. INTRODUCTION

Mechatronic systems incorporate the assembly of heterogeneous physical elements (mechanical, electrical, computer and information technology) with various mutual physical interactions (see Fig. 1). The goal of mechatronic system design is to ensure a proper and coordinated operation of these elements within a feedback structure under all possible operational conditions. One of the big challenges of mechatronic systems design is the use of appropriate models, which describe the mutual physical interaction on common abstract levels. This only allows a sound assessment and estimation of expected real system behavior and performances based on model data. In particular during early development stages, the system models are the only source to generate sound behavior and performance data.

A widely accepted candidate for heterogeneous system modeling is the UML language (see OMG [2010a]). Although it has been designed originally for software systems, it is equally applicable for the modeling of heterogeneous systems like mechatronic systems on a certain abstraction level, e.g. the UML extension for systems engineering - SysML (see OMG [2010b]). The core aspects of a system - structure, behavior and interaction - can be described with UML diagrams and its extensions in a formal and transparent manner, thus forming the baseline system model. This information is helpful in particular for certain system analysis tasks, which are typically performed in early phases of the development life cycle: performance analysis, reliability and safety assessment, evaluation of security, usability, testability and other non-functional properties (Fig. 2).

The benefits of early system analysis are well known and quite obvious: the sooner a system error or deficiency is found, the less effort it takes to eliminate it. Usually each type of system analysis requires only a part of baseline system model information and as a rule, those analysis models should be structured in an individual way.

Starting from a baseline system UML-model some appropriate intermediate models are created and used either for simulation or analytical evaluation. This paper addresses an error propagation analysis. This is a very important aspect within a system development process (Fig. 2). Information about error behavior is extremely helpful for the following tasks:

Fig. 1. Mechatronic System Overview
Fig. 2. Early System Analysis

- Reliability assessment. The majority of architecture based reliability models do not take into account propagation of errors. An extension of these models with the error propagation aspect will improve a precision of reliability evaluation (see Popic [2005]).
- Test case generation. Approaches of automatic model-based test case generation can be augmented with the error propagation analysis or even new methods can be developed using these behavior properties.
- Error localization. As a rule, error localization or error isolation is the reverse of error propagation analysis, which cannot be performed without an application of an error propagation model.

2. STATE OF THE ART AND MOTIVATION

Error propagation analysis (EPA) is closely connected with reliability issues. Existing reliability models often underlay error propagation models (EPM) and vice versa EPA is used for more accurate reliability evaluation. There are fundamental differences in the EPA and EPM approaches for hardware (HW) and software (SW) systems, because of divergent causes of the errors. Mechatronic systems consist of a mix of HW and SW components. In this case an analyst has three options: (i) to use two different models for SW and HW, (ii) to extend a HW model for an overall system analysis, or (iii) to extend a SW model for the same purpose. The option (i) is used for systems with clearly defined error propagation interfaces between the HW and SW parts. The second one (ii) is suitable for systems with simple software structure. The third one (iii) fits for the case of a complex SW structure in comparison with the HW one.

Usually EPA of the HW is based on one of the classical reliability evaluation techniques: Failure Modes and Effect Analyses, Hazard and Operability studies, Fault Trees Analysis, Event Trees.

Software EPMs usually have a higher level of abstraction and can be extended easily for analysis of embedded systems. The majority of them were also grown from architecture-based reliability models, a good survey is given in Goseva-Popstojanova and Trivedi [2001]. The most popular error propagation models have been introduced in Nassar et al. [2002] and Hiller et al. [2001]. Both models consider a system abstraction of independent elements linked with a set of error propagation connectors. Nassar’s model operates with probabilities of connector activation and probabilities of error propagation through the connectors. In Hiller’s model more attention is paid for the probability of error propagation through the element, from the specified input to the specified output. These two approaches and a number of their extensions are based on a single probabilistic data flow graph, which is used for numerical analysis of error propagation through the system. These models are working well for systems with a straightforward order of elements execution, which is proved with real systems evaluations. But they do not take into account more complex situations. For instance when some elements are executed in a cycle; or one element executes more frequently than another because of a conditional control flow fork; or the situation when an error will propagate only if a number of specified elements will be triggered in specified order. For this reason we think that a single graph model is not enough and that EPM should be based on two aspects: control flow analysis and data flow analysis.

This paper describes a concept of a new error propagation model based on probabilistic control flow and data flow graphs. Initially this EPM was developed only for a software error propagation analysis. It was presented as a part of an approach to software error localization in Morozov and Jansche [2010]. The current paper shows the extension of the EPM to mechatronic systems.

3. ERROR PROPAGATION MODEL

The Error Propagation Model (EPM) considers a system as a set of independent elements.

Element represents an executable part of the system. For example it can be a software method, which is executed during system operation or a sensor that performs measurement. Faults can be activated and propagate as errors to the other elements during the execution of the element.

Fault activation is defined as malfunctioning execution of an element, which leads to incorrect data output. The fault activation probability depends on the reliability of the element and is considered as a priori known.

Error refers to a corruption state of the data, which is transferred from one element to another.

Error propagation through a system comprises two aspects: error propagation between the elements and error propagation through the elements. The first is defined as a transfer of erroneous data from one element to another. The second means that erroneous value on the input becomes the reason of corrupted value on the element's output.

Core model abstractions of the EPM are a control flow graph (CFG), a data flow graph (DFG) and an observation model.

3.1 Control Flow Graph

A control flow graph is a mathematical and visual representation of a control flow of the system.

\[ G^{CF} := [E, A^{CF}] - CFG, digraph. \]

\[ E := \{e_1, \cdots, e_N\} - set \ of \ CFG \ nodes \]

\[ A^{CF} := \{\{e_i, e_j\}, \cdots, \{e_k, e_l\}\} - set \ of \ CFG \ arcs \]
As was already mentioned, we consider a system as a set of executable elements. Control flow graph nodes represent the elements, arcs - control flow transitions.

An arc from element $e_i$ to element $e_j$ means that element $e_j$ will be executed immediately after execution of element $e_i$. An existence of more than one outgoing arc denotes a control flow fork after the element’s execution. Backward arcs describe different types of control flow loops.

Probabilities of transitions are defined as the properties of the arcs and written in a transition probability matrix (TPM) - $P^{\text{trans}}$. All values of the TPM’s elements are within the limits of the interval $[0,1]$, as well as a sum of the values of each row. From this point of view a CFG is equivalent to a Discrete Time Markov Chain (DTMC).

The mathematical framework of Markov Chains gives ample opportunities for a numerical control flow analysis. First of all it shows how frequently one or another element is triggered during the execution of an entire system. This affects strongly the probability of fault activation. The Markov representation of a CFG allows calculating conditional probabilities of a sequential execution of certain elements of the system. A superposition of these probabilities on DFG paths forms the basis of the error propagation model. The example shown in Fig. 3 (left side) represents a CFG of a system of nine executable elements with defined probabilities of control flow transitions, which are marked as arcs weights.

3.2 Data Flow Graph

A data flow graph is a logical and visual representation of a system’s data flow.

$$G^{DF} := [E, A^{DF}] - \text{DFG, digraph.}$$

$$E := \{e_1, \cdots, e_{N_E}\} - \text{set of DFG nodes}$$

$$A^{DF} := \{\{e_i, e_j\}, \cdots, \{e_k, e_l\}\} - \text{set of DFG arcs}$$

The nodes of the DFG are the same as the nodes of the CFG. They also represent the elements of the system. The arcs of a data flow graph show the existence of data transfer between them. For example, an arc from element $e_i$ to element $e_j$ denotes that output data of $e_i$ goes as input to $e_j$, see Fig. 3 (right side).

DFG arcs are also seen as paths of error propagation. In this paper we simply consider that erroneous data, which come to the element, will spread to all its DFG descendants. But like in the Hiller’s model, the error permeability factor between the inputs and outputs of the DFG node can be defined.

3.3 Observation Model

The process of error propagation through the system can be described simultaneously on both control flow and data flow graphs. The highlighted path through the CFG on the Fig. 3 determines a set of elements, which were triggered during the system execution. By the mapping of this set to the DFG we can find out which of data flow arcs were activated.

Some of the DFG-elements we call observers. They are marked with a loupe-symbol in the data flow graph (see Fig. 3 right side). A "minus" within the loupe means that the observer detects an error, "plus" means normal behavior.

Observers represent the parts of the system, which are equipped with error detection mechanisms. At EPM level the observer is an element, which reports about either fault activation in the element or error propagation through it. For hardware elements, it can be watchdogs or components which actions are clearly observable by an operator. For software subsystems it can be elements that perform data checking, failure detection algorithms, output to log-files or to the user interface.

Observation results give two kinds of information about the system execution. First, it shows which of observers and in what order they were triggered. Second, it shows where errors were detected. The example on the Fig. 3 shows that in the case of fault activation in element $e_2$ and its further error propagation to element $e_7$, we will obtain the following observation result: $[e_1^+, e_2^+, e_3^+, e_7^-]$.

It means that element $e_1$ and element $e_3$ committed normal behaviour, whereas element $e_7$ detected an error and element $e_2$ was not executed at all.

3.4 Model Assumptions and Application Restrictions

The proposed EPM model gives wide possibilities for a probabilistic error propagation analysis, but the application of the model is restricted by a number of assumptions, which are defined in this section.

First assumption: The control flow satisfies the Markov property and the probabilities of control flow transitions are a priori known.

Second assumption: Observers always detect the error by appropriate means.

Third assumption: The activation of faults in two different elements is treated as independent events and the probability of fault activation in each element during its execution is known.

Fourth assumption: The error propagates through the element from each data input to each output.
We assume that the introduced model can be applied for error propagation analyzes of mechatronic systems with discrete time behavior, whose elements are executed one by one according to the defined control flow.

Control flow branching exists mainly because of the software components. That is the reason why the system under consideration should be controlled by software with rather complex control flow. Otherwise the application of this model becomes not so reasonable.

4. ERROR PROPAGATION ANALYSIS FOR MECHATRONIC SYSTEMS

This chapter shows how the abstract error propagation model can be enhanced and applied for a typical mechatronic system, consisting of mutually coupled equipment hardware and software elements as well as physical elements. The sample mechatronic system in question is a caterpillar mobile robot as shown in Fig. 4.

The task of the control system is to move the robot from a current position to a target position on an uneven surface. Roughness of the surface gives physical back effects, which require repetitive correction of the robot course during the mission. For sake of simplicity and transparency of the proposed modeling approach, the following high level abstraction of the mobile robot path planning and motion control system is assumed (an extension to more detailed and thus more realistic models is obvious and straightforward).

We consider a discrete time control system architecture with a certain control cycle time. At each control cycle the controller receives data from sensors, computes and checks the current robot pose (position and orientation) and calculates an appropriate speed for servo motors to correct the direction of robot motion towards the target location. This is repeated until the robot reaches the anticipated target location.

The abstraction of the caterpillar mobile robot system as a control system block diagram is shown in Fig. 5. The controlled variable is the robot pose (position and orientation). The controller is represented by a computer with embedded control software. Two servo motors connected with the caterpillars serve as actuators. A compass, which is placed on the robot, and a wireless navigation camera for position determination are representing the sensors of the control system.

Within a formal system design process a more general system model of the caterpillar mobile robot in terms of an UML/SysML model serves as the baseline design reference. An extended activity UML diagram of this sample mechatronic system "caterpillar mobile robot" is shown in Fig. 6. The idea of the extension is to take into account also non-software activities, in particular hardware control elements and the physical behavior of the robot.

The activities of hardware elements are the following: 'take_picture' for the navigation camera, 'compass_measure' for the compass sensor and 'activate_motors' for the servo motors. They are marked with a dashed border in the diagrams (see Fig. 6). The physical behavior of the robot (dynamics and kinematics) is represented by the activity 'real_movement' with 'robot_behavior' as output variable. It should be noted that the consideration
Fig. 7. Control Flow and Data Flow Graphs of Reference Mobile Robot System

and integration of the physical behavior introduces an additional "physical" feedback path in the activity diagram (see Fig. 6), which normally is not found in software oriented UML models.

For modeling of the error propagation behavior we need some additional information of the operational profile of the system. This can be derived from previous (historic) experiences or from the system requirements. We are assuming here that the robot is reaching the target position on average with nine control cycle steps, i.e. the target position is reached on average every tenth control cycle step. We further assume that on average every 2-nd control cycle step the robot's course has to be corrected (i.e. activation of 'calc_speed', see Fig. 6). This data will help defining the control flow transition probabilities for error propagation analysis, as will be shown later.

The Error Propagation Model in terms of Control Flow Graph and Data Flow Graph are shown in Fig. 7, they can be derived from the UML Activity Diagram (see Fig. 6) in a straightforward and automated way. The probability weights in the CFG arcs can be easily computed using the statistical operational information as defined above.

Now we have to define DFG observers. Let us assume that our system analysis task is devoted to the definition of system test procedures. Thus we are defining an observer on element 'RM' in the DFG (see Fig. 7) for monitoring of the real physical motion of the robot (a monitoring system or a test engineer).

The existence of cycles in the CFG and DFG leads to an infinite number of the paths and complicates the error propagation analysis considerably, Morozov and Janschek [2010]. This is the main reason, why in a general case (and all real-world systems) simple path-based approaches are not applicable. As a consequence more complex, state-based approaches, like the one proposed in this paper, must be used.

This chapter addresses one possible application of the EPM, the task of Error Source Localization (ESL). As a reference example we will take the caterpillar mobile robot control system from the previous chapter (see Fig. 4). We assume that the probability of fault activation in each element is known and equals to 0.001. Let us consider the situation when the observer in 'RM' detects an error and the ESL should be performed.

The process of fault activation and error propagation during the system execution can be described with a DTMC. This DTMC is generated using the structures of CFG and DFG, the observation model and the parameters of the EPM, i.e., probabilities of control flow transitions and fault activations. Fig. 8 shows a part of this DTMC, which describes the behavior of the elements CD, CS and MC (highlighted in Fig. 7).

Each state of the DTMC represents the state of the system via the following parameters: set of the elements where faults were already activated (FA), set of the elements 'infected' by the errors because of a fault activation or an error propagation (EP) and the element which will be triggered next (EXC). For instance an initial state {FA:[] EP:[] EXC-CD} (see Fig. 8) represents the following situation: element 'CD' will be executed next and there was neither fault activation nor error propagation before. The CFG shows that one of the elements 'CS' or 'MC' has to be triggered after that with a probability equal to 0.5. Also according to the assumption with a probability of 0.001 a fault will be activated in 'CD'.

This information defines four different outcomes which are represented with four different states in the DTMC marked with 'STOP' represent various results of the system’s execution. These elements are absorption states of the DTMC.

Fig. 8. A part of a DTMC for Error Source Localization

5. APPLICATION OF THE MODEL FOR ERROR SOURCE LOCALIZATION

This chapter addresses one possible application of the EPM, the task of Error Source Localization (ESL). As a reference example we will take the caterpillar mobile robot control system from the previous chapter (see Fig. 4). We assume that the probability of fault activation in each element is known and equals to 0.001. Let us consider the situation when the observer in 'RM' detects an error and the ESL should be performed.

The process of fault activation and error propagation during the system execution can be described with a DTMC. This DTMC is generated using the structures of CFG and DFG, the observation model and the parameters of the EPM, i.e., probabilities of control flow transitions and fault activations. Fig. 8 shows a part of this DTMC, which describes the behavior of the elements CD, CS and MC (highlighted in Fig. 7).

Each state of the DTMC represents the state of the system via the following parameters: set of the elements where faults were already activated (FA), set of the elements 'infected' by the errors because of a fault activation or an error propagation (EP) and the element which will be triggered next (EXC). For instance an initial state {FA:[] EP:[] EXC-CD} (see Fig. 8) represents the following situation: element 'CD' will be executed next and there was neither fault activation nor error propagation before. The CFG shows that one of the elements 'CS' or 'MC' has to be triggered after that with a probability equal to 0.5. Also according to the assumption with a probability of 0.001 a fault will be activated in 'CD'.

This information defines four different outcomes which are represented with four different states in the DTMC marked with 'STOP' represent various results of the system’s execution. These elements are absorption states of the DTMC.
Using the mathematical framework of Markov Chains we are able calculating probabilities of absorption in each of them. These probabilities are also shown in Fig. 8. For instance the probability of absorption in state (FA:] EP::] STOP) is the highest one and equals to 0.9975. This state represents the fault free execution of the system. The probability of absorption in state (FA:CD,CS,MC EP::[CD,CS,MC STOP) is the lowest one and equals to 5.0e − 9. This state represents the execution of the system with fault activation in all three elements.

The DTMC for the entire Robot Control Example was generated in the same manner. The states, which contain 'RM' in the EP set, are marked as 'FAIL-STOP' states and have no further transitions. These states represent proper error detection by the observer. Parameters of 'FAIL-STOP' states and probabilities of absorption are presented in Table 1. The probabilities are conditional - each of them shows the probability of absorption under the condition that the process was actually stopped after error detection by the observer in 'RM'. In other words Table 1 demonstrates all possible system states after error detection, sorted by their probabilities. The first column of Table 1 shows the sets of the elements where faults were activated - desired error sources.

Table 1. Probabilities of Error Location

<table>
<thead>
<tr>
<th>FA in:</th>
<th>EP to element 'RM' trough:</th>
<th>Probability:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RM</td>
<td>RM</td>
<td>0.1777450</td>
</tr>
<tr>
<td>AM</td>
<td>AM, RM</td>
<td>0.1777450</td>
</tr>
<tr>
<td>MC</td>
<td>MC, AM, RM</td>
<td>0.1777450</td>
</tr>
<tr>
<td>CS</td>
<td>CS, MC, AM, RM</td>
<td>0.0888280</td>
</tr>
<tr>
<td>CD</td>
<td>CD, CS, MC, AM, RM</td>
<td>0.0888280</td>
</tr>
<tr>
<td>CM</td>
<td>CM, CD, CS, MC, AM, RM</td>
<td>0.0888280</td>
</tr>
<tr>
<td>CP</td>
<td>CP, CD, CS, MC, AM, RM</td>
<td>0.0888280</td>
</tr>
<tr>
<td>TP</td>
<td>TP, CP, CD, CS, MC, AM, RM</td>
<td>0.0888280</td>
</tr>
<tr>
<td>I</td>
<td>I, CP, CD, MC, AM, RM</td>
<td>0.0093213</td>
</tr>
<tr>
<td>I</td>
<td>I, CP, CD, CS, MC, AM, RM</td>
<td>0.0093213</td>
</tr>
<tr>
<td>MC, AM</td>
<td>MC, AM, RM</td>
<td>0.0001779</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>all elements</td>
<td>all elements</td>
<td>9.4e − 27</td>
</tr>
</tbody>
</table>

Because of equal probabilities of fault activation and the assumption that the error propagates through each element, the distribution of probabilities in Table 1 depends only on the structures of CFG and DFG. We can see from the graphs in Fig. 7 that an error existing in element 'CD' propagates to element 'MC' and further only in the case of execution of element 'CS', which does not happen every iteration. This fact is also visible from the results in Table 1. It is approximately two times more probable that the fault was activated in elements RM, AM or MC than in CS, CD, CM, CP or TP.

In our simple example there is only one cycle, which represents the control feedback loop. The element 'I' is out of this cycle and executed only one time, which results in an approximately nine times smaller number of executions than the other elements. As a consequence, the conditional probability of fault activation in the 'I' is smaller as well.

This brief and non-exhaustive discussion of the proposed EPM should have given evidence, why it is mandatory analyzing both graphs simultaneously to get a realistic picture of error propagation in an embedded system.

6. CONCLUSION

The concept of a new probabilistic error propagation model for embedded hardware/software systems, like mechatronic systems, has been described in this paper. Core parts of the model are interlinked control flow and data flow graphs, which can be derived systematically and automatically from UML system models. A control flow graph describes the system behavior using the Markov Chain mathematical framework. A data flow graph determines the paths of data transfer, which are also seen as paths of error propagation. Definition of observers and simultaneous usage of these two graphs offers a possibility for probabilistic error propagation analysis. The derivation and application of the proposed approach has been demonstrated with a reference robot control example. The UML activity diagram of the control software was extended with activities representing hardware elements and physical behavior of the robot. The extended diagram was transformed into the error propagation model. The probabilities of CFG transitions have been defined separately using knowledge of a priori specified operational profiles. The basic idea of an EPM's application for error source localization has been presented in the last chapter. The current research is focused on the automation of model generation and the application of the EPM not only for error localization but for automatic test case generation and system reliability and performance analysis as well.

ACKNOWLEDGEMENTS

This work has been supported by the Erasmus Mundus External Co-operation Window Programme of the European Union.

REFERENCES


