Distributed embedded automation systems: dynamic environments and dependability

G. Deconinck1, V. De Florio2, G. Dondossola3, H. Kuefner4, G. Mazzini5, S. Calella3, S. Donatelli6

1K.U.Leuven, Elektrotechniek (ESAT) – ACCA, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
tel +32 16 32 11 26, fax +32 16 32 18 13, Geert.Deconinck@esat.kuleuven.ac.be
2CESI, Automation Business Unit, Via R. Rubattino 54, I-20134 Milan, Italy
3Siemens AG, SW & Systems Architecture, Otto-Hahn-Ring 6, D-81730 Munich, Germany
4University of Ferrara, Via Saragat 1, I- 44100 Ferrara, Italy
5TXT e-solutions, e-solutions SpA, Via Frigia 27, I - 20126 Milan, Italy
6Università degli Studi di Torino, C.so Svizzera 185, I-10149 Torino, Italy

Abstract

The DepAuDE project wants to develop a design framework to evaluate which techniques, tools and methodologies are best suited to obtain dependability for embedded systems, even when fault tolerance is concerned. 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, replicated memory, support for replication, acceptance tests, atomic actions, 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 coordinates the fault tolerance actions at runtime 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 the language 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-modal 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...
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. This approach 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 IST-2000-25434) that has been accepted 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 [DeLa00].

DepAuDE

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.

Hence, the target system is an embedded system that is distributed at several levels (see figure). • On each site, there is a distributed embedded application, which is locally interconnected via a local area network, or via dedicated connections. This (intra-site) local network is only used by the local application, and the local system has control over it. This network provides features for real-time applications.

- Between the different sites, there is an interconnection that is based on a public medium (Internet, IP-based), on which the application has less control and that is shared with other applications. This (inter-site) network is mainly used for non-real-time communication; however, nowadays and future industrial demands will impose quality-of-service requirements or require (soft) real-time behavior for these connections.

What concerns intra-site aspects, 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.

Typically, these embedded automation applications prevail in a dynamic environment due to different influences:

- Physical faults affecting parts of the intra-site system or the inter-site connection system, leading to unavailability of computing nodes, or parts of the network.
- Design faults (especially Heisenbugs) affecting the software of the application when uncommon input has to be processed or race conditions occur. This leads to the unavailability of several processes or entire nodes.
- Interaction faults from operators erroneously performing some actions again leading to unavailability of subsystems.
- Malicious faults affecting the (possibly public) inter-site connections, which may cause the unavailability of the network (e.g. denial-of-service attacks) or where the integrity of the data is endangered, etc.
- Influence of other applications that make use of the same network, leading to bandwidth reduction, non-determinism or causing load changes in the different sites.

In this dynamic environment we want to ensure the dependability of the embedded distributed applications such that they provide the highest possible service-level. More specifically, in this dynamic environment the customers want to do things like

- Remote diagnosis of problems in the local sites via the public Internet (e.g. step-by-step execution of a production process guided via visual feedback over the inter-site network).
- Remote maintenance of embedded systems.
- Remote control of embedded systems over non-dedicated inter-site connections.
• Software updates or upgrades of system modules without shutting down the entire local distributed system and while still providing services.
• Monitoring and automatic control in loosely connected embedded systems, e.g. to allow for load balancing and for the survivability of the critical infrastructure (e.g. the electricity network), where the inter-site connections are subject to the above-mentioned dynamic changes.
• Adapting fault tolerance strategies depending on the environment without actually affecting the application source code itself. (For instance, by separating the functional aspects of the application from the non-functional ones concerning fault tolerance.)
• Et cetera.

Furthermore, 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. For the intra-site aspects, 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 timed 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.

The DepAuDE project is driven by the interests of the involved industries and addresses both research and technology development. As such, the main focus is on
• the investigation part,
• the methodological part to capture requirements and to set up the modelling scenarios,
• the development of several prototypes to show shortcomings and novel solutions,
• providing feasibility demonstrators integrated in relevant industrial applications.

This project is driven by industrial requirements where more and more customers ask for more dependability in their distributed embedded applications which are interconnected via (possibly public) lines. The answer to this challenge is hence also a guarantee for the exploitation of the results, both in the internal markets of the industrial users as well as in the broader market of embedded interconnected applications.

The project phases

The following figure describes the structure of the project.

The project consists of two parallel threads (an operational thread and a methodological one) and the following four phases.

• Phase 1: Specification of dependability requirements in dynamic environments: characterisation of the requirements and the dynamic environment; methodology for collecting, specifying and validating these in the industrial application domains; set up of the modelling scenarios starting from the requirements specification.

• Phase 2: Investigation of state-of-the-art and evaluation of suitability for application domains; protocols for inter-site communication over IP networks, middleware for intra-site fault tolerance and dependability and modelling for embedded distributed automation systems.

• Phase 3: Development of intra-site and inter-site features: specification, design, modelling and implementation of selected intra-site and inter-site features. Among other, this includes dynamic fault tolerance via the framework approach, secure and real-time-suited protocols for inter-site communication, net models of fault tolerance strategies, evaluation of behaviour in dynamic environment, etc.

• Phase 4: Feasibility demonstration in industrial applications; preparation of target application, integration of selected features; prototypes. The target application domains are the command and control of electrical substations for energy distribution, and distributed embedded automation systems in large industrial applications so to allow remote diagnosis, maintenance and control.
This work is supported by IST-2000-25434 project DepAuDE. Geert Deconinck is a postdoctoral fellow of the Fund for Scientific Research - Flanders (FWO).

**Selected references**


[Dond00] G. Dondossola e.a., Tallorable fault toleRANce frameworks for embedded applications: Project Final Report, D7.9, TIRAN project (Esprit 28620), Milan, 2000, 25 pages.