ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2005, Vol. 34, No. 2

TRANSFORMATION FOR DESIGNING DISTRIBUTED INTERNET INFORMATION SYSTEMS UNDER MODEL DRIVEN ARCHITECTURE

Sigitas Drasutis, Vytautas Pilkauskas, Dalius Rubliauskas

Department of Practical Informatics, Kaunas University of Technology
Studentų st. 50, LT – 51368, Kaunas, Lithuania

Abstract. This article deals with the specification and transformation of distributed systems under Model Driven Architecture (MDA). It reviews basic concepts and benefits of MDA approach, its relations with traditional software development and suggests an alternative method for specification of distributed internet information systems using unified modeling language (UML) notation and transformation of these specifications into program code. Transformation method enables to ease work of the system analyst when creating large and complex information systems. It allows separating full system specification into different parts and converting functions of these parts into platform dependent program code. This article also refers to a synthesis of object interaction diagrams of distributed systems mentioned before.

1. The traditional software development process

Progress in software development is evident from the fact that it is feasible to build much more complex and larger systems. Just think how quickly and efficiently we would be able to build a monolithic mainframe application that has no graphical user interface and no connections to other systems. We never do this anymore, and that is why we do not have solid figures to support the idea that progress has been made.

A typical software development process, as illustrated in Figure 1, includes a number of phases [1].

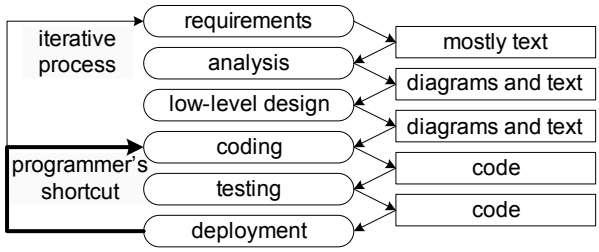

Figure 1. Typical software development process

Still, software development is an area in which we are struggling with a number of major problems [1]:

- The productivity problem – whether we use an incremental and iterative version of this process, or the traditional waterfall process, documents and diagrams are produced during first three phases. These include requirements descriptions in text and pictures, and often many Unified Modeling Language (UML) [2] diagrams like use cases, class diagrams, interaction diagrams, activity diagrams, and so on. Created documents and corresponding diagrams rapidly lose their value as soon as the coding starts. The connection between the diagrams and the code fades away as the coding phase progresses. Instead of being an exact specification of the code, the diagrams usually become more or less unrelated pictures.
- The portability problem – the new technologies (for example XML, UML, J2EE, .NET, JSP, ASP, Web Services, and so on) offer tangible benefits for companies and many of them cannot afford to lag behind. As a consequence, the investments in previous technologies lose value and they may even become worthless.
- The interoperability problem – software systems rarely live in isolation. Most systems need to communicate with other, often already existing, systems. The new end-user application runs in a

Web browser (using various technologies like HTML, ASP, JSP, and so on) and it needs to get its information from existing back-end systems.

The maintenance and documentation problem – writing documentation during development costs time and slows down the process. It does not support the developer's main task. This is one of the main reasons why documentation is often not of very good quality. The only persons that can check the quality are fellow developers who hate the job of writing documentation just as much. With every change in the code the documentation needs to be changed as well – by hand!

2. The Model Driven Architecture

The Model Driven Architecture (MDA) is a framework for software development defined by the Object Management Group (OMG). Key to MDA is the importance of models in the software development process [1].

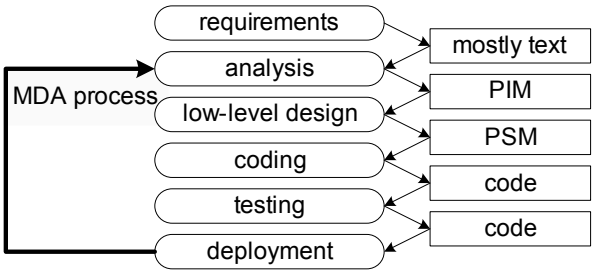

Figure 2. MDA software development process

The MDA development life cycle, which is shown in Figure 2, does not look very different from the traditional life cycle. The same phases are identified. One of the major differences lies in the nature of the artifacts that are created during the development process. The artifacts are formal models, i.e., models that can be understood by computers. The following three models are at the core of the MDA:

- Platform Independent Model (PIM). A PIM describes a software system that supports some business. Within a PIM, the system is modeled from the viewpoint of how it best supports the business. Whether a system will be implemented on a mainframe with a relational database or on an EJB application server plays no role in a PIM.
- Platform Specific Model (PSM). PSM is tailored to specify a system in terms of the implementation constructs that are available in one specific implementation technology. For example, an EJB PSM model typically contains EJB-specific terms like "home interface," "entity bean," "session bean," and so on. A relational database PSM includes terms like "table," "column," "foreign key," and so on. It is clear that a PSM will only make sense to a developer who has knowledge about the specific platform.

Code. The final step in the development is the transformation of each PSM to code. Because a PSM fits its technology rather closely, this transformation is relatively straightforward.

3. MDA benefits

Let us now take a closer look at what application of MDA brings us in terms of improvement of the software development process:

- Productivity – in MDA the focus for a developer shifts to the development of a PIM. The PSMs that are needed are generated by a transformation from PIM to PSM:
  - the PIM developers have less work to do because platform-specific details need not be designed and written down. At the PSM and code level, there is much less code to be written, because a large amount of the code is already generated from the PIM.
  - the developers can shift focus from code to PIM, thus paying more attention to solving the business problem at hand. This results in a system that fits much better with the needs of the end users.
- Portability – within the MDA, portability is achieved by focusing on the development of PIMs, which are by definition platform independent. The same PIM can be automatically transformed into multiple PSMs for different platforms. Everything that is specified at the PIM level is therefore completely portable.
- Interoperability - If we are able to transform one PIM into two PSMs targeted at two platforms, all of the information we need to bridge the gap between the two PSMs is available. For each element in one PSM we know from which element in the PIM it has been transformed. From the PIM element we know what the corresponding element is in the second PSM. We can therefore deduce how elements from one PSM relate to elements in the second PSM.
- Maintenance and Documentation – working with the MDA life cycle, developers can focus on the PIM, which is at a higher level of abstraction than code. The PIM is used to generate the PSM,

which in turn is used to generate the code. The model is an exact representation of the code. Thus, the PIM fulfills the function of high-level documentation that is needed for any software system.

4. Technological architectures of modern internet applications

At first answer the question: if internet systems are distributed systems? A distributed system is a piece of software that ensures that: a collection of independent computers that appears to its users as a single coherent system. There are two aspects of these systems: (1) independent computers and (2) single system – middleware [3]. Distributed systems usually use some kind of client-server organization.

A typical web-based application would have the following logical application logic partitioning:

- *Presentation Layer* – manages user interfaces and user interaction.
- *Business Logic Layer* – performs business transactions and models business processes.
- *Data Layer* - used by the business logic layer to persist business data.

These layers are mapped to corresponding physical tiers where the actual application code resides and is executed. Usually we can expect to have the following tiers [4]:

Presentation Tier

Runs within a Web Server, which hosts a number of so-called web components, based on presentation-related technologies such as: HTML, ASP, Servlets, JSP, etc. In a clustered environment we can have several Web servers hosting the same web components in order to provide maximum scalability of the presentation tier. Web Routers can be used to distribute traffic among the available web servers.

Business Logic Tier

Usually runs within an Application Server, which provides a containment environment for business logic components and a number of middleware services such as transaction management, persistence, state management, security, resource pooling, cashing, etc. In a multi-node configuration this tier consists of several machines each running an instance of the Application Server. Modern application servers provide component clustering functionality, which handles the complexities of replication, load balancing and fail over for business logic components. More concretely, an intelligent active load balancer tracks information about the load of each application server and distributes client requests so that none of the servers is overloaded. Furthermore, the state of business components is replicated across servers. In this way, in case one of the servers fails, requests can be rerouted to another server, which can automatically restore the needed components and provide transparent failover.

Data Tier

Consists of one or more databases and a Database Server, which manages data persistence. This tier may contain data-related logic in the form of stored procedures. As with web servers and application servers, database servers can also be clustered in order to provide maximum scalability, load balancing and fault tolerance in the database tier.

Independent computers, middleware and client-server organization of the internet systems proves that they have all characteristics of the distributed systems. So it is possible to apply all design rules of the distributed system for designing internet information systems.

5. The transformation of the distributed systems

Earlier described software development models are not finally finished. What will happen in traditional software development process or MDA approach if there won't be validation after each development stage and if inaccuracy will be noticed at the deployment time? Project cost will grow twice. To complete project finally few iterations may be not enough. This is why verification and validation (V&V) is needed. The verification and validation is the generic term for checking processes which ensure that the software meets its requirements and that the requirements meet the needs of the software procurer. This is a continuing process through each stage of the software development process. The V&V process has two objectives:

- The discovery of defects in the system.
- The assessment of whether or not the system is usable in an operational situation.

More realistic software development process view with verification and validation of the product after each development stage is illustrated in Figure 3.

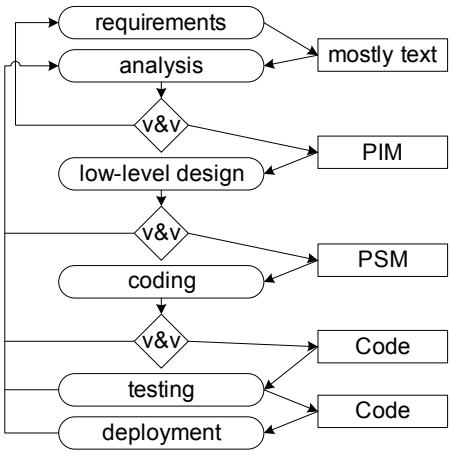

Figure 3. Real MDA development process

This process is suitable for information systems that are not distributed and working like a single whole. It can't be adjusted for the system that consists of few different parts which have to be created separately. Then development process must be separated and every part created in its own way. After all, testing and deployment of the full system is needed.

In case of the distributed system the software development process is more complicated as illustrated in Figure 4.

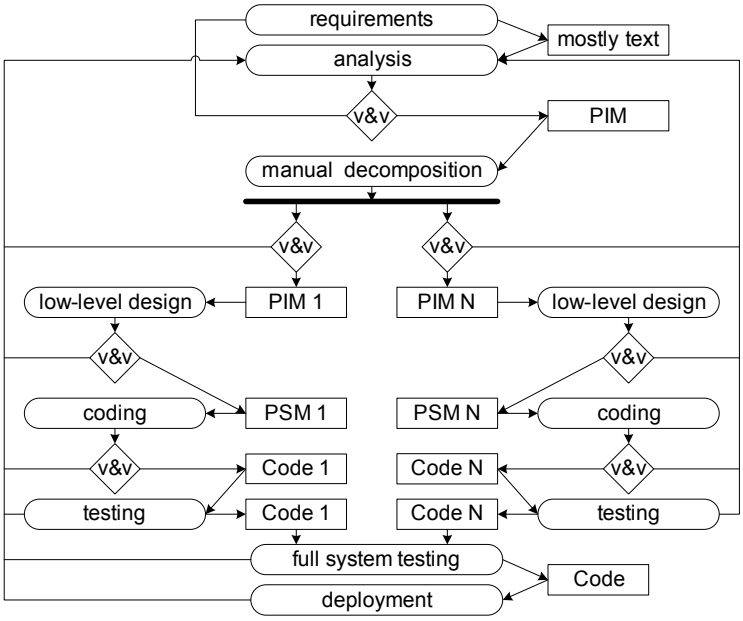

Figure 4. MDA development process for the distributed systems

After analysis step manual decomposition of distributed system to separate parts is required. This process requires of high level software specialist who has to divide system specification into parts and define protocol and messages needed for communication of system components. In this place a lot of knowledge about different platforms, programming languages is expected. Work is really difficult and mistakes can be done easily.

This article refers to earlier published synthesis algorithm ("Synthesis of Object Interaction Diagrams of Distributed Systems" [5]) which will be used for the decomposition of the whole system specification into separate parts and also here will be offered further transformation of the state diagrams into the program code.

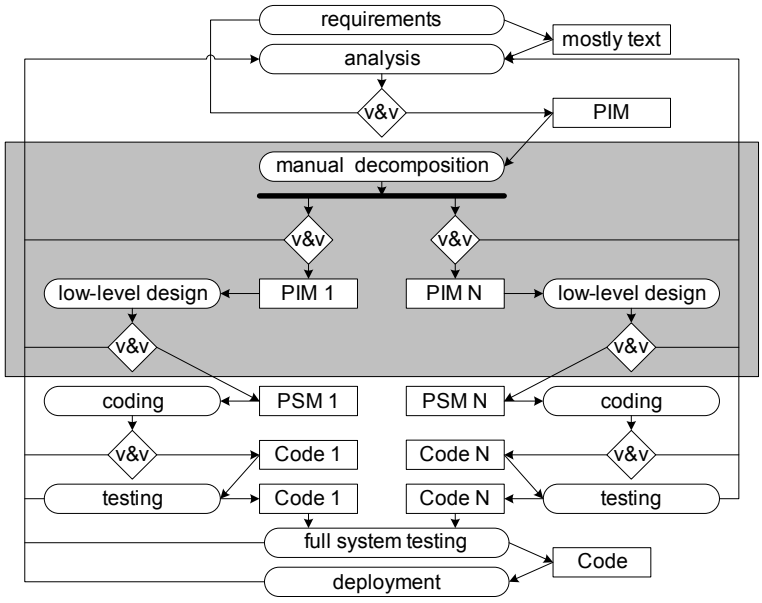

Figure 5. Transformation algorithm under MDA development process

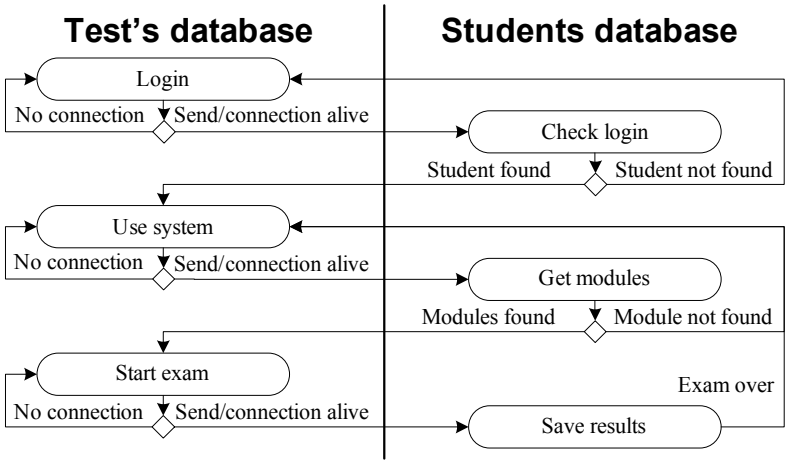

Figure 6. Interaction diagram for the distributed system

Transformation takes PIM written in UML interaction diagram as input and enables automatic model decomposition of the system. The output of the first step is platform independent models for each part of the distributed system: PIM1, PIM2 to PIMn presented with UML state diagrams. Next, low level design using user defined templates is applied and output of the platform specific program code is generated [6].

Transformation algorithm takes into account two steps of MDA software developments process as illustrated in Figure 5.

Notice, that validation and verification of product after decomposition is not needed because applied synthesis method ensures correctness of the algorithm output.

The input data for the transformation are UML interaction diagram of a desired service of the distributed system. Using strips in the diagram is very convenient for separating functions of each part of the system (see Figure 6).

The example shows a test system which communicates with student's database of whole university. On one hand, there is independent software system for students testing and university database on another. It is simple to create an interaction diagram to illustrate whole system work, but there may be some problems to define specifications for each part and communications between them. PHP is software for test's database and university student's database runs on Oracle.

```
<?xml-stylesheet type="text/xsl"
href="x.xsl"?>
<function>
    <title>login</title>
    <data>
        <input>
            <name>login</name>
            <dtype>varchar2</dtype>
            <length>10</length>
            <type>text</type>
        </input>
        <input>
            <name>password</name>
            <dtype>varchar2</dtype>
            <length>10</length>
            <type>password</type>
        </input>
        <output>
            <name>id</name>
            <dtype>number</dtype>
            <length>10</length>
            <type>text</type>
        </output>
    </data>
    <functionality>
        <button>
            <caption>Login</caption>
            <type>submit</type>
        </button>
        <button>
            <caption>Clear</caption>
            <type>reset</type>
        </button>
    </functionality>
    <style>
        <body>
            <background-
color>#669999</background-color>
            <color>#ffffcc</color>
        </body>
    </style>
</function>
```

After all data are collected algorithm of synthesis of object interaction diagrams of distributed systems may by applied.

This interaction synthesis algorithm was proposed using a synthesis procedure of a real-time system protocol created in University of Montreal. This protocol allows a protocol specification for each component of the distributed system and a specification of the communication medium from the desired system's service specification and communication medium's model.

The input data for our algorithm are Interaction Diagram of a desired service of the distributed system (see Figure 6). The output data are state diagrams for each part of the distributed system service [8, 9, 10, 11].

The procedure consists of seven steps:

- First two steps are designed for transformation of interaction diagram to Finite State Machine.

- Steps from sixth to seventh separate functions of the different system components.

The result of the synthesis method described above shown in Figure 7 and Figure 8.

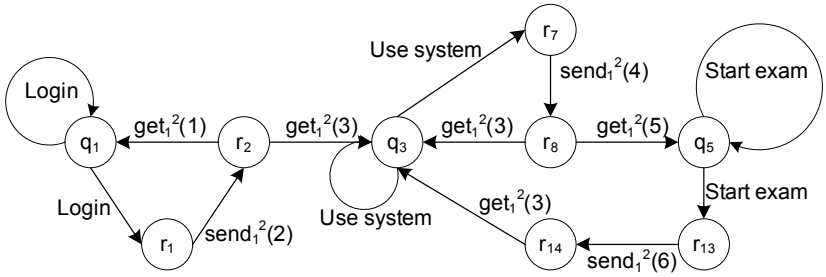

Figure 7. Finite State Machine (FSM) for the first system component

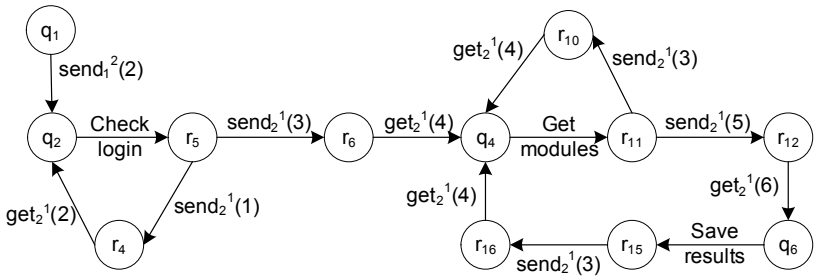

Figure 8. Finite State Machine (FSM) for the second system component

Now it is time to define transformations for each part of the distributed system. Using XSL language [7] for the transformations relieves applying them later because parameters are collected with XML. XSL also is well defined and known language what makes definition process easier because the developer does not have to learn any new notation.

An example of transformation for Oracle PL/SQL stored procedure is shown below:

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0" xmlns:xsl=
    "http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  procedure fn_<xsl:value-of
select="function/title"/>(
  <xsl:for-each select="function/data/input">
    <xsl:value-of select="name"/> in
    <xsl:value-of select="dtype"/>
    (<xsl:value-of select="length"/>),
  </xsl:for-each>
  o_cursor out cursortype)
  is
  a_name varchar2(255);
  s_select_statement varchar2(3000);
  begin
  a_name := 'fn_<xsl:value-of
select="function/title"/>';
  s_select_statement := 'select
  <xsl:for-each
select="function/data/output">
    <xsl:value-of select="name"/>
    <xsl:if test="position()!=last()">
      <xsl:text>,</xsl:text>
    </xsl:if>
  </xsl:for-each>
  from dual';
  open o_cursor for s_select_statement;
  exception
  when no_data_found then
    return;
  when others then
    raise_application_error(-
20001,'Exception:' || a_name);
  end;
</xsl:template>
</xsl:stylesheet>
```

The transformation returns PL/SQL code:

```
procedure fn_login(
    login in varchar2 (10),
    password in varchar2 (10),
    o_cursor out cursortype)
is
    a_name varchar2(255);
    s_select_statement varchar2(3000);
begin
    a_name := 'fn_login';
     s_select_statement :=
        'select id from dual';
    open o_cursor for s_select_statement;
exception
when no_data_found then
    return;
when others then
     raise_application_error
        (-20001,'Exception:' || a_name);
end;
```

Also example of transformation for PHP file:

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0" xmlns:xsl=
    "http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html xmlns="http://www.w3.org/1999/xhtml"
lang="en" xml:lang="en">
<head>
  <title><xsl:value-of
select="function/title"/></title>
</head>
<body>
```

```
<form method="post" action="">
<table>
<xsl:for-each select="function/data/input">
  <tr><td><xsl:value-of
    select="name"/></td>
  <td>
  <xsl:element name="input">
 <xsl:attribute name="type"><xsl:
    value-of select="type"/></xsl:attribute>
    <xsl:attribute
name="maxlength"><xsl:value-of
select="length"/></xsl:attribute>
  </xsl:element>
  </td></tr>
</xsl:for-each>
</table>
<xsl:for-each
select="function/functionality/button">
  <xsl:element name="input">
    <xsl:attribute name="type"><xsl:
     value-of select="type"/></xsl:attribute>
    <xsl:attribute name="value"><xsl:value-of
select="caption"/></xsl:attribute>
  </xsl:element>
</xsl:for-each>
</form>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Once defined transformations could be used many times and changes in main specification accompanied with transformation will return program code without any efforts.

This transformation is important because the proposed algorithm relieves specifying big and complex distributed information systems and helps to avoid much iteration of system analysis and validation.

6. Conclusions

This algorithm was used to create a working internet system for student testing in Kaunas University of Technology. The potential contribution of this research is the use of theoretical systems transformation algorithm in practical way. Transformation algorithm creates about 64% of the system code. For the realization of the research objectives the combination of the different research methods was used, starting from scientific literature analysis on the synthesis methods and its application for the design of the information systems to the specifics of the internet systems comparing to the common traditional software systems. Based on the literature collected on the advantages and disadvantages of the current transformation methods, where some of them are already found and eliminated during the research, some additional improvements still can be included in the transformation algorithm (e.g., real-time criteria). Based on the input data from the analysis, the transformation method was used for designing of the internet information system.

References

[1] **A. Kleppe, J. Warmer, W. Bast.** MDA Explained: The Model Driven Architecture: Practice and Promise. *Addison Wesley*, 2003, 5-25.

[2] **G. Booch and etc.** The unified modeling language user guide. *G. Booch, J. Rumbaugh, I. Jacobson. Addison Wesley Longman Inc*, 1999, 218-225.

[3] **M. van Steen.** Distributed Systems, Principles and Paradigms. *Vrije Universiteit Amsterdam, Holland*, 2001, 1-2.

[4] **S.D. Kounev.** Performance Prediction, Sizing and Capacity Planning for Distributed E-Commerce Applications. *Information Technology Transfer Office, Germany*, 2001, 5-6.

[5] **S. Drasutis, V. Pilkauskas, D. Rubliauskas.** Synthesis of Object Interaction Diagrams of Distributed Systems. *Information technology and control, Kaunas, Lithuania, Vol.*27, *No.*2, 2003, 7-11.

[6] **J.Ch. Park, R.E. Miller.** Synthesizing Protocol Specifications from Service Specifications in Timed Extended Finite State Machines. *University of Maryland, College Park*, 1996, 1-2.

[7] **E.T. Ray.** Learning XML. *Second Edition, O'Reilly, Santa Clara, CA*, 2003, 27-143

[8] **R.L. Probert, K. Saleh.** Synthesis of Communication Protocols. *Survey and Assessment IEEE Trans. Comput., Vol.*40, *No.*4, 1991, 468-476.

[9] **J. Rumbaugh.** Object-Oriented Modeling and Design. J. Rumbaugh, *M. Baha, W. Premerlani, F. Eddy. Englewood Cliffs, New Jersey, Prentice-Hall*, 1991, 97-102.

[10] International workshop on graph-theoretic concepts in computer science. *Graph-theoretic conceps in computer science. Herrsching, Germany*, 1995, 124 - 129.

[11] **A. Khoumsi, G.V. Bochmann, R. Dssouli.** On specifying services and synthesizing protocols for real-time applications. *Universite de Monreal*, 1994, 7-34.