Society Information Grids

Jens H. Jahnke[1], Iryna Bilykh[1], Barbara Kursawe[1], Yury Bychkov[1], David Dahlem[1], Adeniyi Onabajo[1], Craig Kuziemsky[2], and Francis Lau[2]

[1] net#lab, Department of Computer Science, University of Victoria
Victoria, B.C., V8W3P6, Canada
{jens, ibilykh, kursaweb, ybychkov, ddahlem, onabajo}@netlab.uvic.ca
http://www.netlab.uvic.ca
[2] School of Health Information Science, University of Victoria,
Victoria, B.C., V8W3P6, Canada
{craigk,fylau}@uvic.ca, http://hinf.uvic.ca

Abstract. Electronic business processes and distributed data interchange are no longer limited to particular organizations and customer groups. With Web connectivity becoming a standard in most organizations and households, governments seek rationalization of public services (like health care) by using Web-based information integration. Inherently, such *Society Information Grids* require large-scale and asynchronously evolving distributed object networks of interoperating organizations. Many such applications also require information privacy and strict auditability of information flow. This paper describes an approach to realizing such a mediation technology based on the idea of auditable Peer-to-Peer networks. The behavior of this middleware is determined by a formalization based on Petri-net theory. An implementation prototype has been created based on the Microsoft .NET framework. The described research is carried out in tight collaboration of two academic departments with industry and public organizations primarily in the application domain of Health Care.

1 Introduction

Recently, an increasing number of governments and public service organizations have announced interest in exploiting distributed object technology for cutting the costs (and improving the quality) of public services. This development has become possible because of the broad coverage of Web connections in organizations and households. The Canadian government emphasized the importance of information and technology as *key enablers in meeting the challenges of the 21st century and the shared commitment to deploying them within the health sector in Canada* [16]. Auditable yet flexible networking software (so-called middleware) has been recognized as a critical prerequisite for this envisioned *pan-Canadian Health Infostructure* [16]. This vision is just one particular example for an emerging kind of Web information network we refer to as *Society Information Grids* (SIGs).

SIGs are significantly different from most existing distributed electronic-data-interchange (EDI) networks due to the fact that SIGs will be pervasive and participation will be implied (or at least strongly encouraged) for everyone. Consequently, SIG will employ *very large-scale* networks of objects that have to facilitate *rapid* and *asynchronous evolution*. While security is a concern in most EDI networks, SIG applications typically also require effective mechanisms for managing information *privacy* and *audit trails*. Moreover, the ability to deal with system *heterogeneity* on different levels is crucial for SIG. For example, the health care organizations in a federal society like Canada are significantly different from one province to the next. Similar challenges exist in the European Union.

Currently available middleware solutions do not adequately meet such requirements. Therefore, we have embarked in a collaborative research initiative to develop middleware for SIG applications, primarily applied to the area of health care. The main collaborators in this so-called *medEmed* project are the University of Victoria (UVic) Department of Computer Science, the UVic School of Health Information Science and a local corporate partner, RPA Systems Farm Inc. This paper describes our approach to realizing a middleware suitable for SIG based on the concept of *active auditable Peer-to-Peer* (*aaP2P*) networks. *aaP2P* networks combine the key benefits of Peer-to-Peer (P2P) networks (evolvability) and the traditional middleware broker model (auditability). Moreover, unlike most currently existing middleware solutions *aaP2P* middleware components are *active*, i.e., the *aaP2P* middleware actively uses the federated information systems to execute information-flow processes rather than vice-versa. As a result, changing business processes and network topologies require little or no modifications to the implementations of federated information services. The rest of this paper is structured as follows: the next section introduces our work domain with a discussion of related work. Section 3 outlines main requirements on SIG middleware in health care that have driven our design and development. Then, Section 4 informally introduces the *medEmed* mediation technology framework including the *aaP2P* networks as the central part. Section5 continues with a more precise and formal definition of *aaP2P* networks grounded in Petri-net theory. Section 6 reports on a prototype implementation of the proposed middleware within Microsofts .NET^{TM} framework [8]. We evaluate our approach and point out next steps in Section 7.

2 Related Work

The research reported in this paper is related to many concepts developed over the last decade in the area of middleware and distributed object technology [7]. Particularly, the introduction and standardization of so-called Web service standards such as SOAP, UDDI and WSDL [4] has enabled the *medEmed* project. In contrast to traditional object middleware solutions such as CORBA and DCOM [7], Web service standards leverage pervasive communication mechanisms such as HTTP and XML [4]. Furthermore, Web service technology is more scalable

than traditional middleware since it does not rely on dedicated object request brokers or similar communication channels.

The idea of using Web service standards for developing SIGs is largely influenced by earlier research on grid technology [10]. Most of this earlier work has targeted the development of grid infrastructures for scientific applications such as meta-computing and data sharing, e.g., [13, 11]. Recently, the grid paradigm has been adopted on a broader scale for research and development of a "next generation network" for net-centric organizations [14]. This development has lead to the proposal of grid standard architectures such as the *Open Grid Service Architecture* (OGSA) and the Globus toolkit [9]. The development of the notion of SIGs as introduced in this paper was largely driven by requirements elicited from talking to stake holders in our application domain (health care) rather than by a particular standard architecture or framework. We had consciously chosen this scenario-driven approach to avoid constraining our thinking about solutions. Still, we are currently evaluating if our architecture can be re-implemented in compliance to the OGSA framework.

Our research is related to application-specific interoperability standards in our application domain such as HL7 (Health-Level 7) (www.hl7.org). HL7 is an international standard represented in Canada by the Canadian organization *HL7 Canada*. The current HL7 standard specification defines message formats on a very fine level of detail. HL7 is powerful but has not yet widely been adopted in Canada, partially because of its inherent complexity. Much of this complexity is required for message exchange within care-giving organizations but is unneeded for the exchange of more compound information objects among organizations and individuals, as envisioned for our SIG. Therefore, we have concentrated on the part of the HL7 standard that handles larger information granules, called structured documents in the *Common Document Architecture* (CDA). More precisely, CDA is still under development and yet has to be ratified by the HL7 organization. We are members of HL7 Canada and our work is based on draft proposals for the CDA and HL7 version 3.

One project that is tightly related to the *Inducer* concept in our *medEmed* initiative is the InterMed project [19] which is a collaborative effort headed by the Stanford Medical Informatics group. The InterMed project mainly focuses on developing tools for distributing medical guidelines and protocols. Various others project focus on integrating medical information systems with XML-based architectures. For example, the SynEx project [20] aims on creating a standard for medical data exchange. This European project has several objectives similar to *medEmed*, however, it takes a different implementation approach, namely focusing on multi-agent mediation architecture.

Finally, our research is influenced and inspired by works done in the domain of peer-to-peer (P2P) network systems [2]. The main properties we inherit from P2P systems are their decentralized character, scalability, and ability for self-organization. In fact, we are just beginning to explore the aspect of self-organization in the *medEmed* project and this paper contains few details about this. However, we do believe that self-organization is a key requirement on SIG

middleware technology (cf. next section). In contrast to most current P2P technologies, however, our approach provides mechanisms for auditing and certification of information exchange. The next section will systematically outline these requirements.

3 Requirements on a Health Information Grid

We have chosen health care as our application focus in order to attempt to alleviate some of the pressure caused by growing cost of treatment and the ongoing shortage of qualified personnel. Much like other public service areas, many medical organizations already use variety of *Medical Information Systems* (MIS) to manage so-called *Electronic Health Records* (EHR). Although their purpose might be similar, such MIS solutions are typically extremely heterogeneous. Moreover, health care systems differ from one Canadian province to the next. Consequently, public agencies have recognized the impracticality of engineering and imposing one single MIS throughout the country. The organizational structures of the different health systems involved and the pre-existing information technology used therein are just too heterogeneous.

The *medEmed* project has its focus on developing mediation technology that facilitates federation of new and existing MIS among each other with individual stakeholders devices (e.g., patients, clinicians). The key requirements on such mediation technology can be summarized in the following points:

Scalability Although these considerations are theoretical to date and our sample networks are much smaller, we expect that a successful SIG mediation technology must be able to span hundreds of thousands or even millions of nodes.

Adaptability/Adaptiveness A practical solution cannot neglect the fact that currently existing individual MIS installations are highly heterogeneous with respect to technical parameters as well as with respect to semantic issues of data representation. Therefore, adaptability (human-initiated customization) as well as adaptiveness (automatic adaptation) are a key prerequisites for the adoption of SIG technology.

Evolvability is a requirement implied by the pervasive nature and scale of medical SIG. Evolution will mainly happen in two directions: evolution of *business processes* (e.g., services are provided by different laboratories, new data warehouses are created for research and analysis etc.) and evolution of network configuration (e.g., service providers and consumers dropping in and out). The choice of mediation technology should impede evolution as little as possible.

Activeness This issue is related to the previous two points. It acknowledges the fact that traditional middleware technology has been of predominantly passive nature; that is, distributed business processes have been largely executed by applications using the middleware for requesting remote services. While this approach has been satisfactory for smaller, more confined distributed applications, it seriously impedes evolvability and adaptability of

SIG applications. Therefore, global aspects of business processes should be enacted by the middleware itself, i.e., the middleware should be *active*.

Security/Privacy are naturally important requirements on any SIG that deals with personal information. This issue becomes even more sensitive when it comes to the distribution of personal medical data. In this paper, we will somewhat touch on privacy issues with respect to fulfilling the next requirement (accountability). However, a detailed discussion of security mechanisms like authentication and encryption is beyond this papers scope.

Accountability The requirement for accountability is mainly derived from the above privacy requirement. Many societies have given governmental organizations the mandate to ensure that private information is distributed with the consent of individuals only. For this purpose privacy legislations often imply the management of a complete audit trail of the distribution of patient information (e.g., article 25 of the British Columbian privacy legislation). Moreover, individuals should be given effective mechanisms for determining exactly where their private information was distributed (and withdrawing that information if desired). Furthermore, middleware accounting mechanisms can also become a basis for claiming expenses by service providers.

Reliability is a critical requirement in an application domain that impacts human well-being such as health care. Single points of failure are not acceptable. The performance of the SIG should only decline gracefully (and slowly) with individual servers or connections becoming unavailable.

We will describe the conceptual architecture of a middleware solutions that has been designed to meet the above requirements.

4 SIG middleware components

The SIG middleware architecture developed in the *medEmed* project consists of three main concepts named *Grid Federation Envelope* (GFE), *Adaptive Process Middleware* (APM) and the *Medical Exchange Agency* (MEA) (cf. Figure 1). Although this paper has its dominant focus on the APM internals, a short introduction of the other parts of this architecture contribute to the understanding of our approach.

4.1 Grid Federation Envelope

The GFE plays the role of an adapter that interfaces various types of heterogeneous information sources to a standardized interface based on Web service technology. The GFE uses the Simple Object Access Protocol (SOAP) and XML [4] to communicate with the APM. The GFE has a plug-in architecture for various kinds of import/export engines (so-called ImpEx conduits). Each ImpEx conduit can convert native data formats to XML based on the aforementioned HL7 Common Document Architecture, and process CDA related queries against the native data content. To date, we have implemented ImpEx conduits for MS

SQL Server, MS Access, Oracle 9i, and file directories with MS Word (text) documents. A more detailed description of the GFE architecture is beyond the scope of this paper. However, details about a previous version of GFE have been reported in [3].

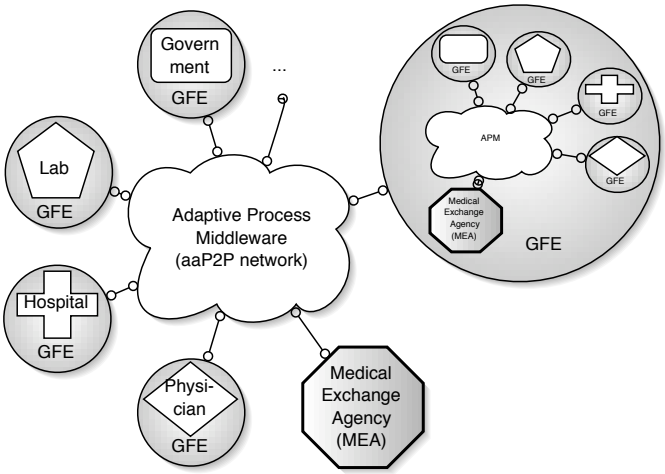


Fig. 1. Conceptual architecture of SIG middleware in *medEmed*

4.2 Adaptive Process Middleware

The APM transports information among various kinds of integrated medical organizations, e.g., hospitals, physicians offices, caregivers, laboratories, insurance carriers, research institutes etc. All information routed by the APM is represented in XML. The APM is *active* in the sense that it can *pull* information from organizations, combine it with information from other sources, translate and enrich it, and *push* it on databases in organizations. Internally, the APM is based on predefined, customizable mediation components and realized as a P2P network of such component instances. Component types are further classified as being *active* or *passive*, respectively. The current APM supports five active component types:

- The *Initiator* is the simplest APM component. Essentially, it allows pulling information units from an input component object and pushing it to an output component object, upon occurrence of a specified event.
- The *Transducer* converts (input) information of one specific representation to (output) information of a different representation.
- The *Merger* combines two (input) information units of (partially) different structure into a combined (output) information unit.

– The *Adder* adds two (input) information units of the same structure into a combined (output) information unit.
– The *Inducer* reads (input) information units and enriches its content with additional information derived by a decision support system. *Inducers* can be based on formalized practice guidelines, e.g., using GLIF and Arden syntax [12].

Initiators require events as triggers for pulling information units from input components. (These events are specified separately for each Initiator.) All other active components perform their operation upon arrival of all required input information units.

The passive component type supported in the APM is called *Staging Area*. Staging Areas are used as temporary memory of information units in the APM for further processing. Staging Areas have been introduced because APM components perform asynchronously and concurrently. Although the six mentioned component types are relatively simple, complex information mediation networks can be built by combining them. We use directed graphs to visually represent APM component object configurations. For example, Figure 2 shows a small sample configuration involving four organizations (blue triangles). Staging Areas are represented as circles and active components are represented as rectangles. Component types are visually distinguished by means of symbols and colors: the four Initiators are marked by a right-pointing triangle on red background, the Adder is labeled with a plus sign, the Inducer contain the icon of a book (symbolizing its knowledge-based character), and the remaining two active components are Transducers symbolizing structural conversion.

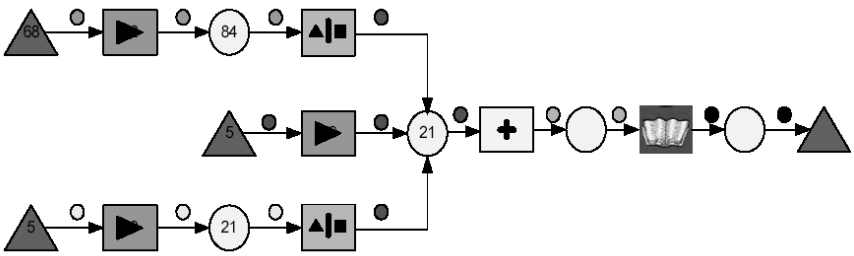


Fig. 2. Sample APM component configuration

The arrows in Figure 2 represent channels that indicate the directions in which information units flow through the APM component network. Note, that we use the abstract term *information units* because we do not want to unnecessarily constrain the application of our middleware to exchanging medical CDA documents only. Among the many other possible types of information units are for example statistical data on drug effectiveness etc. Consequently, there is the necessity for distinguishing between different types of information units. This

necessity would even exist if the middleware were applied in a specific application domain only, e.g., EHRs. Therefore, the connections in our APM network are typed, i.e., labeled with the type of information units that can travel through them. Visually, such labels are represented as small circles with different colors in Figure 2 (difficult to see on black & white copies of this paper). Semantically, these types might correspond to structured document types in the HL7 CDA or other standards.

Specification of APM networks
The APM is realized as a P2P network. Hence, deployment and administration of APM components is done in a decentralized way: generally, there is no single organization deploying and administering all component objects in the APM, nor will all component objects be hosted on the same server machine. Consequently, the network in Figure 2 should be understood as being merely a small excerpt of the entire APM network, which is administered by one particular organization. As an analogy, the reader might think of power line infrastructure or telephone line networks. This particular example (Figure 2) has been taken from one of our application case studies in the area of palliative care. The rightmost organization represents a data-warehouse at a national research institute for researching the effectiveness of pain relieving drugs. The three organizations on the left-side are hospices specialized in end-of-life care and cancer treatment. The three Initiators on the left side have been set up to pull updated information on patient conditions, drug interventions and outcomes in regular intervals (timed events). Before the Adder can combine this information, it has to be converted into a common format, called the Palliative Data Set. The combined information then runs through the Inducer component, an expert system that adds additional derived information content. The resulting information is then imported to the data warehouse at the governmental research lab.

Section 6 will report details on the implementation of tools for deploying and administering the APM network.

4.3 Medical Exchange Agency

An MEA is a dedicated Web repository for medical information that fulfills four main purposes:

Certification Each single APM component object has to register with an MEA. (The *medEmed* middleware architecture allows for multiple MEA servers.) On activation time, component objects check with their MEA whether their certificate is still valid. The component-object will remain inactive if this is not the case. This provides a mechanism for withdrawing permissions for accessing confidential information. Still, in order to avoid a single point of failure in the *medEmed* middleware, APM components are designed to keep operating for a specified period in order to bridge temporary periods when the MEA is unavailable (e.g., due to network faults).

Auditing Each MEA observes the activities of all APM component objects registered to it. In particular it stores meta-information about information

units that flow through those component objects. Essentially, this meta-information provides audit trails about the distribution of confidential information. Auditing messages to the MEA are persistently queued at each APM component object in temporary periods of MEA unavailability.

Scalability Hierarchies of MEAs can be used to federate local communities of APM peers into larger communities. For this purpose, local MEAs assume the role of warehouses for collecting medical data from the local network, e.g., district or province. These MEAs can then be integrated into a more global APM network, e.g., on the national or international level (cf. Figure 1).

Knowledge Repositories MEAs serve as knowledge bases for customizing and updating the practice guidelines enacted by Inducer components. Apart from these middleware-related aspects, MEAs also provide Web-portals for supporting clinicians with the latest research results and recommendations. Knowledge will be stored in the MEA in form of GLIF documents. A detailed description of the MEA is out of this papers scope.

5 Interaction Semantics

In this section, we will define the proposed *aaP2P* middleware more precisely with a formal model. Rather than representing an academic exercise, the critical nature of the chosen application domain strongly suggest such precision. Our formalization enables us to reason about properties of interaction protocols, e.g., with respect to security, privacy, reliability, reachability, conflicts, etc. While these types of analyses or not included in this paper, it shall provide the necessary fundament for such (currently ongoing) studies. The following definition formalizes the signature (syntax) of *aaP2P* middleware graphs.

Definition 1 (*aaP2P* middleware signature).
An active, auditable peer-to-peer middleware signature is a 9-tuple aaP2P*:* $(\triangle, \bigcirc, \square, L, \Rrightarrow, \divideontimes, \Omega, \omega, \zeta, \rho))$ *, where*

- $\triangle$ *is a finite set of* peers *representing organizational data sources.*
- $\bigcirc$ *is a finite set of* Staging Areas*.*
- $\square$ *is a finite set of* active components*.*
- L*:* $\square \longrightarrow \{$ *Init, Merg, Add, Trans, Ind* $\}$ *an exhaustive* labeling function *for associating components with their respective type;*
- $\Rrightarrow$ *is a finite set of* channels*, with* $\Rrightarrow \subseteq \bigcirc \times \square \cup \square \times \bigcirc \cup \square \times \triangle \cup \triangle \times \blacksquare$*, and* $\blacksquare := \{\square_i | \square_i \in \square \wedge \square_i = \text{"}Init\text{"}\}$*.*
- $\divideontimes$ *is a finite set of* auditing servers*.*
- Ω *is a (possibly infinite) set of* information unit types*.*
- $\omega : \Rrightarrow \longrightarrow \Omega^2$ *is a* typing function *mapping connections to sets of information unit types.*
- $\zeta : \triangle \cup \bigcirc \cup \square \cup \divideontimes \longrightarrow \aleph$ *is an injective* identification function *uniquely mapping aaP2P elements to natural numbers.*
- $\rho : \square \longrightarrow \divideontimes$ *an exhaustive* registration function *mapping active components to auditing servers.*

The formalization of the interaction semantics of *aaP2P* signatures is grounded in the theory of *Petri-nets* (PN). Since Petri's initial formalization, many extended classes of PN have been proposed [17]. Recently, there has been effort in standardizing notation and semantics of (high-level) PN [1]. (The standardization process is still pending, respectively, it has been re-instated in 2002.) Our formalization is based on this standards proposal. More precisely, the standardization proposal contains two alternative formalizations of Petri-nets. The formalization adopted in this paper is called *High-Level Petri-net Graph*. For simplicity we will refer to it as Petri-net (PN).

Definition 2 (Petri-net). *A* Petri-net *(PN) is a tuple* $PN : (NG, Sig, V, H, Type, AN, M_0)$ *where*

- $NG := (P, T; F)$ *is called a net graph, where*
  - P *is a finite set of nodes, called* places*.*
  - T *is a finite set of nodes, called* transitions *with* $P \cap T = \emptyset$*.*
  - $F \subseteq (P \times T) \cup (T \times P)$ *is a set of* arcs*, also called* flow relation*.*
- $Sig := (S, O)$ *is a* Boolean signature*.*
- V *is an* S*-indexed set of* variables*.*
- $H := (S_H, O_H)$ *is a many sorted algebra for the signature* Sig*.*
- $Type : P \longrightarrow D$ *is an exhaustive* typing function *used to assign types to places.*
- $AN := (A, TC)$ *is a pair of net* annotations *where*
  - A *is a function that annotates each arc with a term that when evaluated (for any binding) results in a multiset over the associated places type. Formally,* $A : F \longrightarrow TERM(O \cup V)$ *such that for all* $(p,t),(t',p) \in F$*, for all bindings* $\alpha, Val_\alpha(A(p,t))$*,* $Val_\alpha(A(t',p)) \in \mu Type(p)$*.*
  - $TC : T \longrightarrow TERM(O \cup V)_{Bool}$ *is a function that annotates transitions with Boolean expressions.*
- $M_0 : P \longrightarrow \bigcup_{p \in P} \mu Type(p)$ *such that* $\forall p \in P, M_0(p) \in \mu Type(p)$*, is the initial marking function which associates a multiset of tokens (of the correct type) which each place.*

Note, that according to Annex A in [1], the term *Boolean signature* is used to mean a many-sorted signature where one of the sorts is *Bool*, corresponding to the carrier Boolean in any associated algebra. A Boolean signature will also include the constant $true_{Bool}$.

We inherit the execution semantics formalized in Clause 9 of the PN standard proposal [1]. Therefore, we will not re-iterate PN theory, but we will use the rest of this section for formalizing how the introduced middleware model derives its semantics from PN theory. We do this by defining a mapping from *aaP2P* middleware signatures to a PN, which defines the corresponding semantics. We use *graph rewriting* [18] as a technique for defining this mapping in a precise yet easily understandable format. Graph rewriting techniques are usually applied to the abstract syntax graphs (ASG) of languages [6]. For the purpose of this paper, however, we have chosen to define graph rewriting rules directly

on the concrete syntax graphs (CSG). Using CSGs is appropriate because both (source and target) formalisms have graph-based concrete representations and their translation into a generic ASG format is straight forward. Skipping the definition of ASG models allows us to do a more concise (and intuitive) presentation of the formalism mapping in this paper.

We assume that the reader is familiar with the graphical representation of PNs. The CSG representation for *aaP2P* middleware signatures has been introduced in Section 4 (Figure 2) except for the visual representation of *auditing servers* (cf. Definition 1). As mentioned earlier, auditing servers are hosted by the MEA components in our network. For the purpose of this formalization we make the simplifying assumption that there be only one MEA component in a *aaP2P* middleware signature. (The extension to many MEAs is straight forward but resulted in a more complex set of graph rewriting rules.)

Figure 3 shows seven graph rewriting rules that perform the semantic mapping of *aaP2P* middleware signatures to PN graphs. Each rule has a *left-hand side* (LHS) and a *right-hand side* (RHS). The rules are defined in a way such that their LHS shows concepts of the *aaP2P* middleware CSG and their RHS shows the corresponding semantics mapping to PN graphs. Note, that dotted lines are used to distinguish those graph elements that are replaced in each rule from those elements that form the context of the rule. The intuitive semantics of an application of a graph rewriting rule is defined as follows (see [18] for a formalization):

Definition 3 (Application of a graph rewriting rule).
A graph rewriting rule r is applied to a given host graph G by performing the following steps. (r can only be applied to G if all of the following steps are applicable for G.)

MATCH Find two subgraphs G_l and G_r in G such that
- *there is an isomorphism m_l between all elements in LHS(r) and G_l, and*
- *there is an isomorphism m_r between the solid-line elements in RHS(r) and G_r.*

Note that the two subgraphs G_l and G_r overlap in elements that appear on both sides of r. (G_l, G_r) is called a match *of r in G.*

CUT Remove all graph elements in G_l from G, which have been matched to dotted elements in LHS(r).

PASTE Add the dotted graph elements on RHS(r) to G and glue them to the context*, i.e., those elements in the match of r, which have been mapped to solid-line elements in r.*

ASSIGN Assign labels and attributes of new graph elements as indicated in the RHS of r.

We will now discuss each depicted graph rewriting rule in detail.

Rule *MEA* creates four PN places and one transition for each auditing server (hosted by an MEAs). The four places can be seen as four information interfaces of the auditing server. According to our requirements, the two main purposes of an auditing server is, firstly, to maintain audit trails of actions in the

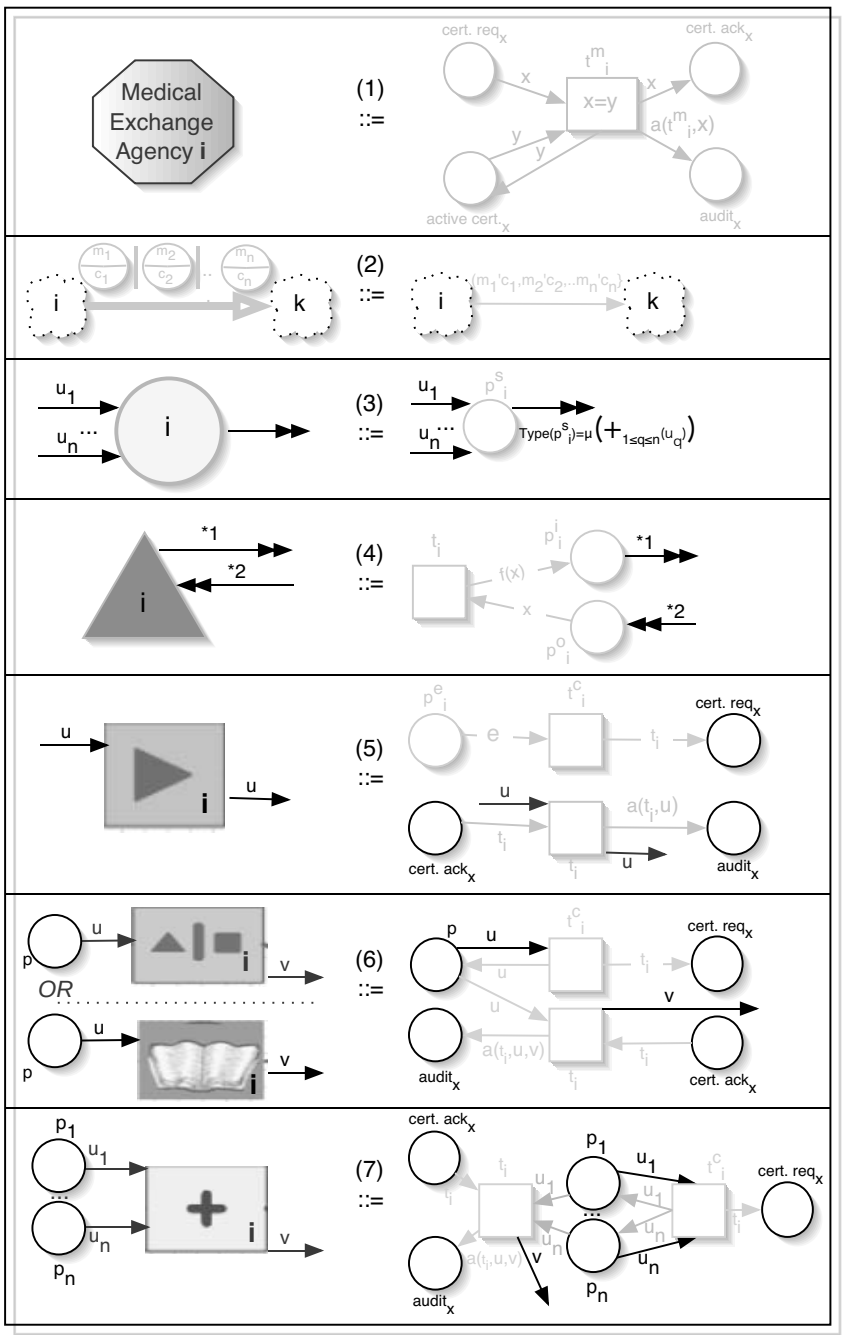


Fig. 3. Semantic mapping of *aaP2P* signatures to PN graphs

aaP2P middleware and, secondly, to give and withdraw certifications for APM components. The place *active cert*$_i$ represents the pool of currently issued and active certificates. (Certificates can be added and withdrawn by the MEA administrator using the APM admin tool described in Section 6.) Upon activation of an APM component, place *cert. req*$_i$ is used as an interface to identify this component and request confirmation of certification from the auditing server. The boolean condition attached to transition T_i^m ensures that this confirmation is given only if the requesting component is in the pool of currently certified components. In this case the confirmation is represented by a token in place *cert. ack*$_i$.

Rule *Channel* translates a channel in the *aaP2P* signature to an arc in the PN graph. Note, that we use clouds as wild card symbols for different kinds of objects in *aaP2P* networks. Rule *Channel* does a trivial replacement but rewrites the annotation of the APM channel to an arc annotation in the PN graph. In our *aaP2P* CSG, we use inscribed circles for specifying information unit types that will travel over corresponding channels, where different information types are distinguished by different colors. Since we cannot distinguish different colors in print matter, we represent different types with the lower inscribed number in rule *Channel* $(c_1, ..., c_n)$. The upper number determines the multiplicity for each information unit type, i.e., how many information units of the corresponding type are transported during one transaction. The RHS of rule *Channel* rewrites this notation to standard PN arc typing for typing token multisets [1].

Rule *SA* trivially rewrites Staging Areas in the middleware signature as places in the PN. Note, that arc labels $u_1, ..., u_n$ are used as placeholders for the actual arc typing. The double-arc is used as a shorthand for denoting the set of all out-going arcs connected to a Staging Area.

Rule *Org* rewrites organizational data sources in the middleware signature as two places and one transition. Place p_i^i represents an information input port of that organization. Upon reception of information units (PN tokens), the transition might respond with the generation of information units, which are then transmitted as tokens in place p_i^o. Analogously to the previous rule, we use double-arcs to denote the sets of all out- respectively in-going arcs of an organization. Note that the upper arc set on the LHS corresponds to the upper arc set on the RHS.

Rule *Init* is slightly more complex. Essentially, a initiator is an APM component that, upon occurrence of certain external events, requests specific information units from an organizational data source and pushes it onto its target component. Examples for external events are automatic timers, user requests, signals from third-party applications etc. The external event interface is represented by place p_i^e in rule *Init*. Upon occurrence of an external event (e), the Initiator does two things by means of transition t_i^c: Firstly, it requests confirmation of certification from the MEA (by putting its identification token t_i in place of *cert req*$_x$). Secondly, it sends a token to the organizational data source (p_x^o) that described the desired information unit. (In our implementation, such query tokens like actual information tokens are rendered in XML.) Upon reception of

the desired information unit (p_x^o) and confirmation of certification (*cert. ack*$_x$), the Initiator moves the information unit to its target component and sends an audit trail token to the registered MEA (*audit*$_x$).

Rule *Trans* shows that Transducer and Inducer components mapped to the same structures in the PN. Upon availability of information units of the right type in the source component (p), Transition t_i^c issues a request for confirmation of certification. The information units of interest are held back at the source component (Staging Area) until this conformation is received (*cert. ackx*). Then, a translated (respectively augmented) information unit is passed on to the target component.

Rule *Add* is analogous to rule *Trans* but considers multiple source components $p_1, ..., p_n$.

Now we have the prerequisites for defining the interaction semantics of a *aaP2P* middleware configuration. As a notational short hand for the following definition, let us denote $R^*(G)$ for the maximum possible number of concatenated applications of any rewriting rule R to an initial source graph G, i.e., $R^*(G)$ is inductively defined as

$$R^*(G) := \begin{cases} G & if \neg def(R(G)) \\ R \circ R\ (G) & else \end{cases}$$

Definition 4 (*aaP2P* Interaction Semantics). *The interaction semantics of a* aaP2P *middleware signature* $M : (\triangle, \bigcirc, \square, L, \Rrightarrow, \divideontimes, \Omega, \omega, \zeta, \rho)$ *(with one single auditing server) is defined by the PN graph G:(NG, Sig, V, H, Type, AN,* M_0*) that is constructed as*

$$G := MEA \circ Channel^* \circ SA^* \circ Org^* \circ Init^* \circ Trans^* \circ Add^*(M).$$

6 Prototype

In this section we describe the current prototype of the *medEmed* mediation technology. First we define the scope of the current implementation as well as name and rationalize system requirements that are reserved for future development phase. Secondly, we specify the overall architecture of the system including description of the core functionality and interoperability of the three main medEmed technologies: GFE, APM, and MEA. Furthermore, throughout this section we also focus on the system administration aspects that were developed in the current prototype.

6.1 Scope

The current prototype demonstrates implementation of a distributed component infrastructure for the large-scale sharing and manipulation of medical information resources. The existing *medEmed* system consists of MEA and GFEs that provide medical data to the APM components via Web services. The interoperability of these components allows the execution of predefined workflows for

distribution, merging, addition, enrichment, and translation of medical data. The prototype also includes the implementation of administration components that have an ability to monitor, customize, and supervise *medEmed* components.

Because the focal point of the initial prototype was satisfying major functional requirements of the system, several non-functional requirements such as security, privacy, and accountability are not full-fledged in the current implementation. However, it is important to note that these requirements are of a critical importance, and while they have not been fully implemented yet, they were thoroughly considered in the previous design cycle of the *medEmed* system and will be matured in the next implementation phases.

6.2 Architecture

Because *medEmed* architecture is based on component-oriented approach, it is flexible, open, extensible, and accommodates a plug-and-play paradigm. Each component is an independent and customizable service that can be used in various coordinated work flow instances. The current prototype of the *medEmed* system uses test data from our Palliative Care use case and functions in academic setting. This sample data originated from a hospice in Victoria but has effectively been anonymized to protect privacy. Figure 4 shows a high-level view on the overall system architecture implemented up to date. (For this diagram, we use the notation suggested by Cheesman et al. [5].)

The base parts of the *medEmed* architecture are GFEs that are essentially information producers and consumers. In this P2P setup they operate as nodes in the grid networks. GFEs provide .Net Web interfaces for managed data access (part of IAPMServices interface). These interfaces are implemented for processing XML queries structured around the HL7 Reference Information Model (RIM) and the HL7 CDA. However, besides providing conduits into native data sources and performing data standardization via system wrapping, GFEs also allow its underlying medical organizations act as hosts to APM components.

APM components are also implemented as .Net Web services and contrary to traditional middleware concepts they proactively participate in information workflows. Active APM components such as Initiator, Transducer, Merger, Adder, Inducer and passive Staging Area component implement an Observer Pattern and perform their tasks based on triggers. Because the APM networks are based on Petri-net theory, the primary triggers are typically data or query tokens. Still, other events such as external triggers and user interactions can generate "virtual tokens" to initiate APM operations.

While all the APM components play important role in how critical medical data is used, the most complex data manipulation takes place in the internals of the Inducer component. The Inducer deals with aspects of interoperability such as semantics of data, knowledge base, and interpretation of complex medical logic. Because the function of this component is to enrich data, it has a responsibility to provide decision support for medical professionals to the best of its abilities.

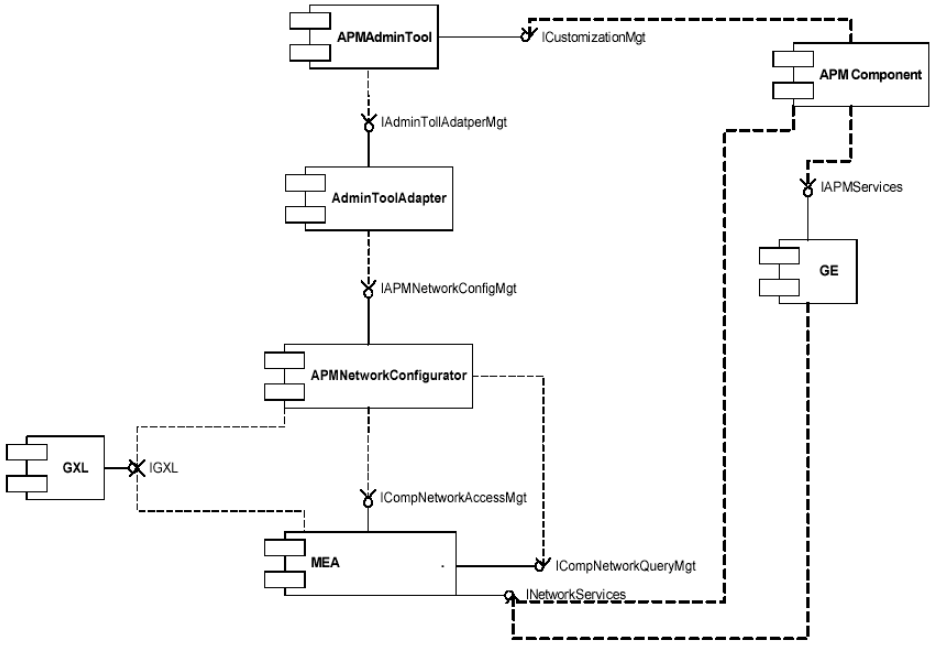


Fig. 4. Component Specification Architecture for medEmed (excerpt)

Currently the backbone of the Inducer component is Protege which is an ontology and knowledge-base editor [15]. In the current implementation Proteges abilities are extended with Odin and GLIF plug-ins. Odin is a component that has been developed in the context of *medEmed* project and is able to generate XSLT scripts for translation among different Protege ontologies. GLIF (Guideline Interchange Format), on the other hand, is a formalization standard for medical guidelines.

Another aspect that was implemented in the current prototype is system administration. Because the *medEmed* is a system that may consist of potentially very large number of components or nodes on a self-guiding grid network, it is important that its performance and configurations be monitored and regulated. Two types of system administration tools should be distinguished: the *APM Admin Tool* and the *MEA Admin Portal*.

For the current implementation the MEA Admin Portal is very rudimentary. MEA serves as a repository of network configurations and as an auditing server and, thus, requires only simple customization properties. On the other hand, the APM Admin Tool has the ability to configure individual components as well as network layers. Both mechanisms provide basic levels of control of information distribution and use which has a critical importance in such content-sensitive environment as healthcare. The user interface of the APM Admin Tool has been

realized within Microsoft Visio 2000 (cf. Figure 5). Most other components have been implemented using the .Net framework and C#.

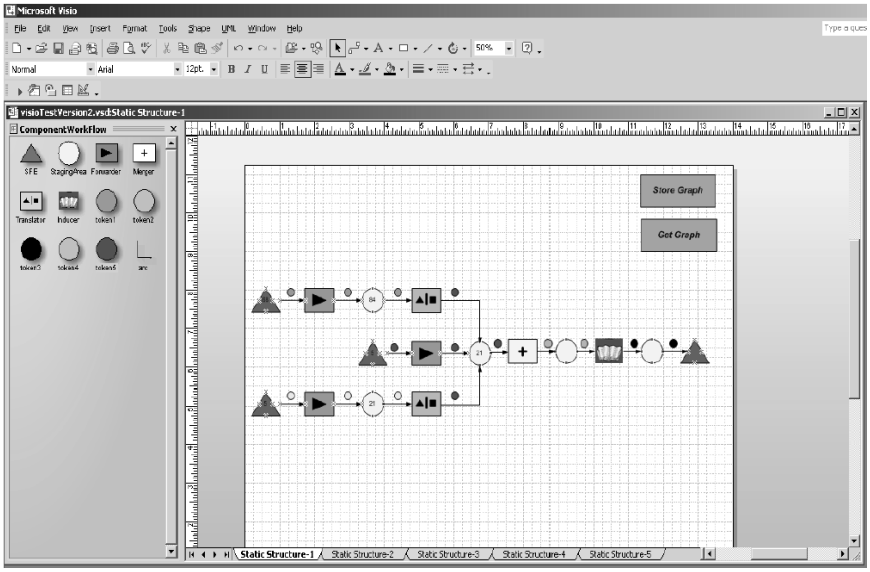


Fig. 5. Main User Interface of the APM Admin Tool

In the future, the MEA Admin Portal will significantly be extended and will provide advanced audit trails, component certification, and policy enforcements regarding data access constraints.

7 Evaluation

7.1 Theoretical Considerations

Clearly, the *medEmed* project is far from being finished and does not yet satisfy all requirements outlined in Section 3. Still, our preliminary results show much theoretical promise. The inherent parallelism of Petri-nets (fundamental for *medEmed*) will allow for great scalability. Moreover, the network is reliable with respect to our requirement for graceful degradation (cf. Section 3). Auditing servers (within MEAs) are the only critical parts of the *aaP2P* network in this regard. In principle, according to our formal model, all APM components stop working when the MEA they are connected to is becoming unavailable. There are several possibilities to avoid this problem. One solution is to add redundancy by assigning multiple MEAs to APM components. However, in this case, the auditing servers in MEAs assigned to the same APMs would have to be synchronized regularly to keep audit trails consistent.

A better alternative would be to use a technique we refer to as *temporary rumor backup* (TRB). In this case, APM components are assigned to a single auditing server only. When this auditing server becomes unavailable, the assigned APM components continue to operate for a finite time period in TRB mode. In this mode, they send auditing trails to their known peers, which in turn forward them to their known peers etc. up to a predefined maximum number of hops. This strategy of temporary redundancy of data storage provide a high amount of insurance that the audit token is not "forgotten" even if the original APM component drops out. When the audit server comes back online, TRB provides mechanisms for forwarding this audit information to its original destination.

A requirement that we yet have to address is adaptiveness, i.e., self-adaptation of the middleware. The component-based graphical deployment of the middleware facilitates adaptability and evolution of networks. However, our current research is on developing mechanisms for automatic network re-configuration. We expect that the formal fundament of our middleware will facilitate the formalization of configuration invariants that can govern the self-adaptation.

Electronic information exchange processes are executed and initiated by the middleware itself and do not require changes to federated legacy systems ("activeness"). Security certificates and privacy concerns (auditability) are addressed with auditing servers. Moreover, we believe that the formal semantics of the underlying *aaP2P* model will enable the development of tools that support SIG administrators in reasoning about important properties of networks such as liveliness and process invariants. This aspect of our research is current work.

7.2 Practical evaluation

Evaluation and practical case studies are crucial for judging the merits of the results produced and establishing credibility in our chosen application domain. The *medEmed* project plan contains three main evaluation phases. The results of each phase feed back into creating the next version of the prototype technology. The phases are as follows:

Simulated case study Set up and simulate a heterogeneous, distributed sample MIS network at the University of Victoria. This network will be distributed among different departments (CS and HINF) and involve different types of data bases and vendor MIS. For this purpose, RPA Systemsfarm Inc. has granted us access to a license of their product *eChart*. The setup will be driven by practical sample scenarios taken from the area of information exchange in palliative care. Overall, this phase is considered successful if our system adequately supports the scenarios specified by domain experts.

Moderated case study Set up a middleware among actual medical organizations and use it for real information exchange. Here, our staff would do the administration and maintenance of all software components. The phase is considered successful if stakeholders at the participating institutions perceive the mediation technology as an significant added value.

Monitored case study This phase would guide external stakeholders in deploying and administering the mediation technology for supporting custom information exchange scenarios. We consider this step successful if computer-literate domain-experts are able to effectively deploy and maintain the middleware.

To date, we have finished the first phase of this practical evaluation with respect to sample scenarios from the domain of palliative care. The experiments have shown that our technology adequately supports "sequential" data query, merging and collection processes. Figure 2 shows an example for such a process that essentially queries, merges and stores data from a set of organizations in another organization. However, we noticed difficulties constructing networks for more iterative and interactive scenarios. Essentially, our current approach becomes problematic when there are information units that need to be routed from a queried organization back to the querying site. Although, our current case study scenario (palliative care) does not need highly interactive queries, we anticipate such requirements will arise when we broaden our focus. Thus, we will have to address the problem in the future. The rest of this section gives a short overview of our current case study domain.

7.3 Case Study: Palliative Care

The ability of a caregiver to make an informed decision regarding a patient's case is often dictated by the amount of relevant information upon which the decision is based. Certainly a caregiver will always try and make the best decision possible, with all issues considered, but there are times when decision-making is limited by the lack of availability of information. Within healthcare in Canada, we are seeing regionalization of health services, and subsequent changes in how healthcare is delivered. Certain treatments or surgeries may only be available in certain areas and result in patients having to travel, possibly to more than one location, to obtain the necessary medical services. A disadvantage of that scenario is the patient's medical data may be spread over several clinical systems, and a caregiver making a clinical decision using only part of the data may not be making the most informed decision possible.

medEmed hopes to become involved with the Canadian Palliative Surveillance Data Network project, which is a project attempting to establish common palliative definitions, indicators and data sets to move towards national palliative data surveillance. (The UVic School of Health Information Science is already involved in this project.) Part of that project involves bringing together data from multiple palliative care sites across Canada and creating a data warehouse where the data can be stored and analyzed. But the heterogeneity of data from different sources can make data comparisons and analysis difficult. The *medEmed* components described in this paper present a practical solution to that problem, as it would allow us to capture, merge, translate and store multiple heterogeneous data sets into a common database schema. A common schema would allow more meaningful data analysis and surveillance to take place. Furthermore, clinical guidelines or other clinical information documents that are developed from the

data analysis could be wrapped in a GLIF token and distributed back to the palliative sites for review and possible integration into the delivery of palliative care.

8 Conclusions

The realization of a pan-Canadian health information highway as envisioned by Health Canada poses many challenges. In this paper, we have identified a catalogue of key requirements on any mediation technology suitable for this purpose. In the *medEmed* project, we have made an attempt to pull together an interdisciplinary group of domain experts, software engineers and MIS vendors to jointly develop a prototype solution for this challenge. The current *medEmed* prototype is the preliminary result of this effort. We believe that it implements interesting concepts that could be significant as a basis for realizing medical SIG as well as SIGs in other domains. We will build on our current results and continue to make the *medEmed* middleware more self-adaptive. We will also intensify our efforts in realizing a powerful Inducer component based on a GLIF execution engine.

Acknowledgements

We thank the Advanced Systems Institute of British Columbia (ASI), the National Science and Engineering Research Council (NSERC) and RPA Systems Farm Inc. for funding this research.

References

1. *High-level Petri Nets - Concepts, Definitions and Graphical Notation.* Final Draft Intl. Standard ISO/IEC 15909, online at http://www.daimi.au.dk/PetriNets/standardisation/, 2000.
2. K. Aberer. P-Grid: A self-organizing access structure for P2P information systems. *Lecture Notes in Computer Science*, 2172, 2001.
3. Y. Bychkov and J. H. Jahnke. Varlet/babel: Toolskit for net-centric legacy data integration. In *Proc. o 3rd Intl. Workshop on Net-Centric Computing (NCC'01), Toronto, May 2001*, 2001.
4. E. Cerami. *Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI & WSDL.* O'Reilly & Associates, Inc., 103a Morris Street, Sebastopol, CA 95472, USA, Tel: +1 707 829 0515, and 90 Sherman Street, Cambridge, MA 02140, USA, Tel: +1 617 354 5800, 2002.
5. J. Cheesman and J. Daniels. *UML Components: A Simple Process for Specifying Component-Based Software.* Addison-Wesley, 2000.
6. H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 2: Applications, Languages and Tools.* World Scientific, Singapore, 1999.
7. W. Emmerich. *Engineering Distributed Objects.* Addison-Wesley, 2000.

8. K. H. et al. *Professional .NET Framework*. Wrox Press Ltd., 2001.
9. I. Foster and C. Kesselman. The globus toolkit. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, pages 259–278. Morgan Kaufmann, San Francisco, CA, 1999. Chap. 11.
10. I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organization. *The International Journal of High Performance Computing Applications*, 15(3):200–222, Fall 2001.
11. D. Gannon. Software component technology for high performance parallel and Grid computing. *Lecture Notes in Computer Science*, 2150, 2001.
12. S. Hoelzer, R. Schweiger, and J. Dudeck. Representation of practice guidelines with xml - modeling with xml schema. *Methods Inf Med*, 41:305–312, 2002.
13. V. Huber. UNICORE: A Grid computing environment for distributed and parallel computing. *Lecture Notes in Computer Science*, 2127, 2001.
14. C. Kesselman. The Grid, Grid services and the Semantic Web: Technologies and opportunities. *Lecture Notes in Computer Science*, 2342, 2002.
15. N. F. Noy, R. W. Fergerson, and M. A. Musen. The knowledge model of protégé-2000: Combining interoperability and flexibility. *Lecture Notes in Computer Science*, 1937, 2000.
16. F. A. C. on Health Infostructure. Tactical plan for a pan-canadian health infostructure. Technical report, Office of Health and the Information Highway, Health Canada, November 2001.
17. W. Reisig. *A Primer in Petri Net Design*. Springer, Berlin, 1992.
18. G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. I: Foundations*. World Scientific, 1997.
19. D. Wang, M. Peleg, S. W. Tu, A. A. Boxwalla, R. A. Greenes, V. L. Patel, and E. H. Shortliffe. Representation primitives, process models and patient data in computer-interpretable clinical practice guidelines. Technical Report SMI Report Number: SMI-2002-0929, Stanford Medical Informatics, Stanford University, Stanford, CA 94305, USA, 2002.
20. Y. Xu and P. Degoulet. Using xml in a component based mediation architecture for the integration of applications. In *Proc. of XML Europe 2001, Internationales Congress Centrum, Berlin, Germany*, 2001.