# UTILITY COMPUTING SIMULATION

Benjamin Heckmann
Ingo Stengel
Günter Turetschek
University of Applied Sciences Darmstadt
Haardtring 100
D-64295 Darmstadt, Germany
E-mail: benjamin.heckmann@gmx.de

Andy Phippen
University of Plymouth
Room 405a, Cookworthy Building, Drake Circus
Plymouth, Devon, PL4 8AA, UK

## KEYWORDS

SaaS, Cloud Computing, SOA, Service Billing, Service Provision, QoS

## ABSTRACT

Utility Computing (UC) misses an explicit definition of the core relation between IT resource utilisation, its total costs and service prices. Additionally, the implications of complex usage scenarios occurring in UC have not been examined for the service operations lifecycle. Missing those, UC service offers fail in: prediction of resource utilisation and dependent operational costs prediction, calculation of subsequent price scales, and subsequent runtime gross price calculations.

In this paper a strategy to handle UC's complexity proposing a simulation model to support each step in the service operations lifecycle is presented. The implementation approach for the model is based on OMNeT++. First simulation outcomes are presented.

## INTRODUCTION

This paper starts with a short definition of the term Utility Computing as a business model. Afterwards a common Service Operations Lifecycle is defined that has been derived from ITIL. After setting the context, the research objectives and the related research approach are introduced. Subsequent the evolved strategy that is able to handle UC's complexity is outlined. This strategy includes the demand for an UC provisioning model and a corresponding simulation model. Both models and the implementation of the simulation of the UC provisioning model are introduced. First outcomes of simulation runs are shown. Corresponding conclusions and further works are discussed.

## UTILITY COMPUTING

This work is focused on the modelling and simulation of service usage in the context of Utility Computing (UC). The term utility thereby refers to the field of industry. Here a public utility (Encyclopaedia Britannica, 2008) describes an enterprise that provides certain classes of services to a wide range of consumers.

The name Utility Computing indicates the vision of IT-based services comparable to public utilities. In this work Utility Computing is defined as a business model (Weill, 2001) for service providers offering IT-based services and charging service consumers per usage, according to (Rappa, 2004). From the provider's IT perspective UC is about service provision that is able to scale dynamically, according to real-time fluctuations in demand (Bunker *et al.*, 2006). Additionally, UC service provision offers its services equipped with the ability to charge service consumption per use (Neel, 2002).

From a consumer's perspective UC is related to "the reduction of IT-related operational costs and complexity" (Yeo *et al.*, 2006). Both perspectives, provision and consumption, have in common to target a better utilisation of generally underutilised IT resources (Andrzejak *et al.*, 2002) on both sides. In summary, UC implicitly claims an abstract description how IT resource utilisation, its total costs and service prices relate (see Figure 1).

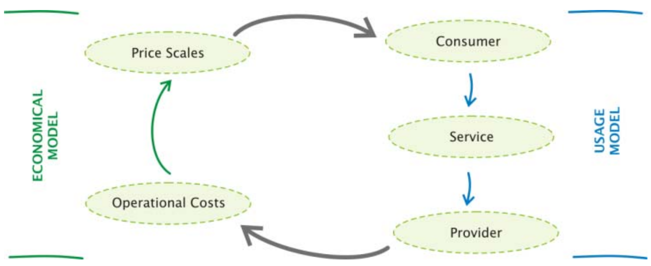

Figure 1: UC's resource – cost – price relation

Thereby Utility Computing does not refer to a specific IT service definition. From a business perspective any service that economically makes sense to be charged by its usage is addressed by UC. Therefore a more abstract service definition will be the most suitable for UC: A service represents a type of relationship-based interaction between a service provider and a service consumer to achieve a certain solution objective. (Zhang *et al.*, 2007) This definition considers the definitions of (Fitzsimmons *et al.*, 2006) from the economics perspective and (Gronroos, 2000) from the marketing perspective. From a technical perspective there are several types of services that fit into this definition, e.g. SOAP web services, HTTP web servers or Xen virtual infrastructures.

## SERVICE OPERATIONS LIFECYCLE

In the context of Utility Computing service provision a lightweight definition of a service lifecycle is necessary to obtain an overview of lifecycle stakeholders and basic activities relevant for service provision. As a basis for the definition of a lifecycle in this work, the basic lifecycle

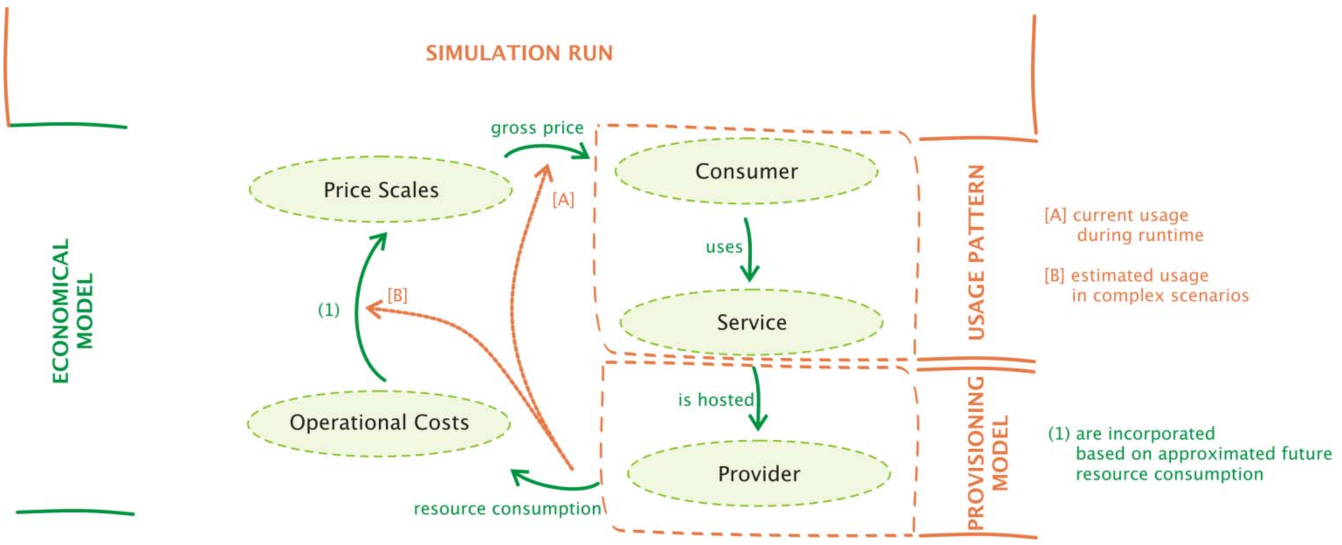

Figure 2: UC relations in the business planning

described in (Zhang et al., 2007) and the aggregation of the ITIL v3 service lifecycle described in (Beard, 2008) are used. Both descriptions can be aggregated to the three main lifecycle phases: Service business planning, service development and service operations.

The lifecycle phase of service business planning is addressing service strategy and service engagement to implement a business model. In classical IT business models, not based on the vision of UC, IT resource utilisation, its total costs and service prices only relate indirectly (see economical model in Figure 2).

During service development the service lifecycle is responsible for the design and implementation of services. This phase also includes the transition process from an implemented service to a deployed, ready for operations service. The phase service operations focuses on the provision of services. This addresses effectiveness and efficiency in delivery and support of services.

## RESEARCH OBJECTIVES

The overall context of this work focuses on specific aspects of the service operations lifecycle (SOL) for service offers based on the business model of Utility Computing. In the phase of service business planning this work refers to the corresponding service properties and service usage profiles resulting from the previous UC definition. During service development and the phase of service operations this work will focus on services in the technical context of Service-oriented Computing (SOC) (Papazoglou, 2003) consistent to the paradigm of Cloud Computing as described by (Boss *et al.*, 2007).

In this context a description of the modifications necessary to transfer a standard service operations lifecycle into a UC SOL is missing. This includes the demand for an explicit definition of UC's core relation between IT resource utilisation, its total costs and service prices. Also specific attention must be given to the implications of complex UC usage scenarios.

The unidentified implications of complex UC usage scenarios, considerably compromise the planning, development and operation of UC service offers. Under these conditions the prediction of resource utilisation and dependent operational costs prediction, calculation of subsequent price scales, and subsequent runtime gross price calculations will fail.

## RESEARCH APPROACH

The overall work starts from the business perspective, as technical requirements depend on the business requirements imposed. Therefore, a five step approach to find solutions for the specified objectives is proposed:

(1) Describe the current state of service usage in the context of Utility Computing.

(2) Elaborate a detailed definition for the relation between a service and its consumer.

(3) Analyse the SOL of UC services.

(4) Determine the implications of complex UC usage scenarios regarding SOL.

(5) Deduct a corresponding strategy to handle the complexity.

This paper focuses on the simulation of the UC model developed as part of the overall work. The simulation, as well as the UC model, is part of the developed strategy to handle the complexity of UC usage scenarios.

## STRATEGY TO HANDLE UC'S COMPLEXITY

The overall work analyses the SOL to identify where modifications allow an optimised support for UC scenarios. Beginning with the phase of service business planning, the classical relation between resource utilisation, costs and prices is examined. Advanced relations for service provision in UC were elaborated as show in Figure 2. The relation marked with [A] adds a runtime relation between

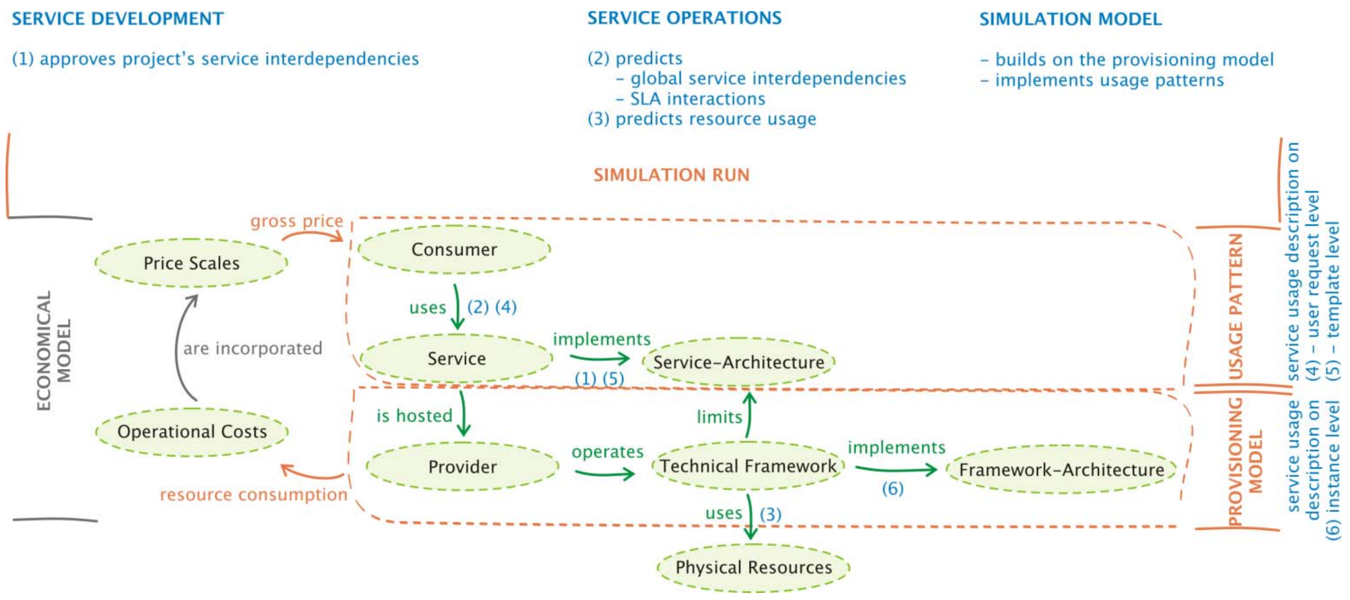

Figure 3: UC relations in service development and operations

the current usage and the gross price calculation that is essential to offer pay-per-use in UC scenarios. Beside this, service providers have to deal with complex usage scenarios, added by relation [B]. To enable the direct relation [A], constraints for UC service provision are necessary. These constraints must describe the requirements to enable this relation during service development and operation.

In the service development phase of the service lifecycle, new data including Usage Patterns specific for on-demand IT infrastructures need to be integrated into the development process (Mendoza, 2007) to improve service quality (Heckmann, 2009). To support the planning of framework architectures or the selection of framework implementations, the definition of relevant UC service provision constraints is necessary.

In the service operations phase, there are no adequate tools to evaluate service interdependencies between all hosted services. Also the Service Level Agreement (SLA) interactions between all hosted services cannot be estimated. Nor the resource planning for services to ensure contracted service levels, respecting resource consumption of other services hosted on shared resources, cannot be analysed without adequate tools for complex UC scenarios.

As a result, of the analyses of the service operations lifecycle, four major strategies for the reduction of the complexity of UC service provision can be identified:

- Define UC constraints for service architectures
- Enable the analysis of service interdependencies on development and operations level
- Permit the analysis of SLA interactions and resource prediction
- Support the proof of price scales

To implement these modifications the development of a technology-agnostic UC service provision model and a corresponding technology-abstracted UC simulation environment is proposed. See Figure 3 for a detailed overview of all previously addressed relations.

## UC MODEL

As the previous strategy suggests, the overall work defined a technology-agnostic UC model (Heckmann, 2007). In summary the model consists of eleven abstract elements, logically grouping demanded functionalities, and three basic workflows, which describe the minimum demanded interaction of those elements.

The abstract elements are consumer groups requesting services with a certain member count, request frequency and characteristics, a broker to forward requests according to costs aspects, a load-balancer to forward respecting load aspects of a request, a host offering resources such as computing cycles, memory, storage and network, and service instances consuming offered resources. Additionally some elements to organise service provision: a registry, monitoring, and a service type element. In a derived technical IT architecture these functional groups can be represented as standalone components, but could also be combined in joint architectural elements. As basic workflows a simple service consumption workflow, a complex service consumption workflow, and a cascaded service consumption workflow where defined.

Other models in this context have been proposed by (Mendoza, 2007), (Zhang *et al.*, 2007) or (Bunker et al., 2006). The model of Bunker and Thomson is the most inadequate of them, since it provides too few details to be helpful for IT architects to design a suitable UC architecture for a specific service provision scenario. The model delivers only a quick overall IT strategy to the provision of UC-based services. The UC model by Zhang, Zhang and Cai was developed from a business management perspective. It specifically aims to the provision of SOAP-based web services and describes in detail how those should be provided. While the model of Mendoza uses a very efficient model building approach, starting from a

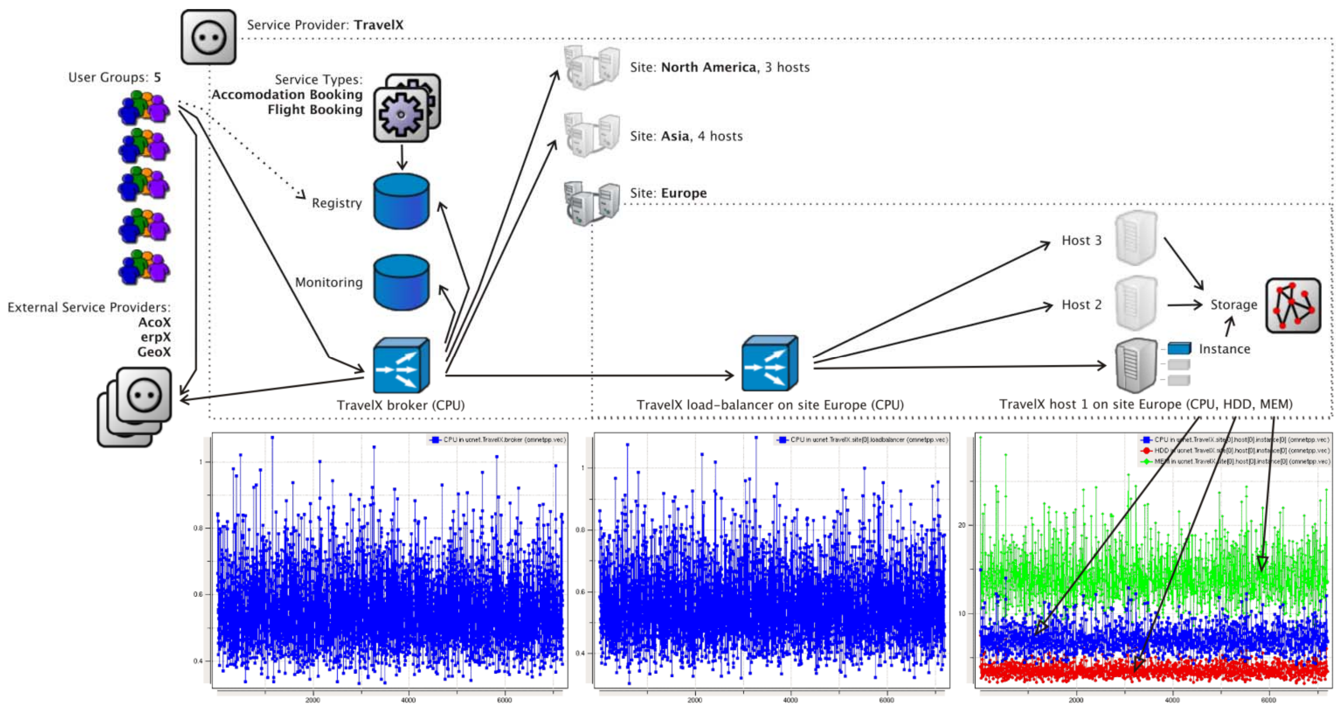

Figure 4: Simulation model as multi-tier architecture

technological perspective. Both of the afore mentioned models are very complex and technology-dependent. Therefore a custom model building was conducted, targeting a lightweight technology-agnostic solution.

## SIMULATION MODEL

The simulation model represents a multi-tier architecture (see Figure 4) for the UC-conform provision of services in service-oriented computing. The functionalities described in the UC model have been transformed into the simulation model that implements this architecture. The current implementation is capable to simulate:

- Complex user behaviour (user group): Messages can be sent with random or fixed timeslots to control the amount of messages arriving at the broker. The resource consumption for transport and processing of the embedded service request can be determined separately. It is also possible to configure transport priorities. Each service request can include a free number of subrequests to represent service cascades.

- Resource measurement and monitoring for computing cycles, memory and disk space: The hardware resources simulated and monitored are computing cycles, memory and disk space. Network traffic (bandwidth and delay) is currently not monitored, but simulated. Additionally monitored are the load-balancer and broker queues and their message transport resource consumption. The overall resource consumption for the storage network is also traced.

- Message billing to service consumers (broker): To each request response a bill based on the processing sites computing cycles, memory and storage costs will be attached. The consumption of these resources during processing of the request is billed, and it is possible to add additional per site and per consumer margins.

- Message routing by site costs (broker): Messages get routed to a site with enough resources to process the request and the least costs for processing.

- Message routing by resource demand (load-balancer): Messages are routed by a site's load-balancer to a host with enough resources.

- Message queuing (broker & load-balancer): Messages are temporarily stored within the service broker or service load-balancer when not enough resources for their processing are available. They are recalled from queue after a certain scheduling time and entered again in the scheduling sequence of either the service broker or the service load-balancer. In doing so the queuing consumes resources in the system, and if the system balancer runs out of resources, incoming messages are dropped.

The simulation model is implemented based on the discrete event simulation environment OMNeT++ (Varga, 2001). For each simulation run it is possible to determine the number of user groups requesting services. It is possible to vary the total number of group members, the behaviour timing as well as the type of request in meanings of service type and request complexity individually per user group. It is possible to specify any service type and any number of service providers. Each provider may have several sites with any number of hosts. Each host can be individually equipped with computing power, memory and storage. Additionally each site has access to a storage network to estimate storage network loads. For each user group and service provider the price relation to each service type can be individually adapted. This highly flexible configuration targets the necessity to represent complex UC scenarios.

## FIRST OUTCOMES

The largest test scenario currently simulated represents a virtual travel booking processor with 2770 consumers in five groups, each sending a single request of the same service type including two subrequests to external service providers. Thereby each request's initialisation is randomly scheduled within a given timeframe. As outcome of each simulation run a data pool consisting of all values stated in the resource measurement and monitoring definition of the simulation model is provided. As examples for graphs based on parts of the outcomes, in Figure 4 the consumption of computing cycles characterised as the CPU utilisation is shown for the TravelX broker, a load-balancer and a host. The host's graph also shows the memory (MEM) and disk space (HDD) utilisation on the contemplated host.

Additionally to simulation runs testing a large amount of concurrent users, the implementation showed that it is able to simulate the behaviour of highly meshed service cascades. Even if it comes to special cases like looping service requests, where subrequest providers themselves use services provided by the original subrequest initiator.

Or in case of internal subrequests occurring, where the provision of a service involves requests to other internally provided services.

## CONCLUSIONS AND FURTHER WORK

This paper identifies the modifications necessary to transfer the standard SOL into a UC SOL. As part of the presented strategy to handle UC's complexity a simulation model is introduced. First tests of this simulation model have shown that it is possible to represent complex scenarios. The prediction of resource utilisation, dependent operational costs and subsequent runtime gross prices has been shown in virtual scenarios. Further the simulation model mustbe validated analysing real world scenarios. The main aspect for adequate representation of the service's behaviour will be the calibration of the simulation runs to reflect the current resource consumption of service requests. Here further research has to be conducted.

Also part of future research must be the documentation of theoretical aspects of the simulation model building. This includes the relation between discrete event simulation and queuing concepts from queuing theory, the revision of relevant probability topics and the relevant background in stochastic processes.

It is assumed that the technology-abstracted simulation model can also be used for the simulation of RESTful or simple services, such as web servers. Here further research will be conducted.

## BIOGRAPHIES

Benjamin Heckmann is a researcher at the aiDa research center in Dieburg and PhD student at the University of Plymouth, UK. He holds a M.Sc. in computer science. His research interests are in the areas of Utility Computing, Cloud Computing, Unified Communications and IT-Security.

Ingo Stengel graduated at the Cork Institute of Technology, Ireland. He is co-founder and Executive Director of the igdv-Centre for Advanced Learning, Media and Simulation at the University of Applied Sciences Darmstadt. His research interests are in the area of Multiagent-Systems, Simulation Software, IT-Security and Advanced Learning.

Andy Phippen received his PhD in the year 2001. He is Senior Lecturer in Business Enterprise and Ethics at the University of Plymouth. His research focuses on the impact of software development and learning & teaching in higher education. Further, he is the director of the IT liaison at the University of Plymouth.

Günter Turetschek is Professor for computer science at the University of Applied Sciences Darmstadt; co-founder and director of the Institute for Applied Informatics Darmstadt (aiDa). His research interests are in the area of Business Computing, Unified Communications and Utility Computing.

## REFERENCES

Andrzejak, A., J. Rolia and M. Arlitt. 2002. "Bounding Resource Savings of Utility Computing Models". HP Labs Technical Report HPL-2002-339.

Beard, H. 2008. *Cloud Computing Best Practices for Managing and Measuring Processes for On-Demand Computing, Applications and Data Centers in the Cloud with Slas*. Emereo Pty Ltd.

Boss, G., P. Malladi, D. Quan, L. Legregni and H. Hall. 2007. "Cloud computing". IBM, developerWorks, WebSphere, High Performance On Demand Solutions.

Bunker, G.; and D. Thompson. 2006. *Delivering Utility Computing: Business-driven IT Optimization*. John Wiley & Sons.

Encyclopaedia Britannica, 2008. *public utility*. In Encyclopaedia Britannica Online, retrieved December, 2008.

Fitzsimmons, J.A.; and M.J. Fitzsimmons. 2006. *Service Management: Operations, Strategy, and Information Technology*. 5th Ed., Irwin/McGraw-Hill, Homewood, IL.

Gronroos, C. 2000. *Service Management and Marketing: A Customer Relationship Management Approach*. John Wiley & Sons.

Heckmann, B. 2007. "Service provision in a utility computing environment". SEIN 2007, University of Plymouth, 14-15 June 2007.

Heckmann, B. 2009. "Technology-agnostic definition of the Utility Computing service operations lifecycle". Transfer Report, University of Plymouth, April 2009.

Mendoza, A., 2007. *Utility Computing Technologies, Standards, and Strategies*. Artech House Inc.

Neel, D. 2002. "The Utility Computing Promise". InfoWorld, April 12, 2002.

Papazoglou, M.P. 2003. "Service-oriented computing: concepts, characteristics and directions". Web Information Systems

Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on 10-12 Dec. 2003 Page(s):3 – 12.

Rappa, M.A. 2004. “The utility business model and the future of computing services”. IBM Syst. J. 43, 1 (Jan. 2004), 32-42.

Yeo, C.S., M.D. Assunção, J. Yu, A. Sulistio, S. Venugopal, M. Placek and R. Buyya. 2006. “Utility Computing on Global Grids”. Hossein Bidgoli (ed), The Handbook of Computer Networks, John Wiley & Sons, New York, USA, accepted in April 2006 and in print.

Varga, A. 1997. “Flexible topology description language for simulation programs”. Simulation in industry: 9th European Simulation Symposium 1997:225-229.

Varga, A. 2001. “The OMNeT++ Discrete Event Simulation System”. In the Proceedings of the European Simulation Multiconference (ESM'2001). June 6-9, 2001. Prague, Czech Republic.

Weill, P. and M.R. Vitale. 2001. “Place to space: Migrating to eBusiness Models”. Boston, Harvard Business School Press.

Zhang, L.-J.; J. Zhang; and H. Cai. 2007. *Services Computing, Core Enabling Technology of the Modern Services Industry*. published by Springer and Tsinghua University Press.