Component Redundancy for Adaptive Software Applications

Authors

Ada Diaconescu (contact author), John Murphy
Performance Engineering Laboratory
Dublin City University,
Dublin, Ireland
{diacones,murphyj}@eeng.dcu.ie

Abstract

The growing complexity of computer systems has led to more complicated and expensive system design, testing and management processes, and decreased systems flexibility. In the last years, research initiatives have started to address these issues, as a great challenge of modern software engineering and in general of the whole I/T industry. Self-evolving and self-adaptive system technologies, autonomic computing and redundancy as a basis for system robustness are some of these initiatives.

With respect to this, we are trying to address the inter-related issues of self-optimisation and self-'healing' of software applications. We focus on the component-based software development (CBSD) approach as a new solution that promises to increase the reliability, maintainability and overall quality of large-scale, complex software applications.

Our goal is to propose a new component technology that adds on the existing component technologies (e.g. EJB, .Net, or CCM) to support new functionalities such as adaptability, self-optimisation, or self-'healing'. We concentrate our solution around the concept of (software) component redundancy. By this concept we understand that a number of component implementation versions, providing the same services are available, at run-time. In case one component version instance fails, or performs poorly in certain conditions, it can be replaced with another component version that offers the equivalent functionalities.
The component technology must accommodate the presence of the redundant component implementation versions and be able to use them for achieving optimal performance and robustness.
As a new requirement on component implementations, each such component has to support and provide a formal description of itself. This description includes component functionality information and performance related parameters corresponding to various environmental conditions (e.g. available resources and number of concurrent client requests).

We identified the following entities needed for utilisation and management of redundant components:
- Application Monitor - monitors and evaluates the performance of active component variants. Identifies 'problem' component(s) in application transaction chains.
- Environment Monitor - keeps track of the current available resources (e.g. memory, storage, communications bandwidth, or processing) and number of client requests.
- Component Evaluator - determines the optimal component version, based on component descriptions, current environmental conditions and decision policies.
- Component Swapping Mechanism - swaps instances of different component versions, while preserving state and references consistency.

These entities operate at run-time in an automated, feedback loop manner. The application performance is monitored and evaluated, the optimal component version is identified and activated for each component and the resulting application is monitored and evaluated.

The aforementioned entities are part of the component technology we propose. Therefore, they do not have to be (re-)implemented for each component based software application that uses this component technology. Application servers are already providing some common

services such as security or transactions. The component redundancy based optimisation and adaptability would be another such service.

We intend to concentrate on specifying the formal component description, Component Evaluator, and Component Swapping Mechanism and identify existent application and environment monitors.
Our prime goal will be to demonstrate our approach for an existent component technology (e.g. EJB) by integrating the aforementioned entities with an open-source application server and simulating environmental conditions variations.

Additional Information

http://www.eeng.dcu.ie/~diacones/workCurrent.html