

Scalability evaluation of a distributed agent system

This article has been downloaded from IOPscience. Please scroll down to see the full text article.

1999 Distrib. Syst. Engng. 6 129

(<http://iopscience.iop.org/0967-1846/6/4/302>)

View [the table of contents for this issue](#), or go to the [journal homepage](#) for more

Download details:

IP Address: 38.107.179.211

The article was downloaded on 15/02/2012 at 15:06

Please note that [terms and conditions apply](#).

Scalability evaluation of a distributed agent system

Anne-Louise Burness, Richard Titmuss, Caroline Lebre,
Katie Brown and Alan Brookland

BT Adastral Park, Martlesham Heath, Suffolk IP5 3RE, UK

E-mail: louise@b29net.bt.co.uk

Received 26 July 1999

Abstract. The use of new computing paradigms is intended to ease the design of complex systems. However, the non-functional aspects of a system, including performance, reliability and scalability, remain significant issues. It is hard to detect and correct many scalability problems through system testing alone—especially when the problems are rooted in the higher levels of the system design. Late corrections to the system can have serious implications for the clarity of the design and code.

We have analysed the design of a system of multiple near-identical, ‘reactive’ agents for scalability. We believe that the approach taken is readily applicable to many object oriented systems, and may form the basis of a rigorous design methodology. It is a simple, yet scientific extension to current design techniques using message sequence charts, enabling design options to be compared quantitatively rather than qualitatively. Our experience suggests that such analysis should be used to consider the effect of artificial intelligence, to ensure that autonomous behaviour has an overall beneficial effect for system performance.

1. Introduction

Telecommunications engineers are turning to distributed computing [1], artificial intelligence [2], and agent technologies [3] to develop future services, expecting that it is economically justified to use large numbers of inexpensive processors [4] to provide these services. Limiting the scope of each component should limit the complexity of the computation, producing simpler, more reliable processing units. Artificial intelligence will provide adaptability within unpredictable environments [2], and many hope that such systems will overcome performance and scalability problems [5]. However, no hard evidence currently exists to support this claim. Indeed, much analysis identifies possible performance problems, from a simple worsening of performance [6], to chaotic behaviour [7]. We have not read any specific analytic scalability studies on AI and agent systems for telecommunications systems; therefore, we have studied an in-house distributed agents system. Although this system is a complex mixture of reactive agents [8] and distributed objects [9], we were able to discover that many scalability problems within this system were identifiable early in system design. This system is a successor to an earlier deliberative system [10]. However, as in [6, 11], many of the adaptive features gave limited benefit with a large performance penalty.

2. Review of previous studies

To be successful, future services need to meet both user expectations of performance, and supplier expectations of

scalability. We define performance as:

- The response time seen by a user under normal working conditions.
- The cost of the system per user; a representation of the hardware requirements of the system.

We consider scalability to be a measure of how these change as the system size is increased—for example, by adding more users.

Performance improvements of software systems are often made through testing and modification in the final stages of development. However, it is hard and expensive to correct problems at this stage, particularly if the problem lies with the system design [12, 1]. System architectures are also assessed through implementation and measurement [13], or simulation [14]. This is time consuming—especially for telecommunications systems with tens of thousands of nodes.

Analysing test results is difficult. For example, in [14] the authors fail to prove the scalability of their system (though our analysis suggests their system is indeed scalable!). They report simply that key parameters show only a small increase as the number of clients increases. If this increase is small, but exponential, problems could occur as the system was scaled to sizes greater than those simulated—a potential situation in telecommunications where service take-up can be unexpectedly high [15].

To express scalability, [16] proposes a scalability metric for distributed object systems: $\lambda/(\tau \times \text{Cost})$ where λ is the system throughput and τ the response time. They consider that a system is scalable between two configurations

(for example, two to a thousand nodes) if the ratio of this metric for the two configurations is greater or equal to one. This encompasses the logical and physical aspects within a single metric—an oversimplification which does not aid understanding of a system, confusing the effects of both architecture and implementation. Such confusion is common: in [13], one architecture is claimed to be less scalable than another, although this is because synchronous communication was used without implementing a multi-threaded server.

Analytic techniques for performance and scalability prediction are currently so limited that such analysis is often only undertaken if problems cannot be corrected by any other means [17]. Many techniques rely on an estimation of the instruction count of every function in the system [18, 1]—essentially impossible to obtain from the design. Also, studies typically represent all elements of a system, impractical for future telecommunications services with millions of agents. The group concept [1], which represents a set of client processes with identical time-averaged behaviour as a single process, claims to address this issue, although, like most current analysis techniques, this assumes a client-server object architecture, rather than peer-to-peer relationships and agent architectures.

3. The Agents platform

The Agents system studied provided visitors to an exhibition with information customized to their location, personal preferences and terminal. Users entering the exhibition chose from a selection of terminals, including a portable computer connected to a wireless LAN, and then customized the system—requesting, say, technical information in French. As the user approached different stands, they were detected (through an infrared transmitter) and customized information about that exhibit was delivered to them in a format suitable for their terminal—for example, by removing pictures before delivering the information to a text-only display.

The system was implemented in Java for rapid development and platform independence (Sun Solaris, IBM AIX and Windows-NT machines have all been used). The agent distribution was supported by a CORBA platform, VisiBroker for Java [19].

Figure 1 shows the basic elements of the system. Figure 2 is the message sequence chart for the following scenario—a near-minimal system, with two Information Agents and only one user. When a person is using the system, their location is monitored by a physical mechanism. This leads to the user's Personal Agent being informed of any change of location (message 1). The Personal Agent then cancels contact with Information Agents who are no longer relevant to the user's new location, and queries the Location Directory to discover which information sources are now relevant. The Personal Agent may then contact these Information Agents to discover more about the information available (message 6). After comparing the available information with the user's preferences, the Personal Agent may direct a suitable Terminal Agent to obtain a particular document and present it to the user (message 8). It is the responsibility

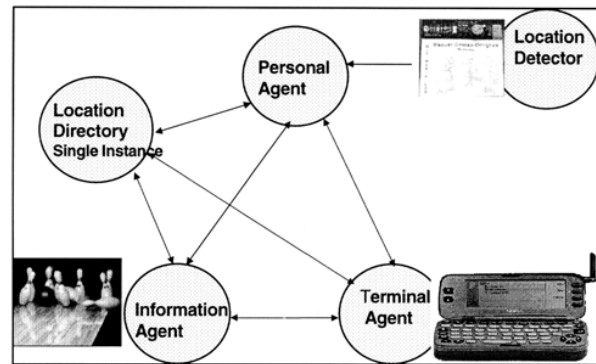


Figure 1. Basic elements of the Helpful Agents system.

of the Terminal Agent to obtain the document and present it in a format suitable for the terminal.

One of the key aspects of the system is that it is made up from potentially many thousands of instances of a small number of agent types. The system is scaled if the number of users (each with a Personal Agent and one or more terminals and corresponding Terminal Agents) or information providers (Information Agents) increases. It is likely that increases in one type of customer will lead to increases in the other. Additionally, at the user's discretion, Personal Agents may also act as Information Agents. Thus we have identified the variables that drive changes and describe the scale of the system.

4. Performance and scalability analysis

4.1. System design

A series of tests on the system led to a number of performance optimizations as illustrated in figure 3, which shows the response time for the interaction of figure 2. Measurement techniques included Java profiling, writing time stamps into the code, use of the ORB interceptors [19] and operating system level measurements. This improvement was qualitatively experienced by users testing the system. As indicated, many improvements to performance were made by low-level, implementation modifications—high-level analysis will not remove the need for such testing.

Scalability tests were carried out by gradually increasing the total number of agents in the system. Catastrophic problems occurred when the system was run with over 120 agents. The cause of these problems could not be readily identified, nor predicted from the smaller system measurements. It was difficult to say if the problem was a result of having limited hardware, or was more fundamental to the system design. Reasons for this difficulty include:

- (1) noise in the system (primarily caused by competition for hardware resource) increases as the system scaled;
- (2) a huge quantity of data is generated;
- (3) events cannot be readily correlated to each other, unless there is strong synchronization between machines, and strict message ordering;
- (4) many changes occur at the same time and may affect the same data.

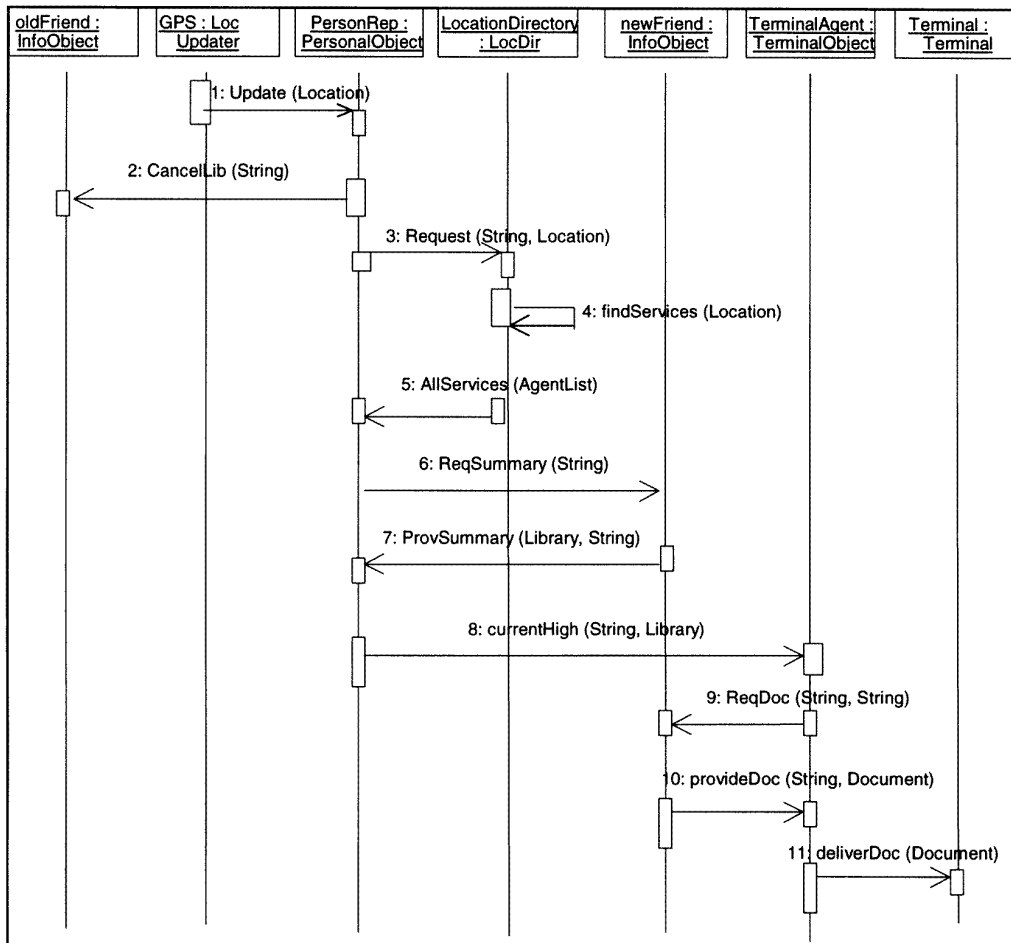


Figure 2. Example sequence chart for the Helpful Agents platform.

We therefore decided to try and see what scalability problems we could identify through analysis of the system design. Implementation and measurement knowledge was used solely to verify the analysis.

For initial analysis we treat each instance of every agent type as statistically identical. We begin with a minimal number of agent instances within the system. As in [18, 20], we draw primary interaction sequences (those which would execute most frequently, or with the most stringent performance requirements)—see figure 2. We then looked at the implications of varying the number of instances of each agent type. By examining how the sequence charts evolve, we can then generate equations explaining how specific parameters change as the system is scaled—avoiding the need to explicitly represent a very large system and enabling quantitative comparison of different designs.

The following sections give examples of a study of the implications of figure 3. A full system analysis cannot be shown due to space limitations.

4.2. Communications

Network communication is often the key performance limitation. Although the networking performance of the system depends upon the physical architecture of the system, a big message should take longer to deliver than a short

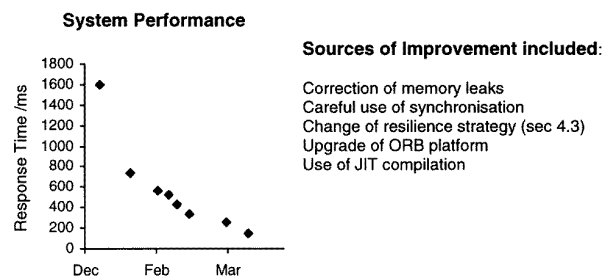


Figure 3. Improvement in system response time during testing.

one, and network requirements will approximately double if the number of messages is doubled. (When underlying networking functionality breaks this simple picture, this should be represented as another layer in the model; use of layers facilitates understanding of complex systems.)

One design debate concerned the way Personal Agents discover Information Agents. When a user moves, the Personal Agent sends a message to the Location Directory requesting a list of Information Agents associated with a particular location (figure 2, messages 3 and 5). The Personal Agent then contacts each Information Agent to obtain summary information (figure 2, messages 6 and 7) to decide which Information Agents may be of interest to

Number request/ reply to directory	α	Number Personal Agents
size(request) (message 3)	=	constant
size(reply) (message 4)	α	Number Information Agents
size(ReqSummary) (message 6)	α	Number (Information Agents * Personal Agents)
size(request)	=	constant
size(ProvSummary) (message 7)	α	size of Information Agent summary information

Figure 4. Estimation of the number and size of messages needed to discover interesting information sources.

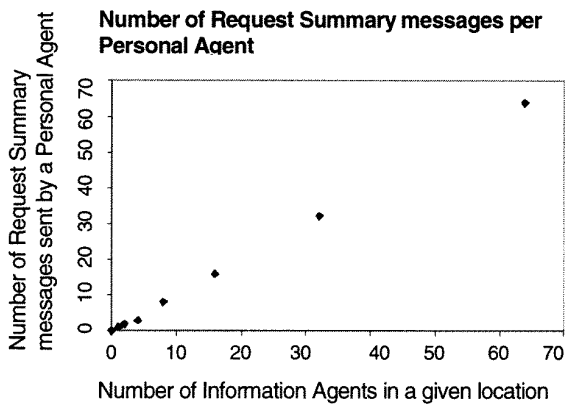


Figure 5. Number of request summary messages per Personal Agent.

the user. This approach protects user privacy by revealing minimal information to the system at large, and minimizes the workload on the location directory, at the expense of an extra communication overhead which can be explicitly calculated (figure 4).

We see that the number of ReqSummary/ProvSummary messages (figure 2, messages 6 and 7) which obtain summary information, increase proportionally with the number of Personal Agents multiplied by the number of Information Agents. Given the likely correlation between the number of Information and Personal Agents, the effect is to cause rapid network congestion—an effect confirmed by measurement (figure 5).

With measurements alone, it is difficult to identify the source of such congestion. This is partly because (figure 2, message 10) the message that delivers a document to the Terminal Agent dominates the small-system network traffic, as it is typically 20 times larger than other messages. The size of this message does not change with the size of the system, and the frequency of occurrence of this message increases simply with the number of Personal Agents in the system, so it would not have a disproportionately large impact on the system scalability.

Figure 4 also shows that it is possible to estimate the size and scalability of individual messages. For example, the message to request information from the Location Directory (figure 2, message 3) will be unaffected by the size of the system, however the list of local information sources that the Location Directory returns to the Personal Agent (figure 2, message 4) will increase as the density of Information Agents

increases. Thus, the total volume of traffic associated with this reply (number of messages \times size of message) grows again proportionally to the number of Personal Agents multiplied by the number of Information Agents (again verified through measurements).

Whilst there may be many messages to assess, the time taken to assess any particular message is short, and the good correlation between predictions and measurements suggests that the effort is worthwhile. As system development progresses, it is necessary to ensure that later choices, for example relating to security, do not alter the analysis. More detailed analysis [18] will reveal the impact of physical architecture and communication platform.

4.3. Data IO

Data IO is also time consuming with each IO operation taking a substantial amount of time relative to the instruction execution speed: a ratio of 1:200 000 is quoted in [12]. Therefore, it is worth explicitly representing and analysing all data IO. This could have helped avoid a problem that was discovered during testing of the system.

The amount of time it took a Personal Agent to decide to send a Request message (figure 2, message 3) to the Location Directory seemed exceptionally long. The delay was caused by the agent saving its state (to provide resilience) when it discovered the user had moved. However, changes to an agent occur when a user is interacting with the agent, so the agent was saving its state (halting processing to ensure consistency) at its busiest time. By removing this operation and providing a suitable failure recovery mechanism the average response time was cut by 100 ms to 340 ms.

4.4. Computational requirements

It is relatively easy to identify potential problems associated with network traffic and data IO. It is much more difficult to predict how much computational effort or memory will be required for any particular function as any high-level function can be implemented in a wide variety of ways. That said, it should still be possible to identify how any function might scale at best. The difficulty remains in assessing the likely seriousness of these problems.

As a simple example of such an analysis, we consider the Location Directory. This maintains a record of the location of all Information Agents. Thus, we expect that the memory requirements of the directory will grow linearly

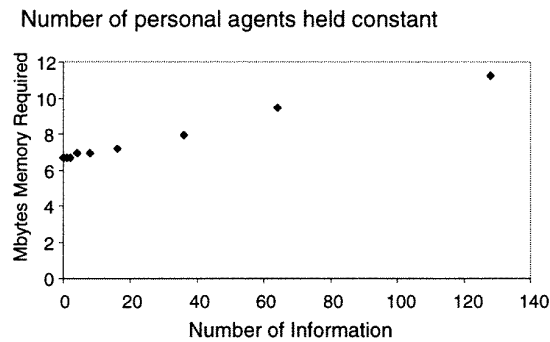
Growth in memory requirements

Figure 6. Memory requirement for location directory.

with the number of Information Agents in the system—confirmed through measurements (figure 6). The large memory requirements are primarily due to the footprint of the Java virtual machine.

Although we can identify the linear trend, we have not yet been able to predict the gradient of this line. By identifying the information that the directory needs to store and its type (e.g. double) we estimated a minimum gradient of 5 kbytes/Information Agent. The actual gradient is 37 kbytes/Information Agent! However, by correctly identifying a linear trend, only a small number of measurements will enable us to predict hardware requirements.

Similarly, we may expect that the time taken for the Location Directory to find which Information Agents are local to a particular Personal Agent will increase linearly as the number of Information Agents in the system is increased—a loop compares the location of the Personal Agent with that of every Information Agent (again seen in measurements).

During the above analysis and measurements there was only one Personal Agent in the system to eliminate any competition for Location Directory resources. We can model the effects of competition with a simple queuing model [21]. Queuing models calculate the average delay in a system based on the ratio of server response time for a single request and the inter-arrival time between requests, both of which are simply assumed to have a Poisson distribution. Early in the design process we do not know these times. However, we do know that the rate of requests on the Location Directory is proportional to the number of Personal Agents and the response time of the service is proportional to the number of Information Agents. However, the same factors are driving the growth of both of these elements. A simple queuing model was made to investigate the effect of this, as shown in figure 7.

Thus, we can see that an unexpectedly early degradation of the Location Directory may occur, even if small-scale system performance appears as expected. More complete analysis of the system suggests that this is the most likely explanation for the unexpected failure of the system. Because the Location Directory was running on a machine which was also supporting a number of other agents, further hardware interactions occurred, with a rapid knock-on effect throughout the whole system.

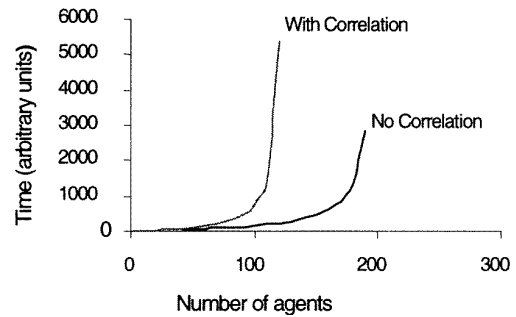
Response Time against number of agents

Figure 7. Queuing model results showing the maximum capacity of a system where the server rates and job requests are correlated (approximately 110 agents), compared with a simple queuing system (approximately 180 agents), where the response time tends to infinity as the hardware limits of the machine are approached. Numbers are included to illustrate the effect: they were not estimated from the design.

5. Discussion and conclusions

Our approach to assessment of scalability of a distributed system with many similar agents is:

- (i) identify the variables which drive changes and describe the scale of the system;
- (ii) draw the small system message sequence charts;
- (iii) use the previously identified variables to draw the equivalent charts for a slightly larger system;
- (iv) use the pairs of charts to identify the changes in network communications, data IO, memory and computation;
- (v) develop the equations that show how the different parameters will change with respect to the scaling variables.

We have demonstrated this technique, identifying potential scalability problems from an analysis of a high-level design. Neither detailed knowledge of the final class hierarchy, nor instruction counts for individual methods were needed. Most notably, this method correctly identified the element of the system that caused the first major scalability problem and explained why this component failed much sooner than anticipated. We were also able to identify and understand problems that only arose if several changes occurred simultaneously.

Whilst we have shown that it is relatively simple to quantitatively assess designs for network or data IO scalability, improvements in the estimation of memory and computational requirements of a process are still required. Currently, if several elements have poor scaling characteristics, perhaps each associated with different design alternatives, it remains difficult to differentiate between them.

The method enables quantitative assessment of a system design, ensuring that the design can be verified before much time and money has been spent on development. This reduces the need for late alterations which may be difficult, costly and have a detrimental effect on the longer term maintainability of the code.

The analysis also enables us to think more clearly about the adaptability that may be needed in such an agent system.

References

- [1] Raguideau, Naruyama and Kubota 1994 Telecommunication service software architecture for next-generation networks *IEICE Trans. Commun.* Vol E77-B, No 11
- [2] Kumar and Venkataram 1997 Artificial intelligence approaches to network management: recent advances and a survey *Comput. Commun.* **20** 1313–22
- [3] Schoonderwoerd, Holland and Bruten 1997 Ant-like agents for load balancing in telecommunications networks *Proc. 1st Int. Conf. on Autonomous Agents* (New York: ACM)
- [4] Jennings 1993 Commitments and conventions: the foundation of co-ordination in multi-agent systems *Knowl. Eng. Rev.* **8** 223–50
- [5] Wiczorek and Albayrak 1998 Open scalable agent architectures for telecommunications applications *2nd Int. Workshop on Intelligent Agents for Telecommunication Applications* (Berlin: Springer) pp 233–49
- [6] Markovitch and Scott 1993 Information filtering: selection mechanisms in learning systems *Mach. Learn.* **10** 113–51
- [7] Huberman 1988 *The Ecology of Computation 1988* (Amsterdam: North-Holland)
- [8] Chapman and Agre 1986 Abstract reasoning as emergent from concrete activity *Proc. Workshop on Reasoning About Actions and Plans* (San Mateo, CA: Morgan Kaufmann)
- [9] Briot and Gasser 1992 Object based concurrent programming and distributed artificial intelligence *Distributed Artificial Intelligence: Theory and Practice* (Dordrecht: Kluwer)
- [10] Titmuss, Crabtree and Winter, Agents mobility and multimedia information *Software Agents and Soft Computing—Towards Enhancing Machine Intelligence* (Berlin: Springer)
- [11] Brooks 1985 A robust layered control system for a mobile robot *MIT AI Memo* 864
- [12] Smith C 1990 *Performance Engineering of Software Systems* (Reading, MA: Addison-Wesley)
- [13] Abdul-Fatah and Majumdar 1998 The effect of object-agent interactions on the performance of Corba systems *IEEE Int. Conf. Performance Computing and Communications* (Los Alamitos, CA: IEEE Computer Society Press)
- [14] Bernardo and Pinto 1998 Scalable service deployment on highly populated networks *2nd Int. Workshop on Intelligent Agents For Telecommunication Applications* (Berlin: Springer) pp 29–44
- [15] 1996 *Electronic Telegraph* UK News May 22
- [16] Jogalekar and Woodside 1998 Evaluating the scalability of distributed systems *Proc. 31st Hawaii Int. Conf. on System Sciences* vol 31 (Los Alamitos, CA: IEEE Computer Society Press)
http://www.sce.carleton.ca/faculty/woodside_pub.shtml
- [17] Bertke D 1995 System development risk reduction using a system performance model *Proc. IEEE National Aerospace and Electronics Conf.* (New York: IEEE)
- [18] Durrant and Civello 1998 Performance driven allocation of objects to processor nodes in a distributed system *IEE Proc. Softw.* **145**
- [19] Information on Visibroker Orb available from webpage <http://www.inprise.com/techpubs/visibroker/visibroker33/>
- [20] Faltin and Lambert 1997 An annotational extension of message sequence charts to support performance *Proc. SDL '97* (Amsterdam: Elsevier) pp 307–22
- [21] Kleinrock 1975 *Queueing Systems* vols 1 and 2 (New York: Wiley)