Ride a Cock-horse ...

John ffitch and Julian Padget

Department of Computer Science, University of Bath, UK
(jpff,jap)@cs.bath.ac.uk

Abstract

We present a general architecture for a net-wide musical service, following previous work on mathematical services on the 'net. The structure and components of this architecture are identified, and a number of sample applications are presented, to show the scope of this architecture.

One aim of this proposal is the option of having music where ever we go, whether delivered on a workstation, laptop or palm-held device. This opens up a number of possibilities for performance or personal delight.

1. Introduction

We are considering what facilities are needed in a widely based sound and music system, using thin clients such as palm-held devices. Our starting point was in the design of a system to help in the composition of computer music and also to provide an interactive listening environment. That system has been described elsewhere[1], and work on it continues.

Starting from the MUSIC V concept of separating the *orchestra* from the *score* we are developing tools to perform simple score editing [2], using agent technologies. Our architecture assigns agents to each instrument used in the piece of music, which could then interact with each other and the composer of the music. This would allow a higher level of control, and more control, over the piece being composed.

A conductor agent provides the interface between the user and the system, which then instructs performer agents when and how to play the requested piece of music. The performer agents change their respective parts of the score in response, and pass their output to a mixer agent. This mixer agent then sends the completed (merged) score to Csound to generate the piece.

As this work continues it became clear that we need to develop a detailed ontology in which to express the modifications, and in particular an ontology of time.

Partly in response to the needs of distributed users we also created the NetCsound facility[3] where users can ask for pieces to be synthesised and delivered to a web browser. This powerful but simple facility led to the question of what would a full web-based synthesis system look like. This paper provides a possible answer and in so doing, drawing analogy from mathematical-based web services[4], proposes a program of work for musical web services.

2. MONET

The Monet project (Mathematics On the NET) was a step toward weaving together the emerging world of web services – the semantic web – and grid computing, in that the aim is deliver sophisticated mathematical problem analysis and the code to compute the answers, for which grid services will be required, in a common open agent-based framework for the description and provision of web-based mathematical services.

The application area is mathematics, and is deeply involved in representing the semantics of mathematical notation, and the automated use of computer algebra system. The area may differ from music, but we content that there are lessons to be learnt from their experience.

In the MONET architecture (see Figure 1) a client, using a browser, submits a mathematical question to a broker. The query contains the problem, described in the OpenMath language[5], some information about the client, as described in the example above, and possibly a description of a particular service which the client wishes to use. This last part is optional, and characterises the difference between a relatively straightforward query — the user says use a given service — and the much harder case, when all that the broker has to work on is the problem itself and must use mathematical reasoning techniques to establish strategies for solv-

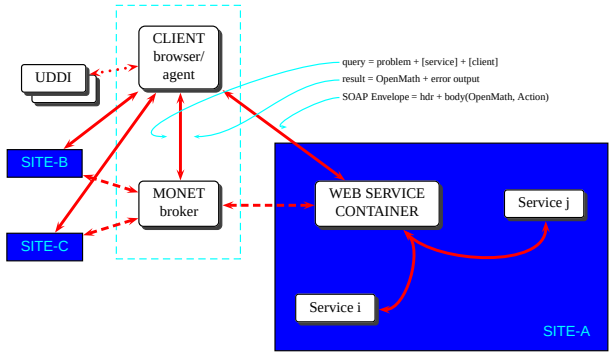

Figure 1. General Monet Architecture

ing the problem. The information within the query is formed from the Mathematical Query Ontology.

We note that there is a separation between the services and a broker that has the task of identifying services and negotiating for their use. The MONET project has developed the concept of the broker[6] significantly. The other component to which we wish to draw attention is the ontology, to which we return later (see section 3.5). In the case of MONET there was a need to develop a Mathematical Service Description Language, in addition to the problem and query ontologies.

2.1. The Mathematical Broker

The broker is expected to operate in a number of roles in order to process mathematical queries and discover services. The ontologies combine to describe both the problems themselves and the services which could be used to solve these problems. A broker intelligently matches services to problems through consultation with the ontologies. Essentially the broker is an intelligent collection of agents, which may communicate with other brokers when choosing a service to solve a particular problem. The client is expected to request their problem to be solved, either by a specified service or software package already known to the client, or by searching for an appropriate capable service.

An additional facility which is expected of the broker is that of a planning service. This is necessary for compound problems which require the facilities of more than one service to solve. The planning service would provide an execution plan to be carried out by either the client, or by an orchestration service on behalf of the broker.

The MONET system has been developed significantly beyond this introduction, but we have presented here sufficient of the architecture to see that it provides a shift from local computation and local facilities to a net-wide GRID-like system of services.

3. A Musical Web Architecture

We now consider how the lessons on MONET can be applied to a putative musical web service.

Musical activities can be thought of as a collection of services; delivery of complete works as PCM or compressed format, synthesis services, musical transformation services (that is filtering) and storage services such as filing or burning CDs.

If we make the systematic changes to the MONET architecture we can see the emergence of a musical web service system. Using a browser as the interface to the web, access is made to a musical broker (which we may name an *impresario*) who has the responsibility for negotiating with a number of musical service providers, presented as web services. An XML-based language is needed in order that these systems may communicate, a language we have dubbed Musical Description Language or MDL; this is consider in more detail in section 3.1. Eventually the results will be returned to the browser.

A natural extension for the musical application is to allow the results to be provided elsewhere, such as a performance system, active speakers, or similar facility. Similarly there will be a need for the results to be sent to a collection of performers, as shown in figure 2. The Banbury process differs a little from the MONET one as there the principal application is to solve a problem, while in the applications we currently envisage the actions are more functional

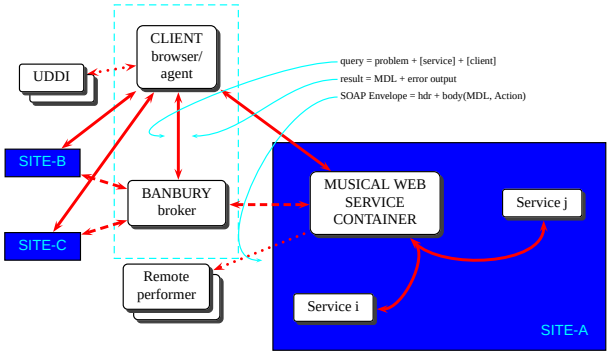

Figure 2. General Architecture of Banbury

— synthesise this, filter that, or perform the other.

We will now consider the various components of this structure, and identify the necessary actions.

3.1. Musical Description Language

In the mathematical arena there has been a significant research effort spent on developing languages to communicate mathematical semantics; OpenMath[5] is perhaps the best known. Much of the equivalent musical work has concentrated on notation (for example see [7] and [8]) but we require a much wider language. Related projects are the MPEG-4 language SAOL[9] which in effect creates a language for synthesis, within a certain limited scope, and Open Sound Control[10] which is a low-level language for controlling performance. Some aspects of SDIF[11] are similar to what we need, dealing with streams of musical data.

The MDL must be capable of representing not just CMN (classical musical notation) and PCM streams, but also analysis data, spectral representations, and low bandwidth control information, as well meta data such as provenance, composer, copyright and licensing information. It is clear that this will need an extensible language to incorporate new analysis methods and those aspect of which we have not yet thought. This has the potential for a sink of effort, so we expect its design to be incremental, expanding as we add applications to the structure.

3.2. Banbury Broker

The broker requirements for the musical architecture are very similar to those of the mathematical broker. The main tasks are to identify web services for synthesis, analysis, filtering, format exchange and so forth, and negotiate their use.

The research consensus is to use ontology modelling languages for the purpose of locating web services, meaning DAML+OIL and the now almost finalised OWL [12]. However, both of these are rather too general and because of the expected growth in web services, extensions to support service descriptions have been created, namely DAML-S [13, 14], OWL-S, respectively, although at the moment, tools are only available for DAML-S, of which RULE-ML [15] reasoners were potentially the most relevant for the MONET work; as yet there is no clear equivalent mechanism for a musical broker, and we need to develop this.

The acquisition process for a service can be looked at as a three step process in which three different kinds of agents have supporting rôles to play. Broadly, these three categories of agents are: service agents, middle agents and service requester agents. The service agents act as front-ends to specialised musical software, the requester agents ask the service providers to carry out their service function, while the middle agents are responsible for locating specific provider agents (matchmakers and brokers).

3.3. Musical Web Service Container

The services identified by the broker are not the direct raw services but are encapsulated in a Musical Web Service Container. We are expecting that many services that the user requests as a single action will in practice be provided by a collection of web services. In order to hide this layer our architecture has container services. An example can be found in the examples section (4.2).

3.4. Interaction frameworks

All musical performance is ultimately grounded in interaction. Even in the case of a solo performance, such interaction may take place between composer and performer, before a performance, as part of the process of the development of a piece. But the more natural reaction is to think of the interaction between performers *during* the performance. Here it is at its simplest in the case of a duet and rapidly gets more complicated as numbers increase until it is accepted that a conductor is required.

It is a novel observation that these interaction frameworks exhibit similar characteristics to so-called institutional structures and thereby to the norms that capture the range of behaviour, rules and procedures that govern interaction between the participants. Although our work in institutions and norms is rooted in the economic concept established by North [16], we believe the principles are much more general. Starting from the ideas put forward by North [17, 18], we have applied them in diverse area such as fish auctions [19] and organ and tissue transplant management [20] as well as making progress on their formalisation for computational and reasoning purposes [21, 22]. In this last, North's concept of norm is stratified into four levels:

abstract norms: which capture high level goals in general terms, frequently un-related to any application domain. The best example of this is that "each agent shall be treated fairly" – obviously the interpretation of this is

quite different in concrete terms if one considers a trading versus an organ transplant scenario.

concrete norms: which derive from abstract norms, but are grounded, typically by means of an ontology, in a particular domain – for which the ontology defines the concepts and their relationships. A musical example could be that in a jazz combo, each agent shall have an improvisatory solo, or that in an orchestra, each agent shall have constrained freedom of interpretation.

regulatory norms: are more specific still, and typically define obligations or prohibitions in terms of a contextual condition and a corresponding action. For instance within a ensemble each player-agent is obliged to follow the conductor, and may not drown-out the melody line,

procedural norms: are the most detailed level and effectively define a protocol, not necessarily as a totally ordered sequence of events, but in which some actions must precede or succeed others. The classical example is that of a conversation, in which – normally – speakers take turns, but at times, one may interrupt the other, perhaps accompanied by some action or signal. A similar protocol can be observed in musical improvisation.

For the sake of the remaining discussion, we should make clear that we will use the term "norm" in the same way as North, that is to refer to any or every constraint on behaviour, regardless of the degree of detail, unless the point depends on the level.

By being able to classify a norm and identify its level, this contributes to further clarification, since it establishes – or not! – the relationship between norms at adjacent levels in the stratification. A vertical set of relationships between levels is the representation of and the means for the validation of a policy.

The purpose of this quite extensive introduction to institutions and norms is to underpin our assertion that economic institutions as defined by North are a specialised application of a more general principle for which we use the cumbersome term interaction frameworks. And furthermore, these interaction frameworks can be used to capture the norms that guide musical performance. As a result, we believe this approach can then be used with multiple performers – both synthetic, in the form of agents playing synthesised instruments, and human – enabling electro-acoustic music to move beyond humans either playing against an unchanging recording or perhaps at best triggering musical events via midi.

3.5. Musical Ontologies

The framework approach that we have just described is a part of the story: it defines the context in which the agents (human or software) interact, but it says nothing about what they communicate to one another. For that we need a language, or perhaps several languages. Again drawing from our background in software agents, we see the FIPA [23] agent communication language performatives as quite adequate for describing the speech acts of one agent to another. The complete list is:

```
accept-proposal, agree, cancel,
cfp, confirm, disconfirm,
failure, inform, inform-if,
inform-ref, not-understood,
propose, query-if, query-ref,
refuse, reject-proposal,
request, request-when,
request-whenever, subscribe
```

But the primitives are `inform` and `request`, with the rest being macros.

However, although the FIPA performatives enable one agent to convey to another the kind of illocution (s)he/it is making, the critical component is the content of the communication. For this we need two things: 1. a content language in which to write expressions – typically this is similar to predicate calculus – such as Jade's [24] SL (Semantic Language). There are many such other languages, but since we are using Jade for our work, we are also using SL. 2. an ontology, whose elements will be the leaves of the expressions, which defines the concepts that constitute a domain of discourse and the relationships between the concepts. Again, there are numerous ontology description languages, but the one that is set to become the most widely used is OWL [12] and we have been developing tools to translate a limited subset of OWL to fit into Jade's (single inheritance based) ontology class structure [25].

One of out primary aims in developing this system is that it should be used for composing, listening to and performing computer music. In such a complex system, there are many concepts which must be modelled for expression in agent↔agent communication and agent↔human communication. With this in mind we intend to develop a collection of layered, time and musical ontologies. So far we have put together a very basic performance ontology covering the standard (Italian) musical terms that frequently decorate scores – as well as implementing the actions that agents must take to apply these instruction to transform scores appropriately. This is an area where there is much to be done,

and we expect development to proceed as applications reveal the detailed needs.

Thus our performance agents can pass relatively simple messages around the system, and the Jade system automatically generates appropriate codec methods for the ontological terms, so it is straightforward for an agent to see what it is being asked to do. Having the ontology does not however remove the problems of the size of musical data, an issue to which we will return in section 4.3

4. Example Projects

In this section we present a range of projects that fit within the web services model for musical delivery, ranging from simple ones to more speculative ideas. Not all of these are complete, but by describing them we seek to convince the reader that the architecture is fundamentally sound.

4.1. NetCsound

NetCsound is a simple web page that allows a user to enter a Csound[2] orchestra and score as an XML document. This can be specified either as a local file or a URL. The computer behind the web page then synthesises the music, optionally compressing it with Ogg – conceptually this could be any compressed format, but there are licencing issues to consider. As synthesis can be a slow process the client has the option to wait for the result, or to elect to have an e-mail sent when it is available.

This can be seen as a very simple Container service. The initial version was in reality a single site, but when MP3 compression is introduced this will be performed on a separate computer which has an MP3 creation licence. The user will not need to be concerned that this component is achieved using a separate site, as the functionality will be provided by the container site.

The initial version of NetCsound was actually driven by a CGI script, but this is now being replaced by a proper web-service to provide the seamless integration of a synthesis and a compression service.

Of course in the larger scheme this service needs to be known to a broker, who may have a number of similar services.

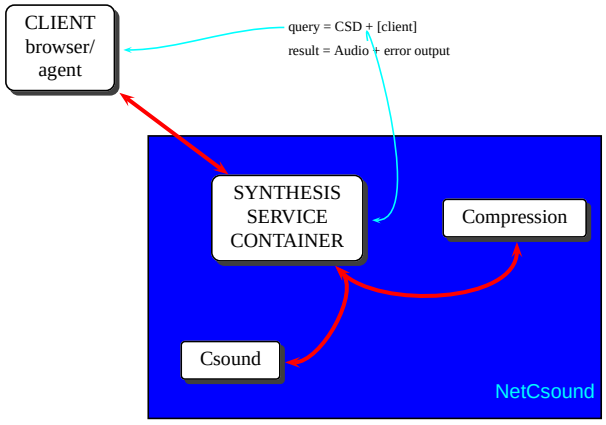

Figure 3. the NetCsound System

4.2. Net-based Audio Processing

The logical development of the NetCsound web service is a collection of audio transformation services that may be composed, the results of one being fed into another. Following the experience of MONET we will be re-using the mechanisms developed in that project, where we used BPEL4WS to generate and handle the workflows between multiple services. The first topic we are working on in this context is signal filtering, since the composition of filters is both useful in itself as well as being an instructive process. Furthermore, as the client experiments with the composition of various simple filters, until the desired effect is obtained, it would then be possible to construct automatically a single filter from the composition and then deploy that as a new service.

For the present we just have a basic filtering mechanism, with a two families of web service providing a second order FIR filter

$$y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2}$$

and a family of second order IIR filters

$$y_n = x_n - b_1 y_{n-1} + b_2 y_{n-2}$$

The client requests that a series of filters are applied to a signal using a Musical Service Description Language, such as given in Figure 4, where the `om` namespace prefix refers to OpenMath [26], an ontology for semantic markup of mathematical information.

This problem statement of the required action may be sent to a broker [6] – a further concept that we are re-using

```
<banbury:action id="my filtering">
  <filter
    source="http://mysound.org/a.wav">
      <filter:cascade>
          <filter:iir>
        <om:OMV name="b1"/>
        <om:OMI>-1.0</om:OMI>
        <om:OMV name="b2"/>
        <om:OMI>-1.0</om:OMI>
      <filter:iir/>
          <filter:fir>
        <om:OMV name="a0"/>
        <om:OMI>1.0</om:OMI>
        <om:OMV name="a2"/>
        <om:OMI>0.5</om:OMI>
        <om:OMV name="a3"/>
        <om:OMI>0.5</om:OMI>
          <filter:/fir>
      <filter/cascade>
  </filter>
<banbury:/action>
```

Figure 4. An example of a filtering problem

from the MONET project – whose task is to identify two filter services, one in each family, and arrange that the output from the first is fed to the second, and the total output returned to the original client.

There are two obvious extensions to this process: one is the use of mathematical services, like those developed and deployed in MONET, to solve algebraic and numerical aspects of DSP, and the other is the orchestration (to use the accepted term for combining web services) of numerous DSP functions published as web services.

4.2.1. MONET and Banbury. MONET has developed generic technology for the description and invocation of mathematical web services. Amongst the demonstrator services that have been set up is one for factoring polynomials. This is directly applicable to the construction of a filter cascade, since by factoring the rational response function of a higher order filter over the reals, into quadratic and linear factors, and then generating the `banbury:action` from the the result of the MONET query, we can identify a set of simple filters that may be composed to achieve – numerically stably – the effect of the high order response function. Integrating this technique with a filter design program such as METEOR[27] is an attractive additional project.

4.2.2. DSP Flow Processing. The example presented above only used filtering, but it is a small step from that to providing a range of simple DSP processes as web services, which can be orchestrated and combined via a complex query and brokerage scheme. Services with zero or one audio input and one audio output are really no different to the filter services. We need to extend to some acyclic directed graph with duplication and merging services and the whole process becomes like a DSP network, as used to illustrate Csound orchestras, or more immediately relevant, to a MIDAS-like distributed synthesis system[28]. The planning of the flow through the signal processing network is controlled by the broker. We do not expect a service based on the worldwide web to provide comparable efficiency, but on a local dedicated network it might be expected to be competitive, and easily upgradable as the hardware improves. It also provides an opportunity for specialised synthesis services, such as a high-quality piano emulation, or a full guitar model, to be incorporated into a non-real time synthesis. While it is beyond the scope of our current work, the brokerage service could negotiate a price for use of a unique sound.

Work has begun on the MIDAS-like system, called **Pactolus** after the river that inherited King Midas' gold creating habit.

4.3. Remote Performance

A problem we have not considered above is the size of audio data. This introduces problems of latency and possible lack of quality of service. In this section we consider a model for remote performance of synthesised electroacoustic music, which has a low data transfer rate between the performer and the performance site.

In order that the data transfer rate is not a limiting factor we need to ensure that only control data is transferred, which requires that the synthesis take place at the performance site. The remote performer can send control data, influencing the actual synthesis.

Based on the MUSIC V model, we take the original score and divide it into regions (not necessarily coinciding with the score sections, but sub-components of sections) for which some gestural modifications are possible. Modifications such as amplitude, tempo or instrumental balance are the most obvious examples, but in any individual piece there may be others. The performer can indicate modifications to the synthesis for the next score region that has not yet started. The additional control data is then sent to the performance site. As long as the control regions are sufficiently long there should be no latency problems. If we use a time synchronisation protocol such as Network Time

Protocol[29] it should be possible to make the remote performance appear at exactly the same time as the performer hears the monitoring system. If a short delay is acceptable then the performer could be playing the piece a few seconds earlier, which would allow for smaller regions and even retrospective control.

The main concept for this remote performance is the physical separation of the performer and the performance, but we can extend the ideas to multiple performers in one space; it does include the possibility of a netbased performance of Xenakis' *Duel* or *Stratégie*, and the opening of opportunities for more heteronomous music[30].

5. Conclusions

We have presented a web-services musical architecture that can provide a secure foundation for a wide variety of musical activities, and especially those that are geographically diverse. In analogy to ubiquitous computing we are providing ubiquitous sounds, and hence music. It can support synthesis from a simple hand-held device, and remote performance processes. The working title of **Banbury** reflects this universality of sound.

In the words of the old English nursery rhyme,

> Ride a Cock-horse to Banbury Cross
> To see a fine lady on a white horse
> With rings on her fingers and bells on her toes
> She shall have music where-ever she goes.

The architecture we have presented here should provide the structure for music wherever we go, whether on laptops, workstations, PDAs or mobile telephones.

References

[1] John ffitch and Julian Padget, "Learning to Play and Perform on Synthetic Instruments," in *Voices of Nature: Proceedings of ICMC 2002*, Mats Nordahl, Ed., School of Music and Music Education, Göteborg University, September 2002, ICMC2002, pp. 432–435, ICMC.

[2] Richard Boulanger, Ed., *The Csound Book: Tutorials in Software Synthesis and Sound Design*, MIT Press, 2000.

[3] NS-Dream, "NetCsound," `http://dream.cs.bath.ac.uk/netcsound`.

[4] Marc-Laurent Aird, Walter Barbera Medina, and Julian Padget, "Monet: service discovery and composition for mathematical problems," in *Proceedings of IEEE workshop on Agent-based Cluster and Grid Computing (at CCGrid 2003)*, Omer Rana and Sven Graupner, Eds. IEEE Computer Society, May 2003, pp. 678–687, IEEE Computer Society Press, ISBN 0-7695-1919-9. Invited paper. Also available from the MONET project website: `http://monet.nag.co.uk`.

[5] The OpenMath Society, "OpenMath website," `http://www.openmath.org`, February 2003.

[6] Walter Barbera Medina, Julian Padget, and Marc-Laurent Aird, "Brokerage for Mathematical Services in MONET," in *Collected papers from Web Services and Agent Based Systems workshop (AAMAS'03)*, Laurence Cavedon, Ed. Kluwer, 2004, in press.

[7] Kai Renz and Holger H. Hoos, "A WEB-based Approach to Music Notation using GUIDO," in *Proceedings, ICMC'98*, M. Simoni, Ed. ICMA and University of Michigan, October 1998.

[8] "Musicxml definition," `http://www.recordare.com/xml.html`, January 2004.

[9] Eric Scheirer, "SAOL Home Page," `http://sound.media.mit.edu/~eds/mpeg4/`.

[10] Matthew Wright and Adrian Freed, "Open SoundControl: A New Protocol for Communicating with Sound Synthesizers," in *Proceedings ICMC 1997*, Perry Cook, Ed. 1997, pp. 101–104, ICMA and Aristotle University of Thessaloniki.

[11] Matthew Wright, Amar Chaudhary, Adrian Freed, Sami Khoury, Ali Momeni, Diemo Schwarz, and David Wessel, "An XML-based SDIF Stream Relationships Language," in *ICMC2000*, Ioannis Zannos, Ed. ICMA, August 2000.

[12] "Web Ontology Language (OWL) Reference Version 1," `http://www.w3.org/TR/owl-ref/`.

[13] DAML, "DAML-based Web Service Ontology," October 2002, Available from `http://www.daml.org/services/daml-s/0.7/`.

[14] Massmo Paolucci, Katia Sycara, and Takahiro Kawamura, "Delivering semantic web services," Tech. Rep. CMU-RI-TR-02-28, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 2002.

[15] RULE-ML, "Rules and Rule Markup Languages for the Semantic Web," 2003, Available from `http://www.dfki.uni-kl.de/ruleml/`.

[16] Douglass C. North, *Institutions, Institutional Change and Economic Performance*, Cambridge University Press, 1991.

[17] P Noriega, *Agent mediated auctions: The Fishmarket Metaphor*, Ph.D. thesis, Universitat Autonoma de Barcelona, 1997.

[18] J-A. Rodríguez, *On the Design and Construction of Agent-mediated Institutions*, Ph.D. thesis, Universitat Autonoma de Barcelona, July 2001.

[19] Joan-Antoní Rodríguez, Pablo Noriega, Carles Sierra, and Julian Padget, "FM96.5 A Java-based Electronic Auction House," in *Proceedings of 2nd Conference on Practical Applications of Intelligent Agents and MultiAgent Technology (PAAM'97)*, London, UK, Apr. 1997, pp. 207–224, ISBN 0-9525554-6-8.

[20] Javier Vázquez-Salceda, Julian Padget, Ulises Cortés, Antonio López-Navidad, and Francisco Caballero, "Formalizing an electronic institution for the distribution of human tissues," *Artificial Intelligence in Medicine*, vol. 27, no. 3, pp. 233–258, 2003, ISSN: 0933-3657.

[21] Marc Esteva, Julian Padget, and Carles Sierra, "Formalizing a language for institutions and norms," in *Intelligent Agents VIII*, Jean-Jules Meyer and Milinde Tambe, Eds. 2001, vol. 2333 of *Lecture Notes in Artificial Intelligence*, pp. 348–366, Springer Verlag, ISBN 3-540-43858-0.

[22] Javier Vázquez-Salceda, *The role of Norms and Electronic Institutionsin Multi-Agent Systems applied to complex domains. The* HARMON*IA framework*, Ph.D. thesis, Universitat Politècnica de Catalunya, 2003.

[23] Foundation for Intelligent and Physical Agents, "Agent communication language," http://drogo.cselt.stet.it/fipa/spec/fipa97.htm, 1997.

[24] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa, "JADE — A FIPA-compliant agent framework," in *Proceedings of the 4th International Conference on the Practical Applications of Agents and Multi-Agent Systems (PAAM-99)*, London, UK, 1999, pp. 97–108, The Practical Application Company Ltd.

[25] Daniel Jiménez Pastor and Julian Padget, "Towards HARMONIA: automatic generation of e-organisations from institution specifications," in *Proceedings of Ontologies in Agent Systems at AAMAS03*, 2003, Published electronically at `http://CEUR-WS.org/Vol-73/oas03-jimenez.pdf`.

[26] The OpenMath Society, "OpenMath website," `http://www.openmath.org`, February 2003.

[27] K. Steiglitz, T.W. Parks, and J.F. Kaiser, "METEOR: A Constraint-Based FIR filter design program," *IEEE Trans. Signal Processing*, vol. 40, no. 8, pp. 1901–1909, August 1992.

[28] T. Anderson, A. Hunt, R. Kirk, P. McGilly, R. Orton, and S. Watkinson, "From Score to Unit Generator — A hierarchical view of MIDAS," in *Proc. International Computer Music Conference*, San José, 1992, pp. 235–238, ICMA.

[29] David L. Mills, "RFC1305: Network Time Protocol (Version 3). Specification, Implementation and Analysis," `http://www.faqs.org/ftp/rfc/rfc1305.pdf`, March 1992.

[30] Iannis Xenakis, *Formalized Music: Thought and Mathematics in Music*, Pendragon Press, 2nd edition, 1992.