BPM Based Robust E-business Application Development[*] (practical experience report)

Gy. Csertán[1], A. Pataricza[1], P. Harang[1], O. Dobán[1],
G. Biros[2], A. Dancsecz[2], and F. Friedler[2]

[1] Budapest University of Technology and Economics
Department of Measurement and Instrument Engineering
csertan, pataric, harang, doban@mit.bme.hu
[2] University of Veszprém
Department of Computer Science
biros, dancsecz, friedler@dcs.vein.hu

Abstract. Companies rely more and more on the dependability of their e-business application. E-business systems are, by their nature, heterogeneous, consisting not only of information technology component but also human- and infrastructural resources. Therefore, the assurance of a proper level of dependability has to cover all aspects of the system. In the paper, a Business Process Modeling (BPM) based approach is presented, which uses an extended UML profile to design the business processes to compensate the weaknesses of resources. A fault model describes the typical failure modes of the individual element types. Analysis methodologies well proven in the field of dependable computing are used to assess the dependability of the system and provide a basis for countermeasures against the faults. The paper describes the experiences with a pilot application.

1 Introduction

E-business systems spread more and more. Recently emphasis is shifting towards systems in which a large part of the business flow of a company is implemented in an electronic form. As a result, companies rely increasingly on the Quality of Service (QoS) of their e-business system.

Such an e-business system combines IT components (like computer hardware, application software, intra- and internet), human- (like workers) and infrastructural resources (like production facilities, raw material) into a complete system. Therefore, properties of an e-business system are influenced by all of these components and the QoS of the entire system can only be increased if all of the components are incorporated into a global dependability concept.

[*] This work was supported partly by IBM and partly by the Hungarian Ministry of Education under contract No: IKTA 00173/2000.

Dependability of the IT components is increased mainly by using a high-availability server [7, 9] or a clustering solution. Dependability of the resources is increased by using more reliable resources (a more reliable production machine, a better trained person). These methods are expensive, especially for SMEs. Therefore, the proper organization of the business processes has to compensate the imperfect dependability of single components. In this paper we present a design approach that provides algorithm based fault-tolerance (ABFT) for electronic business processes. It can be used to increase the QoS of e-business systems.

1.1 Objectives of the Approach

The main objective of the research is to ensure the robustness of e-business services in the heterogeneous and unreliable environment of SMEs by using a proper design approach. The main guideline of our work was the deduction to the well-proven principles and solutions from the field of dependable computing.

The design approach starts with requirement analysis. It is based on Business Process Modeling (BPM), a semi-formal, graphical modeling method that describes corporate resources and activities. We use a combination of the ARIS BPM notation and the UML-BPM profile.

The model is augmented in the next step by adding fault information. Fault modeling aims at describing the local faults and fault propagation of resources and activities in order to be able to evaluate global fault effects and make proper countermeasures. A functional, qualitative fault model is suggested in which faults are grouped, e.g. according to their criticality. Fault modeling is done within the very same notation as BPM by exploiting UML's standard extension mechanism, the stereotypes.

Analysis provides the global fault effects that are interpreted by application experts, who can then decide whether to deal with the local fault leading to the unwanted effect and propose solutions to handle the local fault. The model is modified and re-analyised until its robustness is considered sufficient. This design approach corresponds to the IPSD (interactive, process-oriented systems development) method [2].

The fault tolerance measures are very similar to the ones used in dependable computing: timeout, back-checking, error correction coding, TMR, etc. The application expert and the dependability expert select the proper solution. Further work aims at collecting a library of such measures. This seems to be a realistic goal, since the set of types of activities is limited: user input, output to user, message sending, database access, data calculation, etc.

1.2 Case Study

We implemented a pilot application at our partner Balatontourist (a Hungarian tourism company) to demonstrate and check the practical usefulness of the method. The pilot application covers the management and marketing system of trips organized by the travel agency or by external contractual agents. Part of this application will be used in this paper as a case study.

The aim of the pilot application was to gain experience in using the modeling approach and to collect data during the operation of the system about local faults and the funcioning of fault-tolerance measures. The former results are presented in this paper together with the results of FMEA analysis of the application. The latter results could not be collected because there was no time before the main season to implement the necessary data logging and collection modules. It will be implemented in this season and results are awaited in autumn.

The pilot application interacts with basic product or service providers (external agents, private providers, travel agencies); the headquarter of Balatontourist; and the consumers (external agents, travel agencies, tourists). It provides features for different types of users. Each user type has different rights and possibilities:

Unregistered users can only browse among tours and trips.
Registered users can browse tours and trips, make / modify reservations.
Salesperson can manage reservations (e.g. reserve, cancel, modify), view the participant list, and print vouchers.
Tour operators can manage tours and trips (e.g. create, modify, delete), can assign resources to tips, perform cost and profit calculations, and gain on-line information about current trips (e.g. number of tourists, planned income).
Managers can have statistics about trips, utilization of trips, income and expenditure per salesperson, trip, or tour operator.

The application implements a part of the internal business logic of the company. A larger part deals with selling services (trips) to customers and a smaller part deals with the internal support of tour operators in their work of organizing programs and trips. The following main business processes have been identified during the requirement analysis phase (the case study contains only the trip announcement process):

Searching for tours and trips The application displays the list of trips and their details based on the user-specified time period or tour. Registered users can make reservations for one or more of the listed trips.
User registration The user fills in a registration form and the personal data will be stored in the customer database.
Reservation for trips The user or the salesman can make a reservation by providing the trip data and the necessary additional user data (local address, deposit, etc.).
Reservation management Cancellation of a reservation, modification of a reservation.
Modification of personal user data Registered users can modify the personal data (e.g. name, address, phone number) stored about them in the database.
Acknowledgment of a reservation When the user books a tour the salesman acknowledges his/her reservation.
Tour management Creating a tour, uploading pictures, modifying tour details/descriptions and assignment of resources and suppliers.

Trip announcement The tour operator announces a trip for a tour by specifying the date and time of departure and arrival, the prices for adults and child, stopping-places, etc.
Trip management The tour operator can allocate the actual resources (bus, meal, travel guide, etc.), close the registration of a trip, cancel a trip.
Resource management Assignment of resources to tours and particular trips, modifying capacity and price information, and adding supplier data.

2 Modeling paradigm

The modeling paradigm of the development approach should be able to:

– describe internal / external resources and the business processes of the company;
– model the relation between resources and processes;
– express dependability and performance parameters of system components;
– include dependability measures to increase system robustness;
– give a common description platform for both IT and economics specialists.

2.1 ARIS and UML-BPM

There are two natural ways to select such a modeling approach: start from business process design tools [3, 5] and extend them to cover IT systems, or extend IT CASE notations to represent corporate resources and processes.

BPM arose as a pure illustrative visualization method with an informal semantics. Simulation and limited optimization capabilities were added later. It uses a small number of elements to model the functions of a company similar to flow charts. The number of elements becomes too large in complex models, causing troubles in the understanding. ARIS [3] is one of the most popular BPM modeling tool used by business architects. It has a well elaborated, but proprietary formalism and its modeling scope is limited mainly to non-IT related business processes (e.g. accounting, production).

The Unified Modeling Language (UML) is a favorite candidate to serve as an IT CASE language. It can also be extended to cover non-IT apects. However, it defines a far too complex notation in order to communicate with non-IT specialists. Standardized subsets of UML, so called profiles, are used to define a domain specific subset of the notation. One of them was the withdrawn UML-BPM profile. It concentrated mainly on the IT related processes of a company, it was very coarsely defined and it was less expressive than ARIS.

In our work we extended the UML-BPM profile with the elements of ARIS [1]. It was relatively straightforward, since both ARIS and UML-BPM are multi-view notations, i.e. they are able to describe a given problem from many different aspects. For example, a person can be seen as an element of a hierarchy of people. On the other hand, the same person can be seen as a human resource assigned to an activity.

2.2 The composite notation

The notation defines views (sub-models) each of which contain a sub-set of standard UML elements. Please refer to [1] for the detailed, formal description of the notation. The views of the composite notation are (the origin of the view is given in brackets):

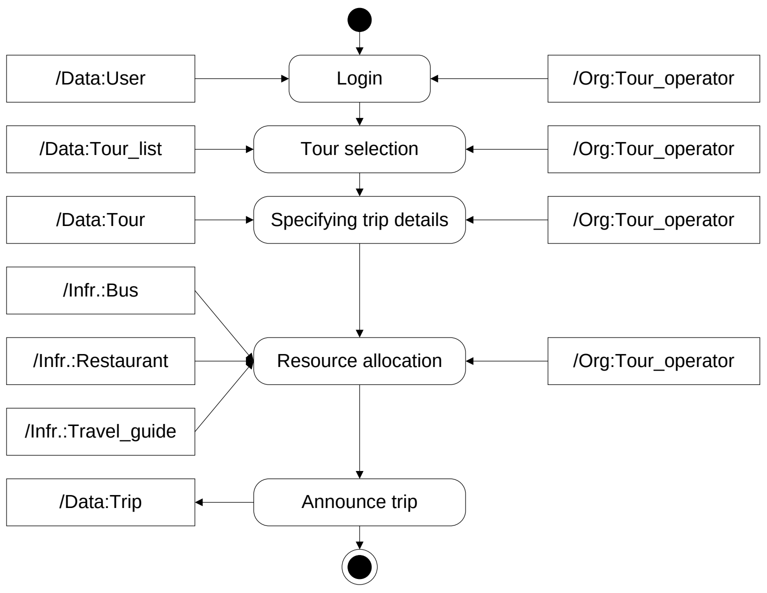

Fig. 1. Control View of the Trip Announcement Process

Process view (UML-BPM) is a use case diagram that defines the business processes of the company, the users of the system and their relation.

Infrastructure view (UML-BPM) is an object diagram that describes the resources and tangible assets of the company and their mutual relation. Their properties are described by object variables.

Organizational view (ARIS) is a class diagram that describes the organization structure of the company; the organization units and their relations to each other, the persons, roles, jobs, and status within organization units.

Function view (ARIS) is a parametrized class diagram that describes the functions of the company in a hierarchical way; a function-tree contains functions (node) and sub-functions (ancestors of a node). Sub-functions cooperate to realize the function.

Data view (ARIS) is a class diagram that defines the data structures used in the implementation of the functions of the company, i.e. to execute the business processes. It corresponds to an extended entity-relationship model. Entities and entity properties are modeled by classes, relationship types by association classes while relationships are described by class associations and generalization. Key properties are described by class qualification. Related data is stored in package clusters.

Control view (ARIS) is an activity diagram that describes the dynamics of the business process, and the resources needed for the execution of a given step.

The pilot application was described by one process view, one infrastructure view, one organizational view, one function view, one data view, and ten control views. The extent of the paper does not allow to include all of these views, only the control view (Fig.1.) of the trip announcement process is included.

The process is started by the login of the tour operator (activity `Login`). The required resources are the user database (data resource `Data:User`) and the tour operator (organizational resource `Org:Tour_operator`). Next a tour is selected for "instantiation" (activity `Tour selection`). Required is the list of tours (data resource `Data:Program_list`). Selection is done by the tour operator. Next the details of the program appear and the tour operator extends them (e.g. by date) in order to create a trip. In the next step (activity `Resource allocation`) the tour operator allocates resources to the trip (infrastructure resource `Infr.:Bus`, `Infr.:Restaurant`, `Infr.:Travel_guide`). Finally the trip is announced (activity `Trip announcement`) by creating a new resource (data resource `Data:Trip`).

3 Dependability Modeling and Evaluation

Although the proposed BPM notation supports the compact modeling of business processes and the communication between software engineer and economist, a formal notation is needed to analyse the system with mathematical precision. In our approach, the model is transformed (using the method [10]) into dataflow networks (DFN) that serves as the language of analysis. This non-deterministic dataflow programming paradigm [6] is very suitable to describe the aspects of faults, their effects and error propagation at the level of functional units of the business process, i.e. activities, resources. The following tasks can be solved this way concurrently with system design:

– fault simulation,
– test generation,
– testability analysis,
– failure modes and effects analysis (FMEA),
– risk analysis.

3.1 Dataflow notation

A DFN is a set of nodes that execute concurrently and exchange data items (tokens) over unbounded, unidirectional, FIFO-like, point-to-point communication channels. Nodes represents the components of the system, channels represent the communication channels and tokens represent the data passed between the components. The graphical representation of a DFN is a dataflow graph (DFG), nodes of which are drawn as boxes and channels of which are drawn as directed arcs.

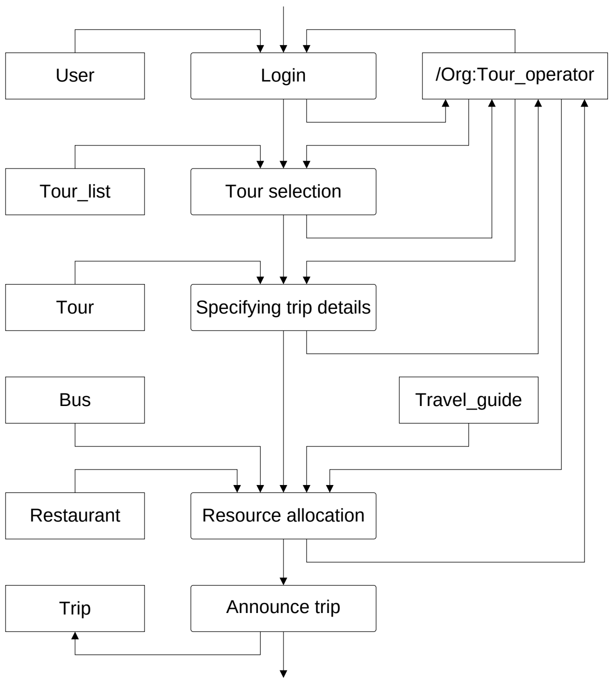

Fig. 2. DFG of the Trip Announcement Process

Definition 1. *A* dataflow network *DFN is a tuple* (N, C, S) *where:*

N *- set of nodes*
C *- set of channels (I-input, O-output, and IN-internal channels)*
S *- set of states; Cartesian product of node and channel states*

Definition 2. *A* dataflow node *n is a tuple* $(I_n, O_n, S_n, s_n^0, R_n, M_n)$ *where:*

I_n *- set of input channels*
O_n *- set of output channels*
S_n *- set of states*
s_n^0 *- initial state,* $s_0 \in S_n$
M_n *- set of tokens*
R_n *- set of firings,* $r_n \in R_n$ *is a tuple* $(s_n, X_{in}, s'_n, X_{out}, \pi)$

s_n, s'_n - *states before and after the execution of the firing,* $s_n, s'_n \in S_n$
X_{in} - *input mapping,* $X_{in} : I_n \mapsto M_n$
X_{out} - *output mapping,* $X_{out} : O_n \mapsto M_n$
π - *priority of the firing,* $\pi \in I\!N$

The meaning of *firing rule* $r_n = (s_n, X_{in}, s'_n, X_{out}, 0)$ is that if node n is in state s_n and $\forall i_n \in I_n$ contains at least the tokens $X_{in}(i_n)$, then r_n can be executed. The execution of r_n removes $X_{in}(i_n)$ tokens from $\forall i_n \in I_n$ and outputs $X_{out}(j_n)$ tokens onto $\forall j_n \in O_n$. After execution the node changes its state from s_n to s'_n.

The dataflow graph of the case study is presented in Fig.2. It describes only the structure of the DFN. It is very similar to the control view, but each resource and activity should appear only once. The description of the announce trip node (n11) is given below as an example for node definition:

node	n11=({trip_details}, {trip_data, finish}, {ok}, ok, {ok}, {r1})
firing	r1=(ok;trip_details=ok;ok;trip_data=ok,finish=ok;0)

The node receives the details of the trip, stores them in the database and finishes the whole trip announcement process. Tokens (ok) describe the fault-free data, since at this step faults are not modeled. The nodes are in fault-free state (ok).

3.2 Fault modeling

The main idea in the fault modeling is the introduction of the notion of faults at the metamodel level of the BPM notation. Roughly speaking, this is the exact counterpart of the notion of stuck-at faults in gate-level logic testing, which associates faults with signals.

In the case of BPM, faults are associated with resources and activities. The first one models primarily the permanent faults, including those, when a resource is missing. The second category is intended to model transient faults, like those in activities carried out by a human operator.

In its current state the approach does not support modeling of design faults and intentional faults carried out by operators, since this may result a distortion of the entire business flow.

3.3 Dataflow-Driven Dependability Analysis

The qualitative analysis is based on the idea of modeling the fault effects and their propagation similarly to the flow of data in the process model [4, 8]. In the case study tokens representing the data are coloured for instance either as *"correct"* or as *"incorrect"* or as *"missing"*. The faulty behaviour of a node manifests in sending such tokens. Accordingly, the fault states of nodes are similarly grouped into *"sends correct data"* or *"sends incorrect data"* or *"does not send data"*. A set of potential error propagation paths can be estimated by tracing the flow of tokens from the fault site towards the outputs. In the model all potential consequences of a fault are incorporated. At the highest level, this abstraction can help to radically restrict the search space for the origins of failures.

Table 1. Local Faults and Fault Effects in the Example

Resource/Activity	Local Fault	Fault Effect
login	does not send data	process hangs (1)
user	does not send data	1
	sends incorrect data	1; wrong access level (0)
tour operator	does not send data	1
	sends incorrect data	0; 1
tour list	does not send data	1
	sends incorrect data	wrong tour selected (2)
tour selection	sends incorrect data	2
tour	does not send data	1
	sends incorrect data	bad trip properties (3)
specifying details	sends incorrect data	3
bus	does not send data	1
	sends incorrect data	3
restaurant	does not send data	1
	sends incorrect data	3
travel guide	does not send data	1
	sends incorrect data	3
resource allocation	sends incorrect data	3
announce trip	does not send data	1
	sends incorrect data	3
trip	does not send data	no data written (4)
	sends incorrect data	3

In subsequent steps of analysis a more refined fault model can be used, resulting in a more faithful description of the fault effects. The potential to use arbitrary user-defined guiding attributes, colorings of the tokens and propagation rules offer full freedom for the analysis of different user requirements. By adding fault occurrence, fault latency and detection probabilities the model can serve as a starting point for a more detailed dependability analysis.

The dataflow graph of the extended model is unchanged (Fig.2), the nodes are extended in order to cover not only fault-free but also faulty behaviour. The description of the announce trip node (n11) is given below as an example for node definition extension:

```
node     n11=({trip_details}, {trip_data, finish}, {ok}, ok, {ok}, {r1,r2,r3})
firings  r1=(ok;trip_details=ok;ok;trip_data=ok,finish=ok;0)
         r2=(ok;trip_details=inc;ok;trip_data=inc,finish=inc;0)
```

r3=(ok;trip_details=dead;ok;trip_data=dead,finish=dead;0)

The extended behaviour contains the fault-free behaviour. We assume that internal faults do not arise in this activity - the node has only fault-free states (ok). If the details of the trip are incorrect (token inc), incorrect data will be stored in the database and the process finishes. If the activity does not receive the details of the trip (token dead), no data is stored in the database and the process does not finish.

4 FMEA Analysis

In our example, the basic potential of the modeling approach is illustrated by the FMEA analysis of the single selected business process. (In case of multiple processes the paradigm is able to handle even inter-process dependencies. An example see later.)

The first step of analysis was to collect possible local faults of the resources and activities, categorize them according to the fault model and assign fault effects to them. The results of this first step are presented in Table 1. At the first time a reference is given to each fault effect (number in brackects). Subsequent occurences are denoted only by this reference number.

Table 2. Global Effects and Protective Measures in the Example

Local Effect	Crit.	Global Effect	Protection
0	1	insufficient authority	
	4	too much authority	
1	2	process dead-lock	process (session) timeout
2	3	trip with wrong data announced	trip check at the end
3	3	trip with wrong data announced	trip check at the end
4	2	no trip announced	data check at the end

The second step of analysis was to propagate the local fault effects in the system. Local faults manifest as erroneous tokens sent by a node. Communication events of the business process are considered atomic in the sense that communication errors have to be explicitly modeled if necessary by additional DF nodes. This propagation is done by fault simulation.

The DFN can be considered as a network of finite-state machines that can be simulated by discrete event simulation. The simulation is done until the tokens reach the output of the DFN, e.g. the local fault effects propagate until they reach the output. In such case, they result in the failure of the business process that is described by the global effects. The global effects for the case study are described in the 3rd column of Table 2. The numbers in the 1st column refer to the local fault effects in Table 1.

Global in the case study is meant for the trip announcement process. In the pilot application all processes are considered, and global is meant for the whole application. In this case the global effect of fault effect No. 3 could be a tourist who does not get what he paid for (displeased) or a trip with deficit.

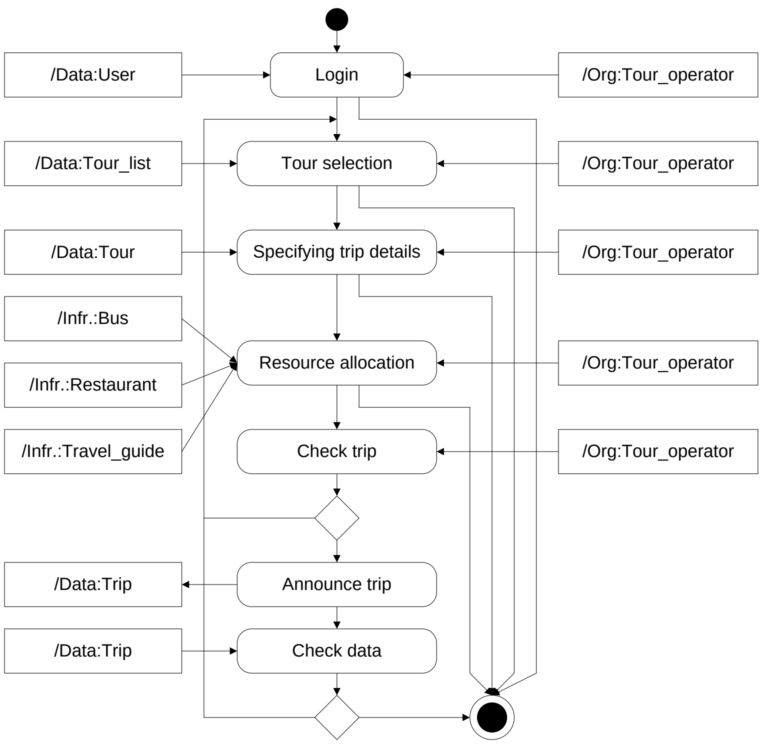

Fig. 3. Control View of the Enhanced Trip Announcement Process

If necessary a criticality factor can be assigned to the global effects. It can be the starting point of criticality analysis. The 2nd column in Table 2 shows a possible grouping of global effects into criticality categories. A trip based on bad data is more critical (3) than not creating the trip at all (factor 2).

The final step of evaluation (not part of FMEA) was to identify fault tolerance measures for the different global effects (failures). It is intended as a solution to improve the dependability of the system. The suggested protection mechanisms are described in the 4th column of Table 2. The interpretation of the 3rd row is the following: if the tour operator does not enter the username and password, the process will wait and hang infinitely. This can be avoided by explicitely finishing the process after a given time if authorization do not occur.

Finally, Fig.3. shows the enhanced trip announcement process that contains the fault tolerance measures listed in Table 2. In the modified process all activities that have an input could also finish the process because of timeout. The details of the trip are presented to the tour operator, who can check them before announcement. Last but not least, there is a check, whether trip data are correctly stored in the database.

5 Conclusions

The experiment (pilot application) showed that modeling paradigms and analysis methods adopted from traditional fields of technical dependability can be used in making e-business systems more robust by providing algorithm based fault-tolerance for the faults of resources.

Our experience shows that the applied fault tolerance measures are simple, logical, many times obvious, but only after the analysis identified the causes of failures by identifying model interdependencies. From the analysis it is clear that counter measures should be applied near to the fault place: tipically at the end of activities and processes.

Acknowledgements

Thanks to Prof. Jie Xu for helping to prepare the final version of the paper.

References

1. A. Pataricza (editor). BPM Modeling Paradigm to E-business Applications. Technical report, IKTA 00173/2000, 2001. in Hungarian.
2. W. Aalst and K. Hee. *Workflow Management: Models, Methods, and Systems.* MIT Press, 2002.
3. Broad spectrum of solutions for the e-organization. Whitepaper, www.ids-scheer.com, 2001.
4. Gy. Csertán, A. Pataricza, and E. Selényi. Dependability Analysis in HW-SW codesign. In *Proceedings of the IEEE International Computer Performance and Dependability Symposium, IPDS'95*, pages 316–325, April Erlangen, Germany, 1995.
5. A. Helton, E. Zulayabar, and P.J.A. Soper. *Business Process Reengineering and Beyond.* Number SG24-2590-00 in Redbooks. IBM Corp., 1995.
6. B. Jonsson. A Fully Abstract Trace Model for Dataflow Networks. In *Proceedings of the 16th ACM symposium on POPL*, pages 155–165, Austin, Texas, 1989.
7. E. Marcus and H. Stern. *Blueprints for High Availability.* Wiley, 2000.
8. Y. Papadopoulos and M. Maruhn. Model-Based Synthesis of Fault Trees from Matlab - Simulink Models. In *Proceedings of the International Conference on Dependable Systems and Networks, DSN 2001*, pages 77–82, Sweden, 2001.
9. S. Russel, O.L.B. Gonzalez, B.D. Coutere, and D. Furniss. *High Availability Without Clustering.* Number SG24-6216-00 in Redbooks. IBM Corp., 2001.
10. D. Varró, G. Varró, and A. Pataricza. Designing the Automatic Transformation of Visual Languages. *Science of Computer Programming*, To Appear.