Service Oriented Architecture-based design of benchmarking testbeds for advanced decision support systems for hotel management

Alfonso Durán Heras, Isabel García Gutiérrez, Teresa Sánchez Chaparro

Ingeniería de Organización. Escuela Politécnica Superior. Univ. Carlos III de Madrid. Avda. de la Universidad, 30. 28911 Leganés, Madrid. alfonso.duran@uc3m.es, isabel.garcia@uc3m.es, teresa.sanchez@uc3m.es

Abstract

Within the context of a multiyear, multiparty research project on yield management-based decision support systems for the hotel sector, this paper analyzes the applicability of benchmarking testbeds for the comparative evaluation of the algorithms proposed in the project. The various functional, organizational and technical requirements that such a testbed would have to meet are studied, as well as their design implications. These requisites are then matched with the traits of alternative development platforms, and the choice of a SOA/ Web Services approach is justified. Implications of this choice for the utilization of the testbed for other research projects and potential generalization of this analysis for other application domains are explored.

Keywords: Benchmarking testbeds. DSS. Service Oriented Architecture. .Net. Yield management.

1. Advanced decision support systems for hotel management. Yield/ Revenue Management algorithms

In recognition of the importance of the adoption by the Spanish hotel sector, and specifically of its Small and Medium Enterprises, of internationally competitive management practices and systems, a multi-year, multi-centre research project was approved and funded at the end of 2005 by the Spanish Ministry of Education*. This project is aimed at the development of an advanced Decision Support System (DSS) for hotel management.

A key element of a DSS in the hotel sector is the utilization of Yield Management (YM), also known as Revenue Management. YM is currently a hot research area (Chiang et al, 2007). Even though YM was born in the airline industry, it has been adopted in other sectors including the hotel sector (Baker and Collier, 1999 and 2003). YM involves dynamic methods and optimization heuristics to forecast demand, allocate perishable assets (for hotels, the hotel rooms) across rate classes, decide when and by how much to overbook and what price to charge different rate classes in order to maximize revenues for the firm.

Within this research project, a series of optimization heuristics, along with the corresponding demand forecasting requirements, has been developed to address the various alternative management approaches and scenarios. This paper is concerned with the design challenges and options involved in the development of a benchmarking testbed to be used to test the algorithms created throughout the research project. It will also attempt to establish whether the applicability of either the testbed itself or its design choices is generalizable beyond that application.

* This work stems from the participation of the authors in a research project funded by the Spanish Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica 2004-2007 (MEC-DGI-SGPI), reference DPI2005-09132-C04-04, title Sistema avanzado de ayuda a la toma de decisiones para la gestión hotelera.

2. YM research results validation and benchmarking. The role of testbeds

As discussed above, a number of alternative Yield/Revenue management algorithms, on which the corresponding DSS could be based, have been published in the last few years, and additional ones are being developed within this research project.

Such algorithms may have been conceived as a result of logical reasoning, modelling and simulation studies, lab-scale experiments, pilot-scale experiments, full-scale applications or different combinations of those. However, the majority of the suggested strategies might have never been thoroughly evaluated by anyone else but the authors and their co-workers, let alone implemented in any full-scale applications and properly validated. It would be beneficial to both the scientific and practical DSS/YM communities, and essential for the success of the current research project, to have an objective, structured method to verify (at least within certain probabilities) whether or not a newly proposed YM algorithm is able to accomplish the DSS user's business objectives more efficiently than other available algorithms.

However, the evaluation and comparative benchmarking of YM algorithms and the corresponding hotel DSS, either by practical implementation or through simulation are difficult (Jeppsson and Pons, 2004). This is due so several causes:

- Hotels would be understandably reluctant to implement a DSS based on untested YM algorithms in such critical business processes as room pricing or room allocation. Besides, as described below, it would be difficult to derive from the application of an YM based DSS in an individual hotel reliable conclusions on its relative merit.

- Observable business results are caused by a network of time-varying interrelated factors, such as demand, making it difficult to isolate the effect of a given factor, such as the choice of the YM algorithm.

- Even within a DSS, several critical sub-modules coexist, e.g. demand forecasting and YM-based decision algorithms, again making it harder to isolate the effect of the YM algorithm choice.

- Different YM algorithms correspond to alternative hotel business process designs, e.g., hotels with a fixed, predetermined number of fixed-price room categories (in that case, the YM algorithm would allocate each available room to one of those price categories) or hotels in which the room price changes as the actual occupancy date approaches (in that case, the YM algorithm would suggest that price). Similarly, within each business process design, there will be algorithms of various levels of complexity, which take different inputs into account to perform the calculations (e.g., might take into account, or not, the current prices of competing hotels).

- Business evaluation criteria might be dependant on the business priorities of the hotel involved.

Similar challenges are also present, in different degrees, in various application areas, and specifically in the evaluation and testing of Manufacturing Planning and Control Systems (MPCS); the YM-based DSS explored here could, in a way, be conceived as being part of a MPCS. In some application domains in which the problem context and objectives can be narrowly and precisely defined, such as very specific optimization algorithms, or when the only criteria to be tested is speed, this has led to the development of widely used benchmarking

suites (Chabrier, 2006; Head et al, 2005 ; Krishna et al, 2005; Hempstead et al, 2004).

However, the simplistic application of a benchmarking suite to an YM algorithm could provide a misleading evaluation of its merits when implemented in a DSS designed to operate in real hotel environments. According to Seltzer et al (1999), who stress the need for application-specific measurements, "...s*uch results are, at best, useless and, at worst, misleading*...". The ability of a given YM algorithm to allocate resources when faced with predictable and time-invariable demand, certainty about resource (room) availability, mechanistic customers, absence of interaction and feedback loops and a predetermined business process design does not necessarily translate into its capacity to achieve satisfactory results in the highly complex real environment described above.

Simulation based benchmarking could offer cost-effective means for a more realistic evaluation of these algorithms. However, attaining comparability for the results requires that the alternative algorithms are tested under "ceteris paribus" environments. This requires that the simulation environment, the external demand, other external influences, other modules such as demand forecasting etc. a) do not introduce a bias in the evaluation of the YM algorithm b) their inherent stochastic variability is ironed out through a sufficiently complete experimental design. However, the cost and complexity of creating a comprehensive and realistic simulation environment on which to test each algorithm, along with the accompanying methodology and standardised simulation procedures, requires devoting significant resources to the development of a testbed involving all these elements (simulation environment, methodology, procedures) that can provide a functional and reliable (meaning that another researcher conducting the same study would conclude the same results) testing environment.

Similar requirements in other application domains in which it is also difficult to narrowly and precisely define the problem context and objectives has led to the development of a variety of application domain-specific benchmarking platforms (Aref et al, 2004; Jeppsson and Pons, 2004; Elhanany and Tabatabaee, 2003; Freya et al, 2003; Wu et al, 2003). This effort has often been undertaken by publicly sponsored multi-party research projects, such as EU research program European Co-Operation in the field of Scientific and Technical research Action 682 (COST 682) Working Group No. 2/COST 624 (whose benchmarking platform development activities are described in http://www.ensic.inpl-nancy.fr/COSTWWTP/Benchmark.htm), or the Special Interest Group SIG4 on Benchmarking and Performance Measures within Network of Excellence on Intelligent Manufacturing Systems, IMS-NoE (http://www.ims-noe.org/sigN.asp?sig=4; Cavalieri el al, 2003). This allows for the general dissemination and usage of the testbed, as reported by the COST 624: "*...The software implementations have been freely distributed to research groups on almost every continent and the basic benchmark is available as a built-in feature within several of the major commercial software packages for WWT modelling and simulation (e.g. WESTs, SIMBAs, GPS-Xt) ... this has been accomplished within a time period of 5 years*...". Even in such projects, while great attention is devoted to provide the flexibility to accommodate different operating environments, ad-hoc modifications can be required to make it usable under some specific design and operational characteristics (Abusam et al, 2004)

3. Revenue management benchmarking testbed requirements and design implications.

The benchmarking testbed required for the testing of the YM-based DSS for hotel management is therefore faced with quite stringent requirements, from the functional, organizational and

technical viewpoints, that will have significant design implications.

3.1. Functional requirements

Critical functional requirements for the benchmarking testbed can be summarized in three:

- Fidelity. For the results of the algorithm testing to be meaningful, the testing platform should bear as close a resemblance to the real hotel environment as possible. This is particularly important (and difficult) in the case of the customer behaviour.

- Flexibility. In order to allow for comparison of YM algorithms that correspond to alternative hotel business process designs, the testbed must provide the flexibility to reflect these various designs, e.g., predetermined fixed-price room categories, room prices contingent on the proximity of the actual occupancy date, etc. Similarly, within each business process design, it must accommodate algorithms of various levels of complexity, which take different inputs into account to perform the calculations (competition...).

- Reliability and comparability. Reliability would imply that another researcher conducting the same study would conclude the same results. This is a non-trivial requirement while comparing YM algorithms that correspond to the same business process designs and level of complexity, and thus utilize the same inputs and forecasts. However, the hardest challenge stems from the comparability requirement, which involves providing meaningful, real-life resembling business performance benchmarks between YM algorithms that correspond to different business process designs and/ or levels of complexity, and thus utilize different inputs and forecasts.

The major design implications of these functional requirements are:

- Simulation based. To meet the fidelity requirement, the benchmarking platform must be simulation based. The simulation environment must reflect, on one hand, the operation of the hotel, and, most critically, the complex customer behaviour and hotel-customer interaction, entering the realm of what has become known as social simulation.

- Modularity. Being able to implement quite different business process designs and YM algorithms without affecting the simulation environment, to meet the flexibility requirement, calls for a modular design, with well defined interfaces between the modules.

- Super-set design and simulation. The comparability requirement is particularly hard to meet. As described, different YM algorithms to be benchmarked utilize different inputs and forecasts. In conventional algorithm simulation testing, ad-hoc simulation environments only recreate the part of the outside environment that interacts with the specific algorithm being tested. Consequently, the bespoke interface that the simulated environment presents to the tested algorithm is also specific to its interaction. As an example, if an algorithm assigns rooms to fixed-price categories with a given anticipation with respect to the occupancy date, the ad-hoc simulation might simply simulate the number of arrivals for each price category for the whole of that lead time. If, however, another algorithm calls for the dynamic adjustment of prices as the occupancy date approaches, that simulation would not be valid. If a different customer simulation is then undertaken, the results are no longer comparable. Furthermore, the parameter-passing interface between the simulation module and the hotel-DSS module would have to be adapted. This leads to a super-set design and simulation requirement: the various modules, and particularly the customer simulation module and

the corresponding interfaces, must be rich enough to support the input requirements of any of the algorithms to be tested. It must therefore implement all the traits that might be required by any of the algorithms, and be able to adjust behaviour and dialog accordingly. That allows the same subjacent customer flow to be simulated for the various algorithms to be benchmarked (each of which will interact with a subset of the customer traits), thus allowing comparability. This is similar to real-life customers, each of which exhibits a complex, multifaceted behaviour, that allows it to interact both with a hotel that presents it with a fixed set of choices and with another one that, for example, provides a spot quote when prompted.

3.2. Organizational requirements

Implementing the requirements outlined will indubitably require a significant software development effort. The research project provides funding only for Engineering Management faculty members, whose profile is not aligned with the software development requirements and whose dedication to the project is unlikely to provide the required amount of effort. Therefore, it is important to reduce the software development requirements as much as possible, by restricting it to the truly application domain-specific, innovative components. Furthermore, there is a practical need to involve non-permanent staff in the development of the various testbed modules, particularly students carrying out academic supervised projects within their Master or Engineering curricula. Given the multiyear nature and evolving nature of the project, this implies that developers will rotate, potentially several times, during the project lifecycle. These considerations lead to three organizational requirements:

- Different developers or development teams must be able to work simultaneously and to a large extent asynchronously on different modules. If a student finalizes his/her project he can not be forced to wait until another student finalizes his. Additionally, work on a given module started by one developer could have to be completed or refined by another one.

- It should be feasible to refine, exploit and eventually redesign these modules long after their original developers are no longer available.

- Software development should be restricted to the application domain-specific, innovative components, by reusing or sharing as much pre-existing applicable functionality as feasible.

The major design implications of these organizational requirements are:

- Development should be componentized, allowing for reuse of existing components and providing internal cohesion and external decoupling (or, rather, low coupling).

- This loose coupling should ideally be implemented as message coupling, in which one module interacts with another module through a stable, public interface and does not need to be concerned with the other module's internal implementation. That would allow progress, and changes, in one module to take place without interfering with the others.

- Modules have to be carefully documented, following sound software engineering practices.

- The development environment/ language should be a widely used de-facto standard, or, better still, seamlessly support various alternatives. This would reduce the entry barrier for

non-permanent, potentially part-time collaborators. Furthermore, it would allow the project to tap shared resources, such as publicly available libraries on application domains such as simulation or agents.

3.3. Technical requirements

In addition to the abovementioned design implications, one more technical requirement and corresponding design implication can be highlighted: the relationship between the various modules, and specifically between the demand simulation subsystem and the hotel simulation subsystem (which, in turn, includes both the demand forecasting and the heuristic algorithms) must be interactive, of a transactional nature, as opposed to merely feeding a set of simulated demand events or "arrivals" like in other simulation environments.

4. Platform alternatives: requirements matching. SOA/ Web Services

Within the framework of the current design guidelines for research testbeds, largely derived from the manufacturing environment (Cavalieri el al, 2003), various alternative development platforms were then matched against these design implications:

- Simulation based
- Modularity
- Super-set design and simulation
- Componentized development
- Loose message coupling through public interfaces
- Software engineering practices including documentation
- Standard development environment/ language or seamless multi-language support
- Transactional interaction among modules

The "simulation based" and "super-set design and simulation" requisites suggested specialized simulation or agent-oriented platforms, such as Witness or Multi Agent Systems (MAS) development environments. However, these systems generally did not meet the "standard development environment/ language or seamless multi-language support" and "transactional interaction among modules". General purpose Rapid Application Development environments did not sufficiently facilitate the simulation orientation, reusability of existing solutions and transactional interaction. The "modularity" and "componentized development" requisites suggested an object oriented approach based on widely used languages such as C++ or Java, however consistently implementing in these environments the "loose message coupling through public interfaces" and "transactional interaction among modules" was deemed to require substantial effort.

Thus the analysis then focused on a Service Oriented Architecture (SOA) / Web Services approach and supporting development framework (Erl, 2005; MacKenzie, 2006), specifically the .Net Microsoft framework, focusing on how would such an environment meet the requisites of this specific testbed project, and also on the potential for generalization of this platform evaluation to other research environments.

The requirement fit was particularly good for five of the requisites: Modularity, loose message coupling through public interfaces, transactional interaction among modules, componentized development and standard development environment/ language or seamless multi-language support:

– Support for modularity, loose message coupling through public interfaces and transactional interaction among modules derives directly from the Web Services architecture.

  - In Web Services environments, functionality is developed in independent modules that implement services which can then be invoked by the other modules. WSDL, the Web Services Description Language, is an XML format that allows service interfaces to be described. Thus, if the hotel simulation module, for example, is modified to reflect different business process designs and YM algorithms, that would be isolated from the simulation environment. Even if the dialog with the simulation environment is modified, that would be described through an updated version of the WSDL service description.

  - Actual interaction would take place through the exchange of XML-based SOAP messages, normally using HTTP; they would follow the Remote Procedure Call (RPC) pattern, whereby one module (the client) sends a request message to another module (the server), and the server sends a response message to the client. That would implement the transactional interaction among modules. Being XML based, performance of the interaction would be limited (that could eventually be solved through XML binary serialization), however for a research oriented environment performance would not be critical.

– Componentized development, aimed at the reutilization of existing components, is a key design criteria in Web Services. Even though at present the availability of sharable services from the research community conforming to these standards still lags behind that of libraries and other reusable components (private sector environments such as ERP solutions are, however, actively reconfiguring their proposals towards the service orientation), Web services provide the foundation for incorporating existing and future components. As discussed by Gold el al (2005), it enables the management of the integration of this flow of Web Services, whether external or produced by successive project members, in a way reminiscent of a supply chain.

– Standard development environment/ language or seamless multi-language support is achieved in its fullest extent through the Common Language Runtime (CLR) component of Microsoft's .NET. This virtual machine is Microsoft's implementation of the Common Language Infrastructure (CLI) standard, which defines an execution environment for program code. The CLR runs a form of bytecode called the Microsoft Intermediate Language (MSIL), Microsoft's implementation of the Common Intermediate Language. Thus, any CLI/CLR compatible language, such as C++ or C#, could be used.

Adherence to sound Software engineering practices including documentation is not directly related to the platform choice, even though the inherent traits of modularity, cohesion and decoupling of the Web Services approach facilitate their adoption.

The two other requisites, simulation based and super-set design and simulation, require access to existing shared libraries or specialized tools/ development environments to minimize the development effort. The seamless multi-language support described above facilitated this.

Furthermore, the most popular open source toolkits are being migrated to the .Net environment, such as the widely used Recursive Porous Agent Simulation Toolkit (Repast), for which Repast .NET is the most recent implementation (North et al, 2006).

With regard to the potential applicability of the benchmarking testbed to other research projects, this design offered the potential of reusing, for example, the simulation module to benchmarking other research findings whose implementation would also take place in a customer-facing environment. It could, for instance, be used to test production planning or independent demand inventory management policies that involve enticing customers with monetary incentives (demand management) to better align demand and supply or to limit the inventory impact of demand fluctuations.

5. Conclusions

To reduce the gap between academic research and the transfer and actual adoption of research results, an objective, structured, credible method to estimate whether the newly proposed approaches will contribute to achieving the adopting organization's business objectives is highly desirable. In certain areas this can be achieved through appropriate research benchmarking testbeds.

These testbeds, however, are subject to quite stringent design requirements. For the specific case analyzed in this paper, aimed at benchmarking Yield Management-based Decision Support System for hotel management, the major functional, organizational and technical requirements identified included: Fidelity, flexibility, reliability and comparability, module independence, developer independence, reusability, and conversational module interaction. These translated into the design implications of: being simulation based, modularity, super-set design and simulation, componentized development, loose message coupling through public interfaces, adoption of sound software engineering practices including documentation, standard development environment/ language or seamless multi-language support and transactional interaction among modules.

Service Oriented Architecture (SOA) / Web Services platforms such as Microsoft's .Net provided the best fit for those requirements among the platforms analyzed. The rationale for this conclusion, presented in this paper, could facilitate the evaluation of the potential generalizability of the analysis to other application domains, on a case-by-case basis. Furthermore, the intrinsic modularity of the proposed testbed should allow its partial reutilization in subsequent research projects.

References

Abusam, A.; Keesman, K.J.; Spanjers, H.; van Straten, G. (2004). Benchmarking procedure for full-scale activated sludge plants, Control Engineering Practice, Vol. 12, No. 3, pp. 315-322.

Aref, W.; Catlin, A.C.; Elmagarmid, A.; Fan, J.; Hammad, M.; Ilyas, I.; Marzouk, M.; Prabhakar, S.; Tu, Y.C.; Zhu, X. (2004) VDBMS: a testbed facility for research in video database benchmarking, Multimedia Systems, Vol. 9, No. 6, pp. 575-585

Baker, T.K.; Collier, D. (1999). A comparative revenue analysis of hotel yield management heuristics. Decision Sciences, Vol. 30, No. 1, pp. 239-263.

Baker, T.K.; Collier, D. (2003). The benefits of optimizing prices to manage demand in hotel

revenue management systems. Production and Operations Management, Vol. 12, No. 4, pp. 502-518.

Cavalieri, S.; Macchi, M.; Valckenaers, P. (2003). Benchmarking the performance of manufacturing control systems: design principles for a web-based simulated testbed. Journal of Intelligent Manufacturing, Vol. 14, No. 1, pp. 43–58.

Chabrier, A. (2006). Vehicle Routing Problem with elementary shortest path based column generation, Computers & Operations Research, Vol.33, No.10, pp. 2972-2990

Chiang, W.C; Chen, J.C.H.; Xu, X. (2007). An overview of research on revenue management: current issues and future research. Int. J. Revenue Management, Vol. 1, No. 1, pp. 97-128.

Elhanany, I.; Tabatabaee, V. (2003). Benchmarking next-generation switch fabrics, Computer, Vol. 36, No. 10, pp. 109-110.

Erl, T. (2005). Service-oriented architecture : concepts, technology, and design. Prentice Hall.

Freya, D.; Nimisb, J.; Wörn, H.; Lockemann, P. (2003). Benchmarking and robust multi-agent-based production planning and control. Engineering Applications of Artificial Intelligence, No. 16, pp. 307–320.

Gold, M.; Corral, K.; Demirkan, H. (2005). Database schema design for a Web services supply chain manager: requirements and proposed infrastructure, Information Systems Frontiers, Vol. 7, No. 3, pp. 257-271.

Head, M.R.; Govindaraju, M., Slominski, A.: Liu, P.; Abu-Ghazaleh,N.; van Engelen, R.; Chiu, K., Lewis, M.J. (2005). A Benchmark Suite for SOAP-based Communication in Grid Web Services. Proceedings of the 2005 ACM/IEEE conference on Supercomputing, p. 19.

Hempstead, M.; Welsh, M.; Brooks, D. (2004).TinyBench: the case for a standardized benchmark suite for TinyOS based wireless sensor network devices, Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, pp. 585-586.

Jeppsson, U.; Pons, M.N. (2004). The COST benchmark simulation model—current state and future perspective. Control Engineering Practice, No. 12, pp. 299–304.

Krishna, A.S.; Natarajan, B.; Gokhale, A.; Schmidt, D.C.; Wang, N.; Thaker, G. (2005). CCMPerf: a benchmarking tool for CORBA component model implementations, Real-Time Systems, Vol. 29, No. 2-3, pp. 281-308.

MacKenzie, M; Laskey, K.; McCabe, F.; Brown, P.F.; Metz, R. (2006). OASIS Reference Model for Service Oriented Architecture 1.0. OASIS (Organization for the Advancement of Structured Information Standards) Standard, available at http://docs.oasis-open.org/soa-rm/v1.0/

North, M.J.; Collier, N.T.; Vos, J.R. (2006). Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit, ACM Transactions on Modeling and Computer Simulation , Vol. 16, No. 1, pp. 1-25.

Seltzer, M.; Krinsky, D.; Smith, K.; Zhang, X. (1999). The case for application-specific benchmarking, Proceedings of the Seventh Workshop on Hot Topics in Operating Systems pp. 102-107.

Wu, J.; Pan, W.M.; Jin, J.M.; Wang, Q.R. (2003). Performance evaluation and benchmarking on document layout analysis algorithms, Proceedings of the 2003 International Conference on Machine Learning and Cybernetics, Vol. 4, pp. 2246-2250.