cloud-ATAM: Method for Analysing Resilient Attributes of Cloud-Based Architectures

David Ebo Adjepon-Yamoah

Centre for Software Reliability,
School of Computing Science,
Newcastle University,
Newcastle-upon-Tyne NE1 7RU, UK
Email: d.e.adjepon-yamoah@ncl.ac.uk

Abstract. In this work, we argue that the existing architecture evaluation methods have limitations when assessing architectures interfacing with unpredictable environments such as the Cloud. The unpredictability of this environment is attributed to the dynamic elasticity, scale, and continuous evolution of the cloud topology. As a result, architectures interfacing such unpredictable environments are expected to encounter many uncertainties. It is however, important to focus on, and present holistic approaches combining aspects of both dynamic and static analysis of architecture resilience attributes. This paper introduces an ATAM derived methodology - *cloud-ATAM* - for evaluating the *trade-off* between multiple resilience quality attributes (i.e. *availability* and *performance*) of a cloud-based Reactive Architecture for *Global Software Development*.

Keywords: ATAM, Resilient architectures, Cloud computing, GSD

1 Introduction

Service Oriented Architecture (SOA) [1] might be treated as a state of the art approach to the design and implementation of enterprise software, which is driven by business requirements. Within the last decade a number of concepts related to SOA have been developed, including enterprise service bus, web services, design patterns, service orchestration and choreography, and various security standards. Due to the fact that there are many technologies that cover the area of SOA, the development and evaluation of SOA compliant architectures is especially interesting [2]. The buzzing concept of *cloud computing* represents a mix of most of these concepts and hence, serves as a good SOA example.

In this work, we argue that the existing architecture evaluation methods have limitations when assessing architectures interfacing with unpredictable environments such as the Cloud [3]. This is because the cloud environment is fundamentally different from the classical environments for which most software evaluation methods were developed [4], [3]. The unpredictability of this environment is attributed to the dynamic elasticity, scale, and continuous evolution of the Cloud topology (e.g. due to new services, mash-ups, unpredictable modes

of service use, fluctuations in QoS provision due to unpredictable load/growth etc.) [4]. As a result, architectures interfacing such unpredictable environments are expected to encounter many uncertainties. These challenges call for holistic approaches combining aspects of both dynamic and static evaluation. The **aim** of this research is to present a methodology - *cloud-ATAM* - for evaluating the trade-off between multiple resilience quality attributes of *small-to-medium size* (i.e. ISO/IEC 14143:1998 & COSMIC Full FP 2.2 for classifying system size) architectures in unpredictable environments such as the Cloud. Our approach applies a well established method in software architecture analysis called Architectural Trade-off Analysis Method (ATAM) [5]. ATAM generates a number of outputs such as: a prioritised list of quality attributes, a list of architectural decisions made, a map linking architectural decisions to attributes, lists of risk and non-risks, and lists of sensitivities and trade-offs. In this work, we use the derived *cloud-ATAM* to *design* the Reactive Architecture [6], and to *analyse* the trade-off between the availability and performance attributes. To support the analysis of the Reactive Architecture, *cloud-ATAM* considers a non-trivial set of *scenarios*, and plan to use a particular specialisation of architectural styles called *attribute-based architectural styles* (ABASs) [7]. We answer the research question: *"What is the trade-off between availability and performance quality attributes identified by the cloud-ATAM for the cloud-based Reactive Architecture?"*

2 Background

A brief overview of the ATAM, and ABAS are presented as a precursor for our discussion about the *cloud-ATAM* in the following section.

2.1 Quality Attribute Trade-Off Analysis with ATAM

Architectural analysis is a key practice for organisations that use Software Architectures (SAs). This is essential because SAs are complex and involve many design *trade-offs*, and ensures that *architectural decisions* appropriately mitigate *risks*. The ATAM is considered as a matured and validated scenario-based SA evaluation method [8]. The *inputs* of the ATAM are *scenarios* elicited by stakeholders and documented descriptions of the architecture. The ATAM is typically constituted with nine steps: (1) Present the ATAM, (2) Present the Business Drivers, (3) Present the Architecture, (4) Identify Architectural Approaches, (5) Generate the Quality Attribute Utility Tree, (6) Analyse the Architectural Approaches, (7) Brainstorm and Prioritise Scenarios, (8) Analyse the Architectural Approaches, and (9) Present Results. The goal of the ATAM is to analyse architectural approaches with respect to *scenarios* generated from business drivers for the purpose of identifying *risk points* in the architecture. This is achieved by a disciplined reasoning about SA relating to multiple quality attributes. There are two important classifications of *risk points* namely *sensitivity points* and *trade-off points*. A *sensitivity point* affects the achievement of one quality attribute; a *trade-off point* affects the achievement of more than one quality attributes, where

one improves and the other degrades. These *risk points*, together with extensive documentations of the architecture, scenarios, and quality-attributes analysis are the *products* of ATAM. The ATAM also explicitly relates architectural risks and trade-offs to business drivers.

2.2 Attribute-Based Architectural Style

An Attribute-Based Architectural Styles (ABASs) [7] is an architectural style in which the constraints focus on component types and patterns of interaction that are particularly relevant to quality attributes. ABASs aid architecture evaluation by focusing the stakeholders' attention on the patterns that dominate the architecture, by suggesting *attribute-specific questions* associated with the style. Such questions are, in turn, inspired by the *attribute characterisations*. Each attribute characterisation is divided into three categories: *external stimuli*, *architectural parameters*, and *responses*. With regards to the *availability* attribute, its *stimuli* are from *source* (i.e. hardware or software faults), and *type* (i.e. value, timing, and stopping). Its *parameters* are the *hardware redundancy*, *software redundancy*, *voting*, *retry*, and *failover*. Finally, the *responses* generated are *availability*, *reliability*, *levels of service*, and *mean time to failure*. Also, the *stimuli* of the *performance* attribute are *mode*, *source*, and *frequency regularity*. The performance *parameters* considered in architectural decisions are mainly *resource* such as *CPUs*, *sensors*, *networks*, *memories*, *actuators*, etc. and *resource arbitration* in the form of *queuing* and *pre-emption*. Finally, the *responses* from the performance characterisation are *latency*, *throughput*, and *precedence*.

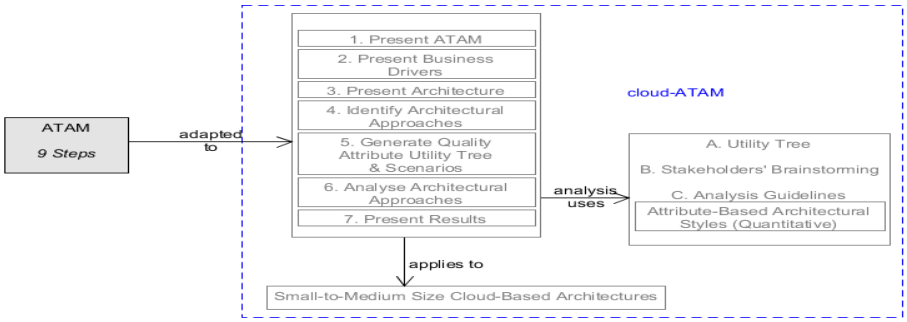

Fig. 1. *cloud-ATAM*: Adapted ATAM with Defined 3-Step Analysis Approach

3 cloud-ATAM

The *cloud-ATAM* (see Figure 1) is motivated by the complex and iterative nature of the ATAM even for small-to-medium size architectures (classed with ISO/IEC 14143:1998 & COSMIC Full FP 2.2). Typically, such architectures do not need to undertake all the steps of the ATAM. Due to the size of such projects, some steps can be combined into a new step, and some activities of some steps can be

optional. Here, *generating the quality attribute utility tree* process of *Step 5* can be combined with the *prioritising scenarios* process of *Step 7* in ATAM. Also, the *analysing the architectural approaches* process of *Step 6 and 8* are repeated, and can be extended with *noting the impact of scenarios on the architectural approaches* of *Step 8*. These changes are particularly important especially in addressing the perceived weakness of ATAM due to its iterative nature which requires a substantial number of human experts on the team at different times [3]. It is often expensive to speculate the availability of such domain experts for such projects due to budgetary or time constraints [3]. Here, we adapt ATAM into a seven (7)-step methodology (see Figure 1). The analysis of the architecture is undertaken in two phases: Phase 1 (Steps 1-5) and Phase 2 (Steps 6-7). **Phase 1** is *architect-centric* and concentrates on eliciting and analysing architectural information. Here, the *cloud-ATAM* uses a non-trivial set of *scenarios* to analyse the cloud-based architecture. **Phase 2** is *stakeholder-centric* and elicits points of view from a more diverse and larger group of stakeholders, and verifies, and then builds on the results of the first phase.

cloud-ATAM presents an *enrichment* in terms of coverage (i.e. unpredictable behaviour) to ATAM in the form of a three-step scenario-based analysis approach: (1) Utility Tree, (2) Stakeholders' Brainstorming, and (3) *guideline* that utilises ABAS to *quantitatively* reason about quality attributes (see Figure 1).

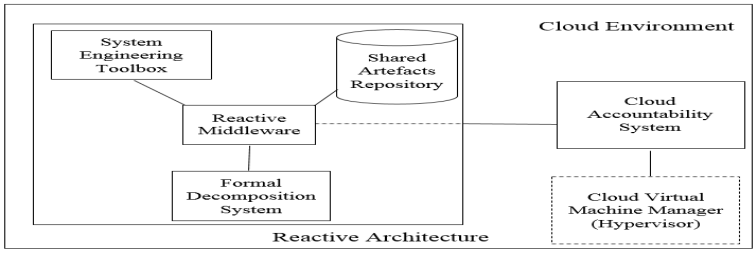

Fig. 2. Overview of Reactive Architecture

3.1 Reactive Architecture for *cloud-ATAM* Analysis

The designed set of scenarios is presented based on a Reactive Architecture (RA) [6] (see Figure 2). The RA is for *global software development* (GSD), where the cloud is the best facilitating environment. This RA is composed of systems such as the Reactive Middleware (RM), the Cloud Accountability System (CAS), the Formal Decomposition System (FDS), the Shared Artefacts Repository (SAR), and the System Engineering Toolbox (SET). These systems are designed as web services (WSs). Web services are necessary to support the deployment and operation on a cloud platform. The RA stands to benefit from the scalability, cost-effectiveness, flexibility, multi-user access, etc. provided by cloud computing. The collective mission of the mentioned systems is to facilitate *artefact-driven* and *role-based* support for cloud-based GSD. To this end, the RM which plays

a central role in the RA provides cloud-based services towards *change management* and *traceability* in projects involving distributed teams. The RM aims to assist system engineers to manage changes and trace the cause-and-effect of these changes on artefacts created or used in the various system engineering processes.

Table 1. Mapping Requirements to Quality Attributes of Reactive Architecture

Attribute Goals	**ID**	**Attribute-Specific Requirements**
Operability	O1	The Reactive Architecture must store all artefacts created in the composing systems.
	O2*	It must monitor and trace all changes to these artefacts to inform system stakeholders. (also P1)
	O3	The System Engineering Toolbox must facilitate sequential and parallel execution of tools in a workflow manner.
	O4	The Formal Decomposition System must provided a high capacity and dedicated channel to coordinate real-time analysis on artefacts for local client computers and on remote cloud environment. (also P3)
	O5	The Cloud Accountability System must gather dependability metrics from several virtual machines, and perform a synchronous analysis of these metrics.
Performance	P1*	It must monitor and trace all changes to these artefacts to inform system stakeholders. (also O2)
	P2*	The Reactive Middleware must enable heterogeneous access and analysis operations on artefacts in the Shared Artefacts Repository. (also A2)
	P3	The Formal Decomposition System must provided a high capacity and dedicated channel to coordinate real-time analysis on artefacts for local client computers and on remote cloud environment. (also O4)
	P4	Security mechanisms must not degrade defined performance threshold. Specifically, response time for create, delete, update, and display artefact/data operations should not exceed 5 seconds at peak cloud (i.e. architecture) period and less than 1 second during off-peak period. (also S1)
	P5	The Reactive Architecture must do all this while meeting the performance and availability requirements to allow it to keep up with the sturdy stream of data and operations on artefacts from the system engineering processes. (also A4)
Scalability	Sc1	The Reactive Architecture must support multiple users concurrently.
	Sc2*	The Reactive Architecture must provide capacity to scale quickly to accommodate changing demands of system developers, and failures. (also A1)
Availability	A1*	The Reactive Architecture must provide capacity to scale quickly to accommodate changing demands of system developers, and failures. (also Sc2)
	A2*	The Reactive Middleware must enable heterogeneous access and analysis operations on artefacts in the Shared Artefacts Repository. (also P2)
	A3	Critical systems such as the Reactive Middleware must not constitute a single point of failure which will affect the uptime of the system and the architecture. (also R1)
	A4	The Reactive Architecture must do all this while meeting the performance and availability requirements to allow it to keep up with the sturdy stream of data and operations on artefacts from the system engineering processes. (also P5)
Maintainability	M1	The Shared Artefacts Repository must be backed up asynchronously to facilitate roll-back of repository artefacts.
Reliability	R1	Critical systems such as the Reactive Middleware must not constitute a single point of failure which will affect the uptime of the system and the architecture. (also A3)
Security	S1	Security mechanisms must not degrade defined performance threshold. Specifically, response time for create, delete, update, and display artefact/data operations should not exceed 5 seconds at peak cloud (i.e. architecture) period and less than 1 second during off-peak period. (also P4)

4 Evaluating The Reactive Architecture

Here, we analyse the cloud-based Reactive Architecture using the scenario-based *utility tree* approach of the three-step *cloud-ATAM* analysis (see Figure 1).

4.1 Present the ATAM

The ATAM has been introduced in Section 2.1, and *cloud-ATAM* in Section 3.

4.2 Present Business Drivers

We briefly present the business drivers (i.e. requirements) of the Reactive Architecture (RA) which cover several quality attributes: operability - **[O]**, performance - **[P]**, scalability - **[Sc]**, availability - **[A]**, maintainability - **[M]**, reliability - **[R]**, and security - **[S]** (see Table 1). We focus on the *availability* (**A1, A2, A3, A4, A5**) and *performance* (**P1, P2, P3, P4, P5**) related requirements.

4.3 Present the Architecture

The Reactive Architecture [6] has been introduced in Section 3.1.

Table 2. Classified Attribute-Specific Questions

Attribute-Specific Questions ID	Attribute-Specific Questions
ASQ1	What facilities exist in the software architecture (if any) for self-testing and monitoring of software components? (Availability)
ASQ2	What facilities exist in the software architecture (if any) for redundancy, liveness monitoring, and fail-over? (Availability)
ASQ3	How is data consistency maintained so that one component can take over from another and be sure that it is in a consistent state with the failed component? (Availability)
ASQ4	What is the process and/or task view of the system, including mapping of these processes/tasks to hardware and communication mechanisms between them? (Performance)
ASQ5	What functional dependencies exist among the software components? (Performance)
ASQ6	What data is kept in the database? How big is it, how much does it change, who reads/writes it? (Performance)
ASQ7	How are resources allocated to service requests? (Performance)
ASQ8	What are the anticipated frequency and volume of data transmitted among the system components? (Performance)

4.4 Identify Architectural Approaches

Here, some *attribute-specific questions* (see Table 2) are asked to draw attention to the patterns that dominate the Reactive Architecture. We also identify some *architectural approaches* in Table 3 which inspires the presented questions.

Table 3. Architectural Approaches for the Reactive Architecture

Architectural Approach ID	Architectural Approaches
AD1	We use the *component-and-connector architectural style* to represent the various components and connections/interfaces of the Reactive Architecture. This is particularly relevant because it expresses the runtime behaviour of the architecture under review. Also, interfaces are defined as *application programming interfaces* (APIs).
AD2	We avoid the *distributed data repository* approach in designing the Shared Artefacts Repository. This prevents situations such as issues with database consistency and possible modifiability concerns.
AD3	The *client-server* approach is a best fit for the data-centric Shared Artefacts Repository system.
AD4	The Reactive Middleware will be adequately represented using the *client-server* approach.
AD5 **AD6**	Since the Reactive Middleware and the Shared Artefacts Repository constitute a single point of failure, we present the following approaches: - *Backup* of artefacts in the Shared Artefacts Repository. - *Distributed services* (or modular set of services) for the components of the Reactive Middleware.
AD7	*Schema-free NoSQL data management system* (DMS) is necessary for the Shared Artefacts Repository to minimise or remove bottlenecks.
AD8	An *independent communication components* approach for communication among the Reactive Middleware, Shared Artefacts Repository, Cloud Accountability System, and the Formal Decomposition System. Such communication approach is particularly relevant for the distributed components of the Cloud Accountability System.

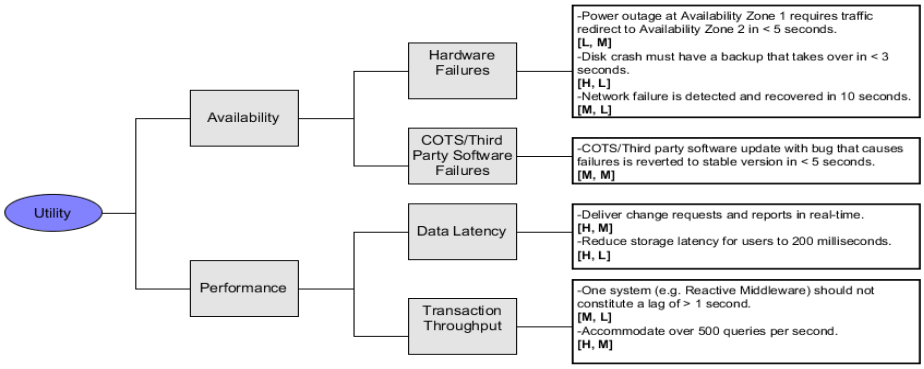

Fig. 3. Attribute Utility Tree of Reactive Architecture (adapted from [9])

4.5 Generate the Quality Attribute Utility Tree and Scenarios

At this point, we identify, prioritise, and refine the most important quality attribute goals in a utility tree format (i.e. Figure 3). We also created *scenarios*

that are used to precisely elicit the specific quality goals against which the architecture is analysed. Also, the quality attribute scenarios are prioritised based on (1) *how important they are to the overall mission of the architecture*, and (2) *the perceived difficulty in realising them in the architecture* (see Table 4).

Table 4. Prioritised Quality Attribute Scenarios (Ordered)

No.	Quality Attribute Scenarios	Scenario ID	Numbered Value
1	Disk (i.e. data repository) crash must have a back-up that takes over in less than 3 seconds	A2	1
2	**Deliver change requests and reports in real-time**	**P1**	**1**
3	Reduce storage latency for users to 200 milliseconds	P2	**1**
4	Accommodate over 500 queries per second	P4	**1**
5	Network failure is detected and recovered in 10 seconds	A3	**2**
6	COTS/Third party software update with bug that causes failures is reverted to stable version in less than 5 seconds	A4	**2**
7	One system (e.g. Reactive Middleware) should not constitute a lag greater than 1 second	P3	**2**
8	Power outage at *Availability Zone 1** requires traffic redirect to *Availability Zone 2** in less than 5 seconds	A1	**3**

Table 5. Analysis of Sensitivities, Trade-offs, Risks & Non-Risks for the Utility Tree

Sensitivities:	* **S1:** Concern over network latency. * **S2:** Using a data-centric and client-server approach for the central repository can facilitate data integrity and consistency, but it makes the architecture sensitive to its faults and bottlenecks. * **S3:** Similarly, the central role played by the Reactive Middleware makes the architecture sensitive to faults, resource (i.e. CPU, memory) malfunctions or unavailability.
Trade-offs:	* **T1:** Availability (+) vrs Performance (-) vrs Reliability (-): defining a central artefacts repository makes artefacts readily available, but may be faced with bottlenecks when there are a burst of queries on the repository. * **T2:** Availability (+) vrs Performance (+): using APIs for component interfaces facilitate readily access to resources, and boosts performance. * **T3:** Availability (+) vrs Performance (-): client-server approach for the Reactive Middleware allows for multi-client service, but there can be an overwhelming network management performance constraint. * **T4:** Availability (+) vrs Performance (-) vrs Reliability (+): backing up the artefacts in the primary Shared Artefacts Repository allows for fail-over assurance and increased reliability, but the asynchronous back-up process can affect performance.
Risks:	* **R1:** Data integrity. * **R2:** The risk is that the Reactive Middleware and the Shared Artefacts Repository constitute a single point of failure.
Non-Risks:	* **N1:** The non-risk is the use of application programming interface (API) approach which should stay compatible. * **N2:** The independent communication connections should enable real-time data transfer.

Table 6. Analysis of Performance Scenario - **P1** - (see Table 5 for the description of S1, S2, T1, etc.) and (**C&C + API**: Component-and-connector architectural style and API, **SAR**: Shared Artefacts Repository, **RM**: Reactive Middleware, and **ICC**: Independent Communication Components)

Analysis of Architectural Approach using a Performance-related Scenario				
Scenario ID : *P1*	**Scenario:** *Deliver change requests and reports in real-time*			
Attribute(s)	*Performance (also Availability)*			
Environment	*Normal Operations*			
Stimulus	*Responsiveness to change events*			
Response	*real-time*			
Architectural Decisions	**Sensitivity**	**Trade-off**	**Risk**	**Non-Risk**
AD1 C&C + API	S1			N1
AD2				
AD3 Client-Server SAR	S2	**T1**	R1, R2	N1
AD4 Client-Server RM	S3	**T3**	R2	N1
AD5 Back-up	S1,S2	**T4**		N1
AD6 DS RM	S1		R1	N1
AD7 Schema-free-SAR			R2	
AD8 ICC	S1			N2

4.6 Analyse the Architectural Approaches

We probed the *architectural approaches* in light of the quality attributes and identified *risks*, *non-risks*, and *trade-offs* using *quality attribute questions*, while noting the impact of each scenario on the *architectural approaches* (see Table 5). In this paper, we only analyse *scenario P1*, and this is shown in Table 6.

4.7 Present Results

The *cloud-ATAM* delivers the main *products*: sensitivities, trade-offs, and architectural risks in Table 5. From Table 6, the *cloud-ATAM* completed a full cycle by linking the *architectural decisions* to the *quality attributes* (i.e. availability, performance), and back to the *business goals* of the Reactive Architecture.

5 Conclusion

In this paper, we have motivated the need for architecture evaluation methods suitable for the dynamic unpredictable cloud environments. In particular, we have presented an evaluation method - *cloud-ATAM* - derived from ATAM for evaluating the *availability* and *performance* quality attributes of a cloud-based Reactive Architecture. The results from Tables 5 and 6 indicate that the *cloud-ATAM* found some trade-offs (i.e. T1, T3, T4). This answers our **research question**, and validates our hypothesis that *the cloud-ATAM is able to identify trade-offs between the availability and performance quality attributes for the Reactive Architecture*. A detailed discussion of the evaluation *products* is limited.

References

1. N. Josuttis, *SOA in Practice: The Art of Distributed System Design*. O'Reilly Media, Inc., 2007.
2. P. Szwed, P. Skrzynski, G. Rogus, and J. Werewka, "Ontology of architectural decisions supporting ATAM based assessment of SOA architectures," in *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, Sept 2013, pp. 287–290.
3. F. Faniyi, R. Bahsoon, A. Evans, and R. Kazman, "Evaluating security properties of architectures in unpredictable environments: A case for cloud," in *Software Architecture (WICSA), 2011 9th Working IEEE/IFIP Conference on*, June 2011, pp. 127–136.
4. D. Ardagna, "Cloud and multi-cloud computing: Current challenges and future applications," in *Principles of Engineering Service-Oriented and Cloud Systems (PESOS), 2015 IEEE/ACM 7th International Workshop on*, May 2015, pp. 1–2.
5. R. Kazman, M. Klein, P. Clements, N. L. Compton, and L. Col, "ATAM: Method for Architecture Evaluation," 2000.
6. D. Adjepon-Yamoah, A. Romanovsky, and A. Iliasov, "A reactive architecture for cloud-based system engineering," in *Proceedings of the 2015 International Conference on Software and System Process*, ser. ICSSP 2015. Tallinn, Estonia: ACM, 2015, pp. 77–81.
7. M. Klein, R. Kazman, L. Bass, J. Carriere, M. Barbacci, and H. Lipson, "Attribute-based architecture styles," in *Software Architecture*, ser. IFIP The International Federation for Information Processing, P. Donohoe, Ed. Springer US, 1999, vol. 12, pp. 225–243.
8. *Evaluating Software Architectures: Methods and Case Studies*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.
9. P. Clements, R. Kazman, and M. Klein, *Evaluating software architectures : methods and case studies*, ser. SEI series in software engineering. Boston, San Francisco, Paris: Addison-Wesley, 2002.