

Dependability for distributed embedded automation systems in dynamic environments

G. Deconinck, R. Lauwereins

K.U.Leuven, Elektrotechniek (ESAT) – ACCA, K. Mercierlaan 94, B-3001 Leuven, Belgium
Geert.Deconinck@esat.kuleuven.ac.be, <http://www.esat.kuleuven.ac.be/acca>

Position paper submitted to the 3rd Information Survivability Workshop, Boston, MA, USA, Oct. 24-26, 2000.

The *framework approach* [DBCD98, DeDL98, DDLB99] provides fault tolerance capabilities to distributed embedded systems by separating the functional behavior from the actions to be executed when an error is detected.

The framework approach was developed because costs are an important driving force for distributed embedded systems, also when fault tolerance is concerned. This calls for open, flexible and configurable solutions that are able to answer in a cost-effective way to a variety of dependability requirements (esp. availability, maintainability and integrity). Such solutions can be based on pre-built and reusable modules, adaptable for a wide range of applications and reusable in different environments, as in the framework approach.

This conceptual framework consists of the following three entities

- A *library of basic tools*: this library provides basic elements for error detection, localization, containment, recovery and fault masking. The tools are software-based implementations of well-known fault tolerance mechanisms, grouped in a library of adaptable, parametric functions. These basic tools can be used on their own, or as co-operating entities attached to a backbone (see below). Examples include watchdogs, voting units, supported for acceptance tests, replicated memory, etc.
- A *control backbone*: this backbone is a distributed application that extracts information about the application's topology, its progress and its status; it maintains this information in a replicated database and it coordinates the fault tolerance actions at run-time via the interpretation of user-defined recovery strategies. The backbone functions as a sort of middleware. It is hierarchically structured to maintain a consistent system view and contains self-testing and self-healing mechanisms.
- A *high-level configuration and recovery language*: this language is used to configure the basic tools and to express recovery strategies. The application developer specifies these configurations and recovery actions via a descriptive language called *ARIEL*. For

configuration purposes, ARIEL is able to set parameters and properties of the basic tools. For expressing recovery strategies - i.e. indicating fault tolerance strategies by detailing localization, containment and recovery actions to be executed when an error is detected - ARIEL allows building queries on the database of the backbone and attaching actions to these queries. These actions allow, e.g., to start, terminate, isolate or inform an entity. Such an entity can be a node, a single task, a group of tasks, etc. As such, it is possible to start a standby task, to reset a node or link, to generate synchronization signals for reconfiguration, etc. Several ARIEL templates support fault tolerance strategies based on standby sparing, recovery blocks, N-modular redundancy, etc.

The innovative aspects of this approach do not come from the implementation of the library of well-known fault tolerance tools, but rather from the combination with the backbone that executes predefined actions as described in ARIEL when an error is detected. Such (possibly distributed) recovery strategies specified in ARIEL let the user separately address the (non-functional) aspects of application recovery from those concerning the (functional) behavior that the application should have in the absence of faults. Furthermore, separating these aspects allows the modification of the recovery strategy with only a limited number of changes in the application code, or vice-versa, resulting in a better maintainability of the application.

Following this framework approach, increasing the dependability of an application implies the configuration and integration of basic fault tolerance tools from the library into the application and writing a *recovery script* in ARIEL, i.e. describing the recovery actions to be executed when an error is detected. This script is translated into a compact code that is executed at run-time by the backbone when an error is detected. It matches well to a number of coarse-grained local and distributed fault tolerance mechanisms.

This framework approach has proven its cost-effectiveness in a number of industrial distributed embedded applications (primary substations for electricity transport, airfield lighting system), and is now being extended to incorporate inter-site distribution aspects. This will be the focus of the DepAuDE project (*Dependability for embedded automation systems in dynamic environments with intra-site and inter-site distribution aspects*) that has been accepted¹ in the Fifth Framework Programme of the European Union – (*Information Society Technologies IST-2000-5.1.4 Large Scale Systems Survivability*) and will run in 2001-2002.

¹ Non-official, yet unconfirmed information when writing this position paper.

The aim of this DepAuDE project is to develop a **methodology** and an **architecture** to ensure **dependability** for non-safety critical, distributed, embedded automation systems with both IP (**inter-site**) and dedicated (**intra-site**) connections.

Example target applications include 1) monitoring and control of energy transport and distribution, 2) remote operation of telecom routers, switches at railroad yards, 3) distributed real-time control applications with different service levels, 4) novel health care applications based on remote networked embedded systems, etc. On a certain location, the intra-site network is dedicated to the distributed embedded system, allowing for real-time control; among the locations, the inter-site network is often IP-based, it can be shared among several applications and may be public; it does not provide real-time features.

Typically, these embedded automation applications prevail in a **dynamic environment** due to physical, design or malicious faults, due to influence from other systems making use of the same inter-site network, or due to the applications themselves. The target systems should remain dependable in this dynamic environment, with ever changing resources. Dynamic behavior of the systems is also required to accommodate for remote diagnosis, maintenance or functionality (software) upgrades.

Intra-site, many of today's approaches towards dependable applications are based on ad hoc hardware-based fault tolerance or on dedicated (expensive) solutions to cope with this type of faults. This often leads to a lack of *reusability* of the fault tolerance functionality, as well as to a lack of *flexibility* to accommodate for changes in the environment. **Inter-site** approaches lack the dependability (esp. security, availability, integrity) required for control of applications.

Industries require an approach (architecture and methodology) that allows

- to reuse solutions for fault tolerance,
- to survive faults from different sources (hardware, software, attacks, etc.) ,
- to accommodate for modifications and changes in the dynamic environment, and
- to maximise the service-level of the application given the available resources.

This project addresses these urgent needs.

From an *operational* viewpoint, it will focus on the inter-site aspects for remote diagnostic, control and interoperability of the applications. Intra-site, the project will focus on the dynamic adaptation of the application to changes in the environment. For these aspects, it will make use of the above-mentioned **framework approach**.

From a *methodological* viewpoint, it will focus on the way to capture and validate dependability requirements, on the way to derive from requirements the structure of the modeling approach, and on the use of modeling to drive the development and the assessment of the proposed solutions. Well-known semiformal (UML) and formal (temporal logic) languages will be used for this. In addition, predictive modeling (based on temporal and stochastic Petri nets) will be used 1) acting as a specification and graphical model of generic fault tolerance solutions and user-specific fault tolerance strategies, 2) and to assess the performance of the proposed approach and to drive the development. This methodology establishes a direct link between the formalized requirements and the operational modeling.

The outline of the resulting architecture (based on the framework approach) will be modular, situated at middleware level, with approaches to maximize the local (intra-site) service-level and with methods to increase the survivability of the services (inter-site). It will consider both computational and communication aspects (dedicated and public IP-based networks). It will also be characterized by a systematic methodology to apply this architecture to the target systems. Configurability and portability of solutions onto different targets will be a driving concept. In this context it will separate the specification of the functional behavior from the fault tolerance strategies for the application, via the framework approach.

Selected references

[DBCD98] G. Deconinck, O. Botti, F. Cassinari, V. De Florio, R. Lauwereins, "Stable Memory in Substation Automation: a Case Study", *Digest of Papers of 28th Annual Int. Symp. on Fault-Tolerant Computing (FTCS-28)* (IEEE Comp. Soc. Press, Los Alamitos, CA), Munich, Germany, Jun. 1998, pp. 452-457.

[DeDL98] V. De Florio, G. Deconinck, R. Lauwereins, "Software Tool Combining Fault Masking with User-Defined Recovery Strategies", *IEE Proc. - Software, Special issue on dependable computing systems* (IEE, London, UK), Vol. 145, No. 6, Dec. 1998, pp. 203-211.

[DDL99] G. Deconinck, V. De Florio, R. Lauwereins, R. Belmans, "A Software Library, a Control Backbone and User-Specified Recovery Strategies to Enhance the Dependability of Embedded Systems", *Proc. 25th EUROMICRO Conf. (EuroMicro'99)* (IEEE Comp. Soc. Press, Los Alamitos, CA), *Workshop on Dependable Computing Systems*, Milan, Italy, Sep. 8-10, 1999, Vol. II, pp. 98-104.