INTELLIGENT DECISION-MAKING TIER IN SOFTWARE ARCHITECTURES

D. Ong and S. Khaddaj
Faculty of Computing, Information Systems and Mathematics, Kingston University London, Kingston upon Thames, U.K.

Keywords: Software Architectures, Intelligent Decision Making, N-tier Architectures, Control Systems.

Abstract: Despite recent developments there are still many challenges in the application of intelligent control systems, as intelligent decision-making providers in constructing many software architectures which are required in many applications particularly in service-oriented computing. In this work we are particularly interested with the use of intelligent decision making mechanisms for the management of software architectures. Our research aims at designing and developing an intelligent tier which allows dynamic system architecture configuration and provisioning. The tier is based on a number of logical reasoning and learning processes which use historical data and a set of rules for its analysis and diagnosis in its attempt to offer a solution to a particular problem.

1 INTRODUCTION

The adoption of a service-oriented architecture (SOA) approach in the software development lifecycle has increased significantly over recent years. The SOA approach is an extension of distributed computing, modular programming and object-oriented programming concepts, where each software component in the architecture can be considered as a service provider. Each service provider is able to operate as an individual independent unit, as well as collaborate with other services to fulfil a particular application objective. A software package which is based on business requirements and processes can be assembled via the manipulation of these services.

For many applications, especially those based on services, it is very important to have a system that is able to reconfigure itself dynamically based on its operational situation. To achieve this dynamism, this paper proposes an intelligent decision-making mechanism which can be incorporated into existing n-tier architecture as i-tier to form new n(i)-tier architecture.

We start by discussing intelligent decision making mechanisms and processes. Then, the different layers in typical software architectures are described and some problems and challenges are identified. The intelligent decision making layer is then presented which offers dynamic optimisation of the way services (resources) in the n-tier architecture are utilised. Then, the layer potential use as an intelligent service provider is discussed. We conclude with some suggestions for future work.

2 INTELLIGENT DECISION MAKING MECHANISMS

Most intelligent control systems have some form of decision making mechanisms build into their systems, but many obstacles remain as they are usually built for a specific purpose / scenario, relying heavily on predetermined algorithms, and autonomous behaviour that is based on preconfigured / static configurations (Kramer, 1985), (Sun , 2000), (Satoh, 2002), (Beckert, 2006). Historically, the control unit of many intelligent systems is not easily adapted to other architectures / systems because it is statically designed to solve a particular problem.

To overcome the above challenges, this work proposes an intelligent control unit that is capable to deploy a generic decision-making mechanism across various software architectures. The intelligent decision-making unit provides decision-making functionality for the software architecture in the form of system configuration and provisioning. Based on the decision-making scope, the decision-

Ong D. and Khaddaj S. (2010).
INTELLIGENT DECISION-MAKING TIER IN SOFTWARE ARCHITECTURES.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Databases and Information Systems Integration*, pages 355-358
DOI: 10.5220/0002868103550358

making unit is expected to make a sensible decision upon every request, and to learn from its past decisions via the received feedback.

The main obstacle in developing intelligent decision-making mechanisms is to understand how we perceive an "intelligent" entity (Ong and Khaddaj, 2009). A general concept on how an intelligent decision-making ability is achieved using combined functionalities is shown in figure 1. This is used as a general guideline in constructing the proposed intelligent control unit where the learning and logical reasoning processes use historical data for its analysis and diagnosis as well as sets of rules, both pre-defined and self-defined, to offer a solution to a particular problem.

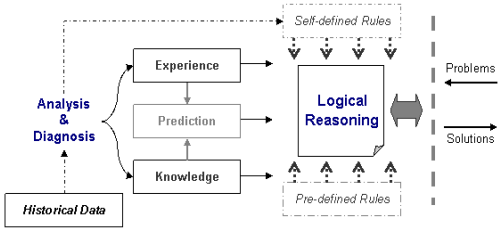

Figure 1: A decision-making ability.

3 SOFTWARE ARCHITETURES

The traditional software architectures have a tendency toward developing computer applications as standalone visible software (commodity), where it can be sold as an off-the-shelf product. Typically, it is designed for a specific business function (e.g. an in-house payroll system) or a specialised general-purpose application such as word processing, desktop publishing, operating system, etc.

However, continuous growth in the integration and globalisation of the current computer infrastructures has offered a new opportunity for a "next step" in software evolution. A service-oriented architecture (SOA) has emerged as a popular architectural platform, where it has moved significantly from the traditional design (Choset, 2000). This has created a great opportunity for the full integration of existing applications into one universal service provider with each software component capable of acting autonomously as a service provider. This service provider can range from a small task (e.g. statistical calculation, sorting functions, etc.) to a larger enterprise solution, where it can consist of an integration of various small tasks to form a complex service such as online banking, e-store, etc. Furthermore, any improvement or deployment of this component to other solutions can be accomplished without any noticeable effect to the overall infrastructure (Pfister, 2002), (Gyurjyan, 2003), (Soundararajan, 2008), (Bar-Kana, 1989), (Frankovic, 2009).

An n-tier concept is commonly used in designing an SOA solution, where it is generally adopted and implemented as a software development scheme particularly for web services (e.g. the core design concept of Microsoft .Net (Microsoft, 2007)).

3.1 N-tier Architecture

The n-tier architecture's aim is to provide an approach that unifies business, technical and data access logics into one single flexible software infrastructure. Business logic provides aims and purposes of the software (e.g. content management for different types of user, on-line payment and billing solutions, etc.). Technical logic gives a technical integration of the business logic into a viable technical solution (e.g. interpretation and transformation of screen display according to level of information or privileges set by the business logic). Lastly, data access logic provides a data management solution for the software such as data access solution, data storage solution, data translation solution, etc.

The n-tier architecture utilises a component-based approach in its design practices. This approach permits the architecture to be more flexible and acceptable to any changes with a minimal impact to its structural integrity. The flexibility offered by this approach enables the architecture to reuse existing components, which has led to a large reduction in the redundancy of functions (codes).

In general, n-tier architecture can be divided into three main components – "User Interface", "Application Structure" and "Data Storage" (figure 2). The "user interface" component provides a physical medium for a software solution to engage with an end-user. The "application structure" component incorporates business, technical and data access logics into three integrated tiers to provide a software solution. The "data storage" component gives a data storage and management facility to an "application structure" component.

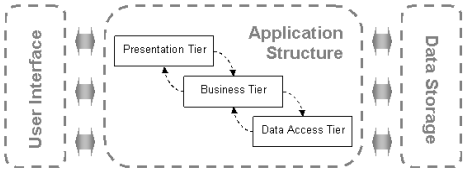

Figure 2: General concept of n-tier architecture.

4 AN INTELLIGENT DECISION-MAKING LAYER

Earlier, it was indicated that there is a requirement to design a system that can reconfigure itself dynamically based on its operational situation. In order to achieve this dynamism, a new n(i)-tier architecture is proposed as an enhancement to the current n-tier architecture, where a newly proposed intelligent decision-making layer (i-tier) is introduced. This i-tier is expected to take over the responsibility of handling the architecture's communication medium (figure 3) by managing every communication requirement autonomously through an intelligent resources delivery and management facility.

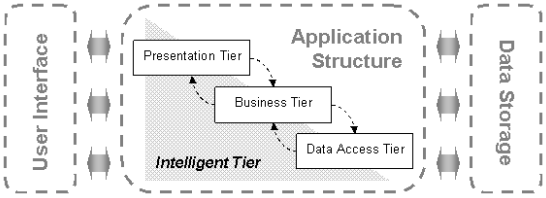

Figure 3: General concept of n(i)-tier architecture.

The fundamental characteristics of the i-tier in n(i)-tier architecture could be described as follows: -

a) To be capable of detecting and adapting to different communication protocols used by the architecture's tiers.

b) To be capable of adapting to any changes occurring in the infrastructure.

c) To be capable of optimising and fixing any faulty elements detected within its infrastructure.

d) To be capable of initiating its own investigation when there is a problem discovered or a new element introduced to the infrastructure.

e) To be capable of recovering from any failures.

The intelligent resources delivery and management facility is based on a dynamic decision-making mechanism. The role of intelligent tier would be to make the best possible decision to reflect the current situation of the whole architecture. Integration of these decision-making processes into one independent tier enables a wider identification and evaluation of the architecture's overall operating status that inadvertently contributes to the formation of a better final outcome. This process is determined by pattern identification methods and types of logical reasoning in the tier's decision-making mechanism (Ong and Khaddaj, 2009).

Initially, the presentation tier sends an end user's requests to the data access tier. The data access tier needs to access information provided by the business tier, in order to evaluate which type of information is required by the presentation tier. If the business tier almost reaches the breaking point in its processing capacity, a performance degradation of the overall architecture will slowly emerge (i.e. a snowball effect) because the presentation tier is still continually sending requests to the data access level and is not aware of any problems in the business tier. The intelligent tier, which oversees the status of the overall architecture, instructs (co-ordinates) the presentation tier to reduce the frequency of requests sent to the data access tier. It will therefore alleviate the processing pressure from the business tier and the overall architecture in general, thus facilitating optimum processing of the data and an improvement in the performance of the n(i)-tier as a whole.

4.1 Intelligent Service Provider

As mentioned earlier the current n-tier architecture has three tiers, which are capable of interacting with each other to achieve a software solution for the end user. Moreover, it relies heavily on a static configuration to achieve its functioning goals. This configuration often requires manual modification to reflect its current operational relevancy. However, a dynamic decision-making process in n-tier architecture is essential, as any software solutions produced must reflect the current operating environment.

In order to achieve this dynamic process, the intelligent tier was proposed earlier to provide the intelligent decision-making mechanism for the architecture tiers. The i-tier layer can be deployed as an n(i)-tier based solution for an intelligent service provider where it is taking over the decision-making responsibilities from other tiers (figure 4). The intelligent service provider can be used in service-oriented architectures where many services need to interact to produce a workable application.

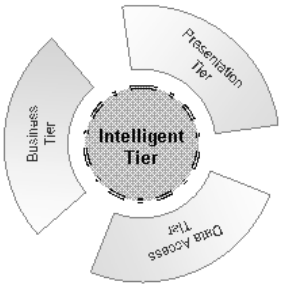

Figure 4: intelligent service provider.

The dynamic decision-making process in each tier is delegated to the intelligent tier, where the best possible decision is made to reflect the current situation in the whole architecture. Integration of these decision-making processes into one independent tier enables a wider identification and evaluation of the architecture's overall operating status that inadvertently contributes to the formation of a better final outcome. This process is determined by pattern identification methods and types of logical reasoning in the tier's decision-making mechanism.

5 CONCLUSIONS

In this paper an intelligent control system framework for the dynamic configuration and operation of software architectures has been presented. To achieve a high flexibility and dynamism in the framework design, the paper has looked into an intelligent decision-making layer which can be adopted as a control unit of the intelligent control system. This was then applied into the current development of n-tier software architecture. The need of introducing an intelligent decision-making layer that is capable of acting instantaneously in response to any changes to its requirements and environment was also discussed. The paper attempted to offer a viable technical design and solution to constructing an n(i)-tier infrastructure for software development environments. It is suggested that the introduction of an intelligent decision-making layer in the n-tier design would enable software architectures such as the service-oriented architecture to be more flexible and adoptable. However, the realisation of the proposed architecture to its full potential would be considered in future work.

REFERENCES

Kramer, J, 1985. Dynamic Configuration for Distributed Systems. *IEEE Transactions on Software Engineering*, *11(4),* 424-436.

Sun Microsystem, 2000. Dynamic Host Configuration Protocol. *Technical White Paper*. <http://www.sun.com/software/whitepapers/wp-dhcp/dhcp-wp.pdf>.

Satoh, I, 2002. Dynamic Configuration of Agent Migration Protocols for the Internet. In *Proceedings of International Symposium on Applications and the Internet (SAINT 2002)*, IEEE Computer Society.

Beckert, B, 2006. Intelligent Systems and Formal Methods in Software Engineering. *IEEE Intelligent Systems*, 72-78.

Ong, D., Khaddaj, S., 2009. The Potential of Imprecision In Intelligent Decision Making. In *International Conference on Application of Digital Information and Web Technologie,* London, UK.

Choset, H, 2000. Sensor Based Exploration: Incremental Construction of the Hierarchical Generalized Voronoi Graph. *International Journal of Robotics Research, 19(2),* 126-148.

Pfister, S, 2002. Weighted Range Sensor Matching Algorithms for Mobile Robot Displacement Estimation. In *IEEE International Conference on Robotics and Automation 2002*.

Gyurjyan, V, 2003. FIPA agent based network distributed control system. *Computing in High Energy and Nuclear Physics*, 24-28.

Soundararajan, K (2008). Design patterns for real-time distributed control system benchmarking. *Robotics and Computer-Integrated Manufacturing, 24(5)*, 606-615.

Bar-Kana, I, 1989. Unsupervised parallel distributed computing architecture for adaptive control. In *Proceedings of IEEE International Symposium on Intelligent Control*, 174-178.

Frankovi¡c, B, 2009. Creation of Intelligent Distributed Control System Based On Multi-Agent Technology. *Journal of Electrical Engineering, 60(1)*, 29-33.

Microsoft, 2007. *MSDN Solution Architecture Center*. <http://msdn2.microsoft.com/en-gb/architecture/default.aspx>.

Khaddaj, S., 2008. The Impact of Component and Distributed Systems on Software Quality. In *Semantic Enterprise Computing,* 30-38.