MERIT-Infonomics Research Memorandum series

Retrieval of Service Descriptions using Structured Service Models

Rudolf Müller and Stefan Müller

2001-015

MERIT – Maastricht Economic Research Institute on Innovation and Technology
PO Box 616
6200 MD Maastricht
The Netherlands
T: +31 43 3883875
F: +31 43 3884905

http://meritbbs.unimaas.nl
e-mail:secr-merit@merit.unimaas.nl

International Institute of Infonomics

PO Box 2606
6401 DC Heerlen
The Netherlands
T: +31 45 5707690
F: +31 45 5706262

http://www.infonomics.nl
e-mail: secr@infonomics.nl

Retrieval of Service Descriptions using Structured Service Models
Rudolf Müller and Stefan Müller
May 2001
Keywords: Application Service Provider, Retrieval, subgraph homomorhism
JEL: C61
ACM: H33, G22

Abstract

The Application Service Provider (ASP) market leads to rapidly increasing numbers of sites that offer software as an online service, rather than for download and installation. This creates a demand for intelligent solutions to retrieve the best service to resolve the user's problem. This paper introduces Structured Service Models to represent software services and explores a retrieval mechanism on repositories of structured service models. The mechanism is based on computing graph similarity on a special class of directed acyclic graphs. Finding most similar models is NP-complete, however the special structure of the graphs can be exploited for exact and heuristic algorithms. The paper also presents a prototype system designed as a three-tier client-server application where the client is implemented in Java. The system provides facilities for stating queries on a remote repository by drawing a structured service model in a Java applet. The paper concludes with an initial evaluation of the system.

Rudolf Müller
International Institute of Infonomics and Department Quantitative Economics
University of Maastricht, PO Box 616, 6200 MD Maastricht, Netherlands
Phone +31-43-3883799, e-mail: r.muller@ke.unimaas.nl

Stefan Müller
Wilhelm Schickard Institute for Computer Science
University of Tübingen, Sand 13, 72076 Tübingen, Germany
Phone +49-7071-2975485, e-mail: muellers@informatik.uni-tuebingen.de

1 Introduction

The Internet gives access to numerous libraries of mathematical models for decision making. For example, NetLib [12] and a Princeton Web server [11] contain about 900 AMPL [4] models. So far these services require the models to be installed on the local computer. But *Decision support on demand* [1], where the model can be used interactively on a remote server, is likely to become widely available in the near future. Indeed, application service provision is enjoying a market breakthrough these days, and decision support in applications such as finance and product selection is already offered by many sites.

There is litle retrieval function available for online libraries of models or for decision-support services. This is partly due to the lack of appropriate service descriptions and, as a consequence, of appropriate retrieval algorithms for repositories of such descriptions. Retrieval via model libraries can currently be implemented either by directed search, based on a classification index, or by full text retrieval. This paper introduces a completely new approach in which *structured service models*, a special class of acyclic directed graphs, are used for service representation. A query model from a user is checked for similarity with models in the repository and those that are most similar are reported as a retrieval result.

Our approach is based on *Structured Modeling* by Arthur M. Geoffrion [6]. Broadly speaking, a structured model is an acyclic directed graph whose nodes represent components of the model (entities, decision variables, etc.) with arcs representing definitional dependencies between components. Structured Modeling has been proven to have the potential to capture the essential characteristics of a model in the Management Science field [8]. This motivated us to evaluate its applicability as a paradigm to represent decision support services and to use this representation for retrieval.

The paper is structured as follows. Section 2 introduces *Structured Service Models* and shows the differences to Structured Modeling. Section 3 presents methods for retrieval. We define a distance between models based on mapping the nodes of one model to those in another model to provide a maximum number of arcs. Finding the best mapping is an NP-complete optimization problem,

but the special structure of our models supports the design of exact and heuristic algorithms for similarity computation. Section 4 describes our prototype implementation. First test results are presented in Section 5. We conclude with a summary.

This paper focuses on the application domain of decision support services, yet our approach can easily be extended to repositories of other software services on the Internet. Indeed, our service models define the entities, parameters, variables, constraints, and objective of a service—key elements in the description of a software service.

2 Modeling services by Structured Service Models

We call the description of a service a *Structured Service Model* (SSM). An SSM describes the problem that is solved by the service: data that is input to the service, data that is computed by the service, and the relationships between these data. This approach is based on *Structured Modeling* [6] which was designed to define decision models in Management Science, but which has shown great potential in representing models from other fields as well, e.g. database models [2]. Our SSM can be viewed as a limited version of Structured Modeling, to simplify the usage of the modeling language. Simple structures also open the door for efficient retrieval from model libraries.

An SSM is a directed acyclic graph with textual node labels describing their semantics. Each node represents an item, each arc represents a definitional dependency between items. An SSM distinguishes 6 types of nodes.

An *entity* node represents a primitive item whose definition does not depend on other items. A *parameter* node represents an attribute describing an entity, or combinations of entities, and whose value is used as input to the service. A *variable* node stands for an attribute whose value is computed by the service. In a decision model we can think of it as a decision variable. The definition of parameters and variables are dependent on the definition of the entities they describe. A *function* represents a rule to compute a value from variables and parameters.

In a decision model it represents, for example, the objective. A *test* is a function that evaluates using true or false. It is an expression that defines a constraint on a combination of parameters and variables. A *multi-test* is a collection of tests, representing the fact that a constraint has to be valid for a range of combinations of parameters and variables. We illustrate this in more detail in the example below. The distinction between parameters and variables, as well as between tests and multi-tests, extends concepts of Structured Modeling to ease the construction of SSM from models written in other widely used modeling languages (e.g. AMPL).

The SSM graph contains arcs (v, w) for all nodes v and w for which the definition of the item represented by node w depends on the definition of the item represented by node v. For example, if a node w stands for a parameter that is associated with an entity, represented by node v, an arc (v, w) is added. Or, if w represents a test, we add arcs pointing to w from all variable and parameter nodes that are input to that test.

Arcs are only allowed between specific types of nodes: (1) from entities to variables and parameters, (2) from variables and parameters to functions, tests and multi-tests. Thus, SSM are acyclic directed graphs with three layers of nodes: a layer of entities, a layer of parameters and variables, and a layer of functions, tests, and multi-tests. This restricts structured modeling, as the last of these would allow, e.g. dependencies between functions. As we will see, this supports the computation of similarity of models. On the other hand we believe that it does not excessively restrict the expressiveness of the SSM. Indeed, arcs between functions represent intermediate steps that help to modularize the model, rather than capturing the semantics of a service. Choices of modularization might be rather arbitrary. This could even negatively influence the retrieval quality.

Figure 1 shows the SSM for the Hitchcock-Koopman transportation model. The decision model describes the situation where we have a collection of plants and customers represented by the entities PLANT and CUST. Each plant has a supply and each customer has a demand, modeled by parameters SUP and DEM. For each plant-customer combination we observe per unit transportation costs from plant to customer, modeled as parameter COST. Decision variables

are the numbers of shipments between each plant and each customer, given by the variable FLOW. The total shipment to a customer has to satisfy demand, and total shipment from the plant may not exceed supply. These constraints are modeled by multi-tests T-SUP and T-DEM. The objective is to maximize revenue, which is represented by the function REV.

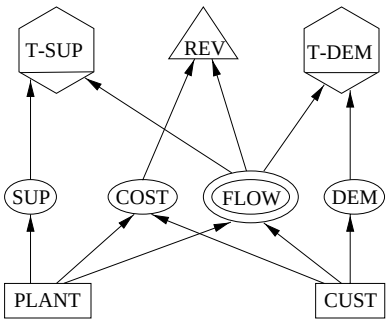

Figure 1: SSM for transportation problem

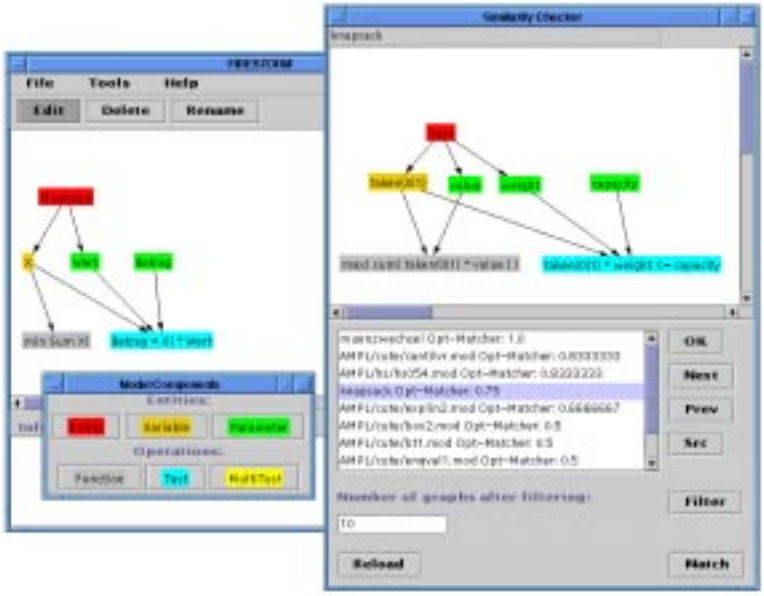

Figure 2: The FIRESTORM user interface

Retrieval of software services can use an SSM in the following way: a model repository provides a library of service models, including the URL to access the service. A user submits a query to the repository by creating his own SSM describing the service he/she is looking for. The retrieval mechanism returns those SSMs that are “close” to the query model. This approach poses the following research questions:

1. Just how precisely can an SSM describe the semantics of a software service and a user’s request for such a service? For which type of service is it particularly valuable?

2. What are appropriate measures for problem similarity? In particular, will retrieval based on searching for similar models provide appropriate precision and recall? How efficiently can we implement retrieval on SSM repositories?

This paper can only address certain parts of these questions. Essentially, we assume that SSMs are good representations of services. This assumption is qualitative and based on the extensive literature on Structured Modeling [7]. Having made this assumption, we choose a structural difference between SSMs (see Section 3) as the measure of similarity (or dissimilarity). Based on this choice we present our prototype repository and retrieval system that provides us with answers to question 2.

3 Retrieval on model repositories

In the previous section we introduced SSM as an approach to model services, particularly decision-support services. The task of SSM is twofold. Firstly, they can be used as a graphical representation of a service that helps a user to decide whether the service meets his/her requirements. Secondly, providers of services can register them in repositories, allowing users to search those repositories for matching services. Furthermore, robots might automatically create an SSM from other service descriptions, such as a description in an algebraic modeling language for a decision model. This section deals with supporting searches in service repositories. We have designed and implemented retrieval algorithms that compute graph similarities and use algorithms from combinatorial optimization.

The core of our retrieval mechanism is a similarity measure that uses the adjacency structure of SSM graphs. We saw in the previous section that graphs consist of three layers, with two node types on the second layer and three node types on the third layer. Thus we can represent a graph by ($U_E, U_P, U_V, U_T, U_M, U_F, A$), where U_E are the entity nodes, U_P the parameter nodes, U_V the variable nodes, U_T the test nodes, U_M the multi-test nodes, U_F the function nodes, and A the arcs. Given two graphs G and G', we look at partial mappings π of the

nodes from G on the nodes of G' which map nodes of the same type, e.g. nodes in U_E on nodes in U'_E. We then count the number of arcs from G that are implicitly mapped on arcs in G', i.e., arcs (v, w) such that $(\pi(v), \pi(w)) \in A'$ and call this number q. The *matching quality* realized by the mapping π is defined as $d(\pi) = 2q/(|A| + |A'|)$. The similarity between two graphs is then defined by the maximum over all matching qualities of mappings from G to G'. We require mappings to be 'one to one'.

The mapping quality is a rational number between 0 and 1, where the quality 1 indicates that a mapping exists between the library graph and the query that exactly matches all arcs. As we can assume w.l.o.g. that there are no isolated nodes in an SSM, this is the case if, and only if, both graphs are isomorphic.

An immediate question is whether we are able to compute the similarity between graphs in polynomial time. The answer is, unfortunately, no.

Theorem 1 *Given two SSM graphs G and G' and a number $s \in [0, 1]$ the problem to decide whether the similarity of G and G' is greater than or equal to s is NP-complete.*

Proof. The NP-complete problem 3-DIM-MATCHING [5] can be reduced to this problem. We leave it out here but simply mention that it shows that even with one type of node on layers 2 and 3, e.g. only variables and function nodes, the problem remains NP-complete. Complete proof can be found in [9].

Although the problem of finding a mapping of optimal quality is NP-hard, the special structure of our graphs supports us in designing heuristic methods. This is due to the 3-layer structure of our graphs. Let us assign new names U_1, U_2, and U_3 to layers, where $U_1 = U_E$, $U_2 = U_P \cup U_V$, $U_3 = U_M \cup U_F$. Given another graph $G' = (U'_1, U'_2, U'_3, A')$ a mapping from a subset of G nodes to G' nodes, mapping nodes of the same type, breaks down into three mappings $\pi_i : U_i \to U'_i$, $i = 1, 2, 3$. Suppose we fix two of these mappings, π_1 and π_2, and want to change the third in order to improve the matching quality. How efficiently can this be achieved? The good news is given below.

Theorem 2 *Given a mapping π between two SSM graphs G and G', splitting into parts π_1, π_2, and π_3, we can efficiently compute for every $j \in \{1, 2, 3\}$ a mapping with optimal matching quality across all mappings π' with $\pi'_k = \pi_k, k \neq j$. Furthermore, for fixed π_2, we can compute a mapping with optimal matching quality across all mappings π' with $\pi'_2 = \pi_2$.*

Proof. For the first part of the theorem we observe that we can find an optimal π'_j by solving a weighted bipartite matching problem in an appropriately constructed bipartite graph $H = (V, V', E)$. Indeed, we take $V = U_j$, $V' = U'_j$, $E = \{(v, v') |\ v, v'$ belong to the same category $\}$. The weight of an edge $\{v, v'\}$ is set to the number of arcs that are realized by π' if v is mapped to v'. Because of the layered SSM stucture this number does not depend on the mapping of other nodes on the same layer. A maximum weighted matching in H thus corresponds to a best π'_j with respect to fixed $\pi'_k = \pi_k, k \neq j$. For the second part of the theorem observe that for layers 1 and 3 the edge weights in our matching graph only depend on how the middle layer is mapped. In both cases the optimal match can be found in polynomial time (see e.g. [10]).

By optimizing π on one layer while fixing it on the other layers the assignment on the other layers does not stay optimal. Nevertheless our theorem is a good base for exact or heuristic algorithms.

For an exact approach we can enumerate all feasible mappings of nodes from the second layer and, for each of them, solve the matching problems on layers 1 and 3 to optimum. This algorithm is polynomial for a fixed number of nodes on layer 2, and a reasonable algorithm for small numbers of nodes on layer 2.

For a heuristic approach we can use a local search framework (see, e.g. [10]). A feasible solution is given by mapping π. Every feasible solution has two neighbors, given by partial fixes of π: keep layer 2 fixed, keep layer 1 and 3 fixed. We choose the best mapping for both neighbors. We can implement different search strategies on this neighborhood structure, best neighbor, tabu search, or simulated annealing.

Note that the neighborhood graph consists of disjoined cycles. It might therefore be better to extend the neighborhood structure by looking at more than just

two feasible solutions. This can be achieved by changing a mapping π only in a few positions in the unfixed part, instead of just optimizing using Theorem 2. For the time being we have implemented this version of a local search. It starts at an arbitrary mapping π of nodes and calculates the number of realized edges. While there are a pair of nodes of the same category in G, for which an exchange of images under π increases the number of realized edges, this exchange is carried out. If no such exchange is found the algorithm reports the solution reached as a local optimum.

4 Description of the FIRESTORM prototype

The purpose of the prototype is to validate our proposals for service retrieval and to improve them. On the one hand this requires that we benchmark our algorithms on large model repositories and, on the other hand, it requires real users to test the system and report their experiences. We have therefore implemented a Web-based client-server system, consisting of a Java applet on the client side that implements the user interface, a retrieval server with retrieval algorithms, to which the Java applet sends retrieval requests, and a database with collections of SSMs. This prototype has been named FIRESTORM (FIrst a REtrieval SysTem for Operations Research Models).

We start with an illustration of the user interface as it best describes the functionality of the system. Once the user has loaded the Java applet from the site http://134.2.11.23/firestorm a FIRESTORM frame opens (left part of Figure 2). Within this frame the user can edit an SSM, which represents the query to the system. In a future release service providers can use this frame to submit models, thus extending the collection of models in the FIRESTORM database. The client provides all functions to insert, delete and move nodes and arcs. After creating a query model, the user opens the similarity checker for retrieval (right part of Figure 2). This frame activates the graph similarity algorithms from section 3 with a possibility to restrict the size of the query result. It contains functions for browsing through the retrieval results and viewing additional information (e.g. the

source model, if the SSM was generated from an AMPL model).

The retrieval server encapsulates the retrieval algorithms. Retrieval is carried out in main memory; all models are currently loaded at the beginning of a user's session. A local search for graph similarity is available, using two exchanges to define the neighborhood structure with a best neighbor strategy as search strategy, as well as an exact method covering all assignments of the middle layer nodes (see Section 3).

Evaluating retrieval systems requires a sufficient number of retrievable documents, but where do you get a large number of service models from? In future, we will use the system to track service providers and request that they register with us, but in the meantime we had to find an alternative solution. We therefore decided to concentrate on a particular application domain and, within the domain, on a particular type of service. The domain is Operations Management (OM), and within the domain we looked for services that already have a formal description. This led us to digital libraries for decision-support models represented in various modeling languages (e.g. AMPL, MP, spreadsheets). Again as a first step, we decided to focus on the modeling language AMPL. A large library of AMPL models is provided by Robert Vanderbei at Princeton University [11], containing around 850 AMPL models. We used this library as an initial test-bed. We have implemented a parser that automatically translates AMPL models into SSM. The parser generates a library of SSMs stored in a relational database on the server side.

5 First evaluation results

We currently only have feedback from colleagues and students within our institutions. However, we observed that people who are familiar with optimization models find it easy to convert their idea of a problem into an SSM. People lacking this experience face problems in identifying entities, parameters, variables and the dependencies between them. This is not surprising since this is the core of mathematical modeling, which is generally considered to be a fairly difficult task. A useful extension might be a wizard that supports a user through the model

creation process. Wizards could also provide model skeletons for application domains, which a user initially selects and then refines according to his/her specific problem description.

In a second test we added a knapsack model to our library and asked users to model the problem of changing a bill into a minimum number of coins, where coins of different value are available, as an SSM. This problem is a special case of the knapsack problem. The result was that the knapsack model in the library always had a very high ranking in the retrieval result. Aside from a few combinatorial optimization models, the model base mainly consists of non-linear optimization problems, due to the lack of extensive libraries from other fields (e.g. linear optimization). Test users should therefore be familiar with non-linear models, or the model base will have to be extended. Again, we might use AMPL models, but model libraries written in other languages, or models provided by users are also a possibility. We are currently developing a mapping from Unified Modeling Language (UML) class diagrams into SSM. This extends our system to a retrieval system for reusable software components with retrieval mechanisms similar to those presented in [3].

In terms of efficiency our results are very satisfactory. For the current size and number of models in the database the local search-based graph similarity algorithm reports results almost immediately. The exact method should only be used on a reduced number of models, for example those for which the fast heuristic reports offer high similarity. Further filter algorithms with other distance measures that are faster to compute (e.g. a distance according to edge-type vectors) will be explained in a forthcoming paper. They will be essential in reducing response times when both the database and the stored models are growing larger.

Let us mention the most critical point at the end. This is the question of how well SSMs capture the semantics of services in general, and models from OM in particular. So far only structure has been compared, which is not likely to be precise enough. However, the purely structural approach can simply be extended, without leaving the algorithmic framework of computing graph similarity. Firstly, we increase the semantic expressiveness of SSM by further specializing the node

types (e.g. introducing *Sum* or *Product* as special function element type). This could even decrease the complexity of the graph similarity problem as it restricts the number of feasible mappings. Secondly, we can combine the structural retrieval with syntactic retrieval on the element names. This would be consistent with the algorithms we have presented in Section 3, since syntactic distance of node names can refine the measurement of the quality of a node mapping—for example by using it to define edge weights in the bipartite matching algorithm from Theorem 2.

6 Summary

We have presented a new and easy-to-use approach for representing software services as SSM. We have shown that this representation can be used as a basis for retrieval from repositories of services using the network structure. While we are able to show that finding the best mapping between model nodes is NP-complete, we could also show that the mapping problem has structure that supports the design of exact or heuristic methods. We have described the functionality of the prototype system, available on the World Wide Web at http://134.2.11.23/firestorm. We finally concluded with a preliminary evaluation of our approach.

Depending on further feedback from test users we will extend the system by using more specific types of model components, refining the similarity measure between models and, depending on a certain application domain, by offering frequently used node names and modeling wizards. We will also provide a set of generic models that a user could customize according to his or her needs.

Acknowledgement

We thank the referees for their valuable comments in helping us to improve the quality of our contribution.

References

[1] H.K. Bhargava, R. Krishnan, and R. Müller. Decision support on demand: Emerging electronic markets for decision technologies. *Decision Support Systems*, 19(3):193–214, 1997.

[2] K. Chari and T. Sen. A graphical modeling system: Applications in organizational model management. *Omega, Int. J. Mgmt Sci.*, 25(2):241–253, 1997.

[3] B. H. C. Cheng and J.-J. Jeng. Reusing analogous components. *IEEE Transactions on Knowledge and Data Engineering*, 9(2), 1997.

[4] R. Fourer, D. M. Gay, and B. W. Kernighan. A modeling language for mathematical programming. *Management Science*, 36(5):519–554, 1990.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1987.

[6] A. M. Geoffrion. An introduction to structured modeling. *Management Science*, 33(5):547–588, 1987.

[7] A. M. Geoffrion. An Informal Annotated Bibliography on Structured Modeling. WMSI Working Paper 390, UCLA, 1999. available at http://www.anderson.ucla.edu/faculty/art.geoffrion/home/biblio/ia.htm.

[8] C. V. Jones. An introduction to graph-based modeling systems, part i: Overview. *ORSA Journal on Computing*, 2(2):136–151, 1990.

[9] R. Müller and S. Müller. Retrieval of service descriptions using structured service models (extended version). Working paper, 2000. Available at http://www-db.informatik.uni-tuebingen.de/~muellers/papers/ssm_ext.pdf.

[10] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.

[11] Princeton University. Nonlinear Optimization Models – Sample AMPL Examples, 2000. Available at http://www.sor.princeton.edu/~rvdb/ampl/nlmodels/index.html.

[12] UTK and ORNL. Netlib Repository of mathematical software, papers, and databases, 1995. http://www.netlib.org/.

MERIT-Infonomics Research Memorandum series
- 2001-

2001-001 **The Changing Nature of Pharmaceutical R&D - Opportunities for Asia?**
Jörg C. Mahlich and Thomas Roediger-Schluga

2001-002 **The Stringency of Environmental Regulation and the 'Porter Hypothesis'**
Thomas Roediger-Schluga

2001-003 **Tragedy of the Public Knowledge 'Commons'? Global Science, Intellectual Property and the Digital Technology Boomerang**
Paul A. David

2001-004 **Digital Technologies, Research Collaborations and the Extension of Protection for Intellectual Property in Science: Will Building 'Good Fences' Really Make 'Good Neighbors'?**
Paul A. David

2001-005 **Expert Systems: Aspects of and Limitations to the Codifiability of Knowledge**
Robin Cowan

2001-006 **Monopolistic Competition and Search Unemployment: A Pissarides-Dixit-Stiglitz model**
Thomas Ziesemer

2001-007 **Random walks and non-linear paths in macroeconomic time series: Some evidence and implications**
Franco Bevilacqua and Adriaan van Zon

2001-008 **Waves and Cycles: Explorations in the Pure Theory of Price for Fine Art**
Robin Cowan

2001-009 **Is the World Flat or Round? Mapping Changes in the Taste for Art**
Peter Swann

2001-010 **The Eclectic Paradigm in the Global Economy**
John Cantwell and Rajneesh Narula

2001-011 **R&D Collaboration by 'Stand-alone' SMEs: opportunities and limitations in the ICT sector**
Rajneesh Narula

2001-012 **R&D Collaboration by SMEs: new opportunities and limitations in the face of globalisation**
Rajneesh Narula

2001-013 **Mind the Gap - Building Profitable Community Based Businesses on the Internet**
Bernhard L. Krieger and Philipp S. Müller

2001-014 **The Technological Bias in the Establishment of a Technological Regime: the adoption and enforcement of early information processing technologies in US manufacturing, 1870-1930**
Andreas Reinstaller and Werner Hölzl

2001-015 **Retrieval of Service Descriptions using Structured Service Models**
Rudolf Müller and Stefan Müller

Papers can be purchased at a cost of NLG 15,- or US$ 9,- per report at the following address:

MERIT – P.O. Box 616 – 6200 MD Maastricht – The Netherlands – Fax : +31-43-3884905
(* Surcharge of NLG 15,- or US$ 9,- for banking costs will be added for order from abroad)

Subscription: the yearly rate for MERIT-Infonomics Research Memoranda is NLG 300 or US$ 170, or papers can be downloaded from the internet:

http://meritbbs.unimaas.nl
http://www.infonomics.nl
email: secr-merit@merit.unimaas.nl