Semantic–enabled Integration of Voice and Data Services: Telecommunication Use Case

Tomas Vitvar
Digital Enterprise Research Institute
National University of Ireland
Galway, Ireland
tomas.vitvar@deri.org

Jana Viskova
University of Zilina
Zilina, Slovakia
viskova@kis.fri.utc.sk

Abstract

Convergence of information and telecommunication networks leading to integration of voice, data and video services will bring new opportunities for users as well as service providers. With a growing number of services available on the market, dynamic discovery of services, their composition, selection, mediation as well as execution will be required. In this paper we show how voice and data services can be seamlessly integrated using existing client as well as server VoIP systems based on a widely used SIP signaling protocol and newly emerging technologies in Semantic Web Services. Within particular use case scenario, we describe all phases of the call set-up between two call participants. We show how Semantic Web Services technology can facilitate dynamic and optimal integration of voice and data services with different characteristics while at the same time conforming to users' needs and preferences.

1. Introduction

Convergence of data and telecommunication networks leading to one and transparent technology promises better services, better quality and better deals for anybody, anywhere and anytime. Liberalization of the telecommunications market brings freedom to users who can choose different operators for different services, enables portability of services with respect to users' needs, and enables better price and quality ratios. Convergence of networks will also bring so called *triple-play* services into market – combination of data, voice and video. A simple example is the integration of voice or video services such as "make a call" with data services such as "lookup a phone number" or "compare prices of operators". Convergence of networks combined with the impact of liberalization will open new opportunities for service providers. While they differ in quality, price and offers of added value, hundreds of services available for users will be hard to use without means of automatic service discovery, selection and composition. A user as a subscriber of two or more operators and manually comparing their services and prices to make a call for the best possible value is likely to appear more often in the future. In order to meet users' needs and preferences in this dynamically changing environment, integration of voice and data services should be automated and transparent to users.

Although practical integration of voice and data services still requires a certain amount of human effort, several approaches to automate such integration already exist. These approaches are however based on rigid configuration of systems and hard-wired integration. An example is a click-to-dial application allowing a caller to make a call using a callee name. In this scenario, a callee number is first selected from a pre-configured phone directory and call is established using a pre-defined operator. Operators used for making a call that appear at run-time can also be selected from a set of pre-defined, pre-agreed ones. Therefore, a caller will have a limited number of choices to use the best operator, to make a reliable or a cheapest call. However, if operator's network fails and no alternate operator was defined when designing the application, the phone call will fail. Instead of having a rigid configuration, dynamic and reconfigurable integration is a step beyond traditional approaches. It introduces dynamics by automatically locating the best service providers for a given user request and consequently optimizes the integration process.

In this paper we illustrate how two promising technologies, namely Voice over IP (VoIP) and Semantic Web Services (SWS), can be used to promote voice and data services integration. The paper is structured as follows: In sections 2 and 3 we introduce concepts and technologies of VoIP and Semantic Web Services which we will use throughout the paper. In section 4 we describe a use case scenario and in section 5 we illustrate how voice and data services can be integrated with use of recent technologies in these areas, namely Asterisk as a SIP proxy and WSMX as Semantic Web Services execution environment. Finally, in section 6 we outline the related work and in section 7 we summarize the paper and discuss our future work.

2 Voice over IP

Voice over IP (VoIP) also known as Internet telephony or IP telephony (IPtel) refers in general to the real-time transport of voice or multimedia between two or more parties over IP network [21]. It presents an alternative solution to existing Public Switched Telephone Network (PSTN) voice services enhanced with multimedia transport. VoIP by its nature enables new opportunities to integration of voice and data services native in IP networks. Since IP network was originally designed for non-real-time services (e.g. e-mail, file transfer), supplementary technologies had to be developed to ensure quality of services (e.g. acceptable delays) with respect to real-time communication as well as *signaling* functionality. Signaling is the basis for service creation in telecommunications. It refers to the exchange of control messages in order to create, manage, and terminate communication among participants. In fact, a simple "make a call" service is initiated with an extensive exchange of signaling messages such as ensuring location of the called party, his/her availability as well as compatible media type negotiation. Only after that, transport of voice data for the actual communication can start. Two standard signaling protocols have emerged to fulfill these needs in IP networks: H.323 [6] and Session Initiation Protocol (SIP) [14]. In our use case, we will make use of the latter one.

Session Initiation Protocol (SIP) is an application layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants [14]. Participants can be humans or machines (e.g. a voicemail server). They are identified by SIP URL addresses in a form of *sip:user@domain* or *sip:user@IPaddress*. SIP is a client-server protocol built mainly on the HTTP and SMTP principles. Its architecture includes *end system applications*, such as softphones or SIP phones and *network servers*, such as Registrar, Redirect and Proxy servers. They all facilitate VoIP communication in terms of looking up and keeping updates on the location of users, and redirecting or forwarding signaling requests to next-hop servers or a called party.

2.1 Asterisk

Different VoIP standards have been adopted in the telecommunications domain. Such standards include signaling protocols, different media types or media codecs. Although they differ in quality or usability, they must often co-exist together also with legacy telecommunication systems, such as PSTN. In other words, it should be possible to integrate existing VoIP technology and standards as well as legacy systems in one environment. One of the solutions addressing these requirements is called Asterisk [2]. Asterisk is a software Private Branch Exchange (PBX) being developed under an open source licence. Although it primarily serves functionalities of PBX, it also acts as a SIP proxy server as well as middleware, connecting various telephony technologies including legacy (e.g. PSTN) as well as VoIP interfaces (e.g. SIP, H.323). On top of its fundamental functionality, Asterisk provides a set of APIs for development of new applications allowing programmers to interface with Asterisk at any stage of call setup and teardown.

Although Asterisk is not considered to be a fully-fledged SIP proxy server[1], it provides all SIP functionality required by our use case. In addition, Asterisk with its open architecture and development interface allows us to accomplish the seamless integration with the Semantic Web Services technology. We will make use of these features in our use case.

3 Semantic Web Services (SWS)

Web services have added a new level of functionality to the current Web by making a first step towards the seamless integration of distributed software components using web standards. While current web service technologies around SOAP, WSDL and UDDI cover mainly aspects of syntactic interoperability between services through common standards, the Web Service Modeling Ontology (WSMO) [19] complements these standards by providing a conceptual model and language for the semantic markup describing all relevant aspects of general services which are accessible through a web service interface. The ultimate goal of such markup is to enable the (total or partial) automation of the tasks (e.g. discovery, selection, composition, mediation, execution, monitoring, etc.) involved in both intra- and inter-enterprise integration of web services.

WSMO has its conceptual basis in the Web Service Modeling Framework (WSMF) [15], which adheres to the principles of loose coupling of services and strong mediation among them. WSMO defines an underlying model for the WSMX Semantic Web Services execution environment [17] as well as draws up requirements for a WSML ontology language [12] used for formal description of WSMO elements. Thus, WSMO, WSML and WSMX form a complete framework to deal with all aspects of Semantic Web Services.

WSMO top-level conceptual model is composed of *Ontologies*, *Goals*, *Web Services* and *Mediators*.

Ontologies provide formal explicit specification of shared conceptualization that is formal semantics of infor-

[1] Alternatively, other SIP proxy servers could be used for our use case, such as SIP Express Router (SER) [4].

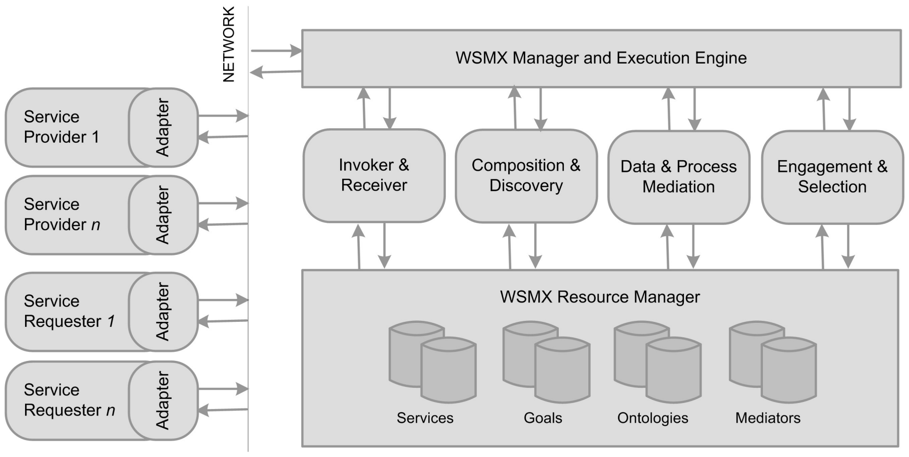

Figure 1. WSMX Architecture

mation used by other components (goals, web services, and mediators). WSMO specifies the following constituents as part of the description of an ontology: *concepts*, *relations*, *functions*, *axioms*, and *instances* of concepts and relations, as well as *non-functional properties*, *imported ontologies*, and *used mediators*. The latter allows the interconnection of different ontologies by using mediators that solve terminology mismatches.

Goals provide description of objectives of a service requester (user) that he or she wants to achieve. WSMO goals are described in terms of desired information as well as "state of the world" which must result from execution of a given service. In WSMO, a goal is characterized by a set of *non-functional properties*, *imported ontologies*, *used mediators*, a *requested capability* and a *requested interface* (these definitions are the same as for web services).

Web Services provide a functionality for a certain purpose, which must be semantically described. Such description includes *non-functional properties*, *imported ontologies*, *used mediators*, *capability* and *interfaces*. *Capability* of a web service is modeled by *preconditions* and *assumptions* for the correct execution of the web service as well as *postconditions* and *effects* resulting from this execution. The interface for every web service is modeled as *choreography* describing communication pattern (interactions) with this web service and *orchestration* describing partial functionality required from other web services.

Mediators describe elements that aim to overcome structural, semantic or conceptual mismatches that appear between the different components that build up a WSMO description. WSMO specification currently covers four types of mediators: (1) *OOMediators* import the target ontology into the source ontology by resolving all the representation mismatches between the source and the target, (2) *GGMediators* connect goals that are in a relation of refinement and resolve mismatches between those, (3) *WGMediators* link Web services to goals and resolve mismatches, and (4) *WWMediators* connect several Web services for collaboration.

3.1 SWS Execution Environment (WSMX)

Based on WSMO concepts, the Web Services Execution Environment (WSMX) is the execution environment for discovery, composition, engagement, selection, mediation and invocation of Semantic Web Services. The global WSMX architecture and its components are depicted in figure 1.

The WSMX Manager and the Execution Engine facilitate a Semantic Web Services execution process (execution semantics) by triggering discovery, composition, engagement, selection, mediation and invocation components upon receiving users' request (goal) and within the whole interaction process between service requester and service providers. The WSMX execution semantics is the core of the WSMX intelligence providing value-added services to traditional communication. Different execution semantics can be used according to the domain-specific requirements, e.g. mediation components are only required for heterogeneous environments, a selection component is used when (semi) automated decisions to select the best services are required based on requesters' preferences, etc.

The Resource Manager is responsible for the management of repositories to store definitions of web services, goals, ontologies and mediators.

Discovery and Composition of web services is one of the key processes of the SWS technology. A number of services could be returned from this step, services which satisfy the goal composed to a predefined process (*once-for-all*

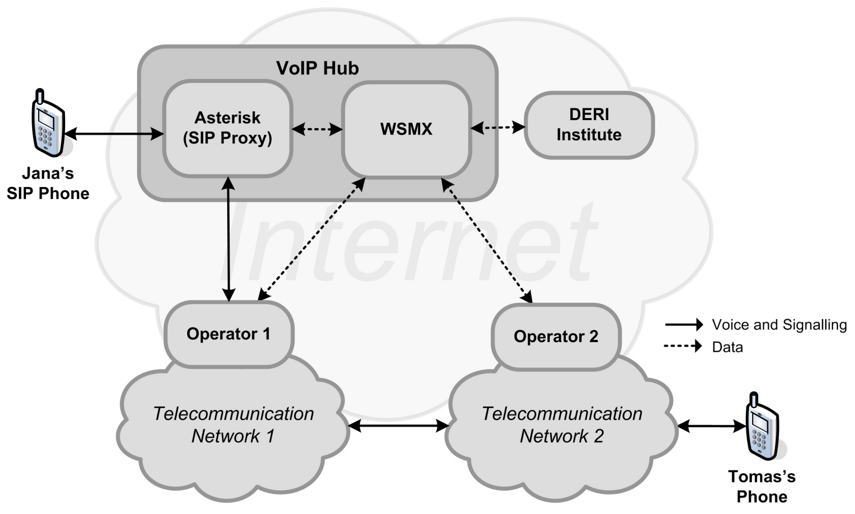

Figure 2. Voice and Data Integration Use Case

composition) as well as dynamically created process (*on-the-fly* composition). Such composition of services could also include "duplicate" services with the same capabilities however with different characteristics (non-functional properties). For duplicate ones, selection of services will be performed.

Engagement is composed of two phases, namely *contracting* and *negotiation*. Usually, discovery and composition operate on more general (abstract) goals. They result with a set of web services that can potentially fulfill a requester goal. However, for a complete guarantee that discovered web services will be able to provide requested concrete service, communication between a requester and a provider is necessary. This phase is called contracting. In addition, negotiation with each perspective web service to reach agreement on terms of services can also be performed.

Selection of the best or optimal service is performed when a number of duplicate services is returned from the discovery and composition process. To find an optimal service, different techniques can be applied, ranging from simple selection criteria ("always the first") to more sophisticated techniques, such as multi-criteria selection of variants also involving interactions with a service requester. Different variants of services could be described by different values of parameters (non-functional properties specific to web services), such as financial properties, reliability, security, etc.

Data and Process Mediation facilitate interactions between two entities when different ontologies or different choreographies are used by these entities. Data Mediation is ensured by mappings between concepts from one ontology to another. It is based on paradigms of ontology engineering, i.e. ontology mapping/aligning [11]. Process mediation provides the necessary functionality for a runtime analysis of two given choreography instances and compensate possible mismatches that may appear, for instance, grouping several messages into a single one, changing their order or even removing some of the messages.

Invoker and Receiver implement an entry point of the WSMX responsible for receiving of incoming requests and invoking web services respectively. Invoker and receiver also handle grounding of WSMO services to underlying WSDL and SOAP protocol.

Adapters facilitate the interoperability between a requester and a provider at the technical/syntactic level. WSMX is designed to internally handle WSML messages encapsulated in WSDL and sent or received using SOAP protocol. Therefore, adapters must ensure that interactions between a service requester and provider can be performed using different communication protocols (e.g. FTP) as well as in different languages.

4 Use Case Description

In our use case depicted in figure 2, Jana (using her standard SIP phone) intends to make a cheapest call with Tomas of who she only knows he works with DERI institute. She neither knows Tomas's phone number, nor available telecommunication operators, their services and prices. Instead, she is registered and connected with the *VoIP Hub* provided by a 3rd party operator (Jana is a VoIP subscriber). VoIP Hub is an intermediary between Jana as a service requester, all service providers as well as Tomas's phone. VoIP Hub comprises of Asterisk and WSMX systems. Among all web services registered with WSMX, two types of web services exist. The first is a telecommunication operator web service *authorize-call* registered by an operator 1 and 2, and the second is a phone directory web service *resolve-name* registered by DERI institute. We further describe these services later in this section.

In order to accomplish Jana's desire, following steps are performed. Given Jana's request ("make a call with Tomas who works with DERI") including her preferences ("cheapest"), web services fulfilling this request are discovered, composed, selected, and invoked, i.e. web service to resolve Tomas's number, and authorization web service of an operator through which the call will be made. As a result, Jana's call to Tomas is authorized and established for the best possible price. In this paper we further demonstrate this process in detail.

4.1 Authorize-call Web Service

The purpose of the *authorize-call* web service is to "open" an operator's gateway to which Jana's call will be redirected from the VoIP hub. We presume, operators provide an access to their network over the Internet using SIP-compliant gateway however only authorized calls are allowed. For example, if Jana is not registered with the operator 1 (in other words, Jana is not a subscriber of the operator 1) she can't make direct calls to/over the operator 1 network but only through another operator where Jana is registered, i.e. the VoIP Hub provider. At the same time, agreement and "peering" between the VoIP Hub and the operator 1 must exist so that calls can be made between both as well as billing. In our use case, such "peering" is built on the Internet and SIP signaling. Furthermore, prices for calling from one network to another can differ significantly. In general, a customer calling through the operator 1 network to the operator 2 network can get better deals comparing to a price a customer of operator 2 pays when calling to the same network. A good example is the Skype [7]. Although it is based on proprietary solutions, Skype offers better deals for many networks than many operators do for their own local calls. Apart from a number of operators registered with WSMX, operator 1 as well as operator 2 have authorize-call web service registered with WSMX.

4.2 Resolve-name Web Service

The purpose of the *resolve-name* web service is to resolve people's or company's names to phone numbers. Typically, such service could be provided by global and local phone directories, such as Yellow Pages or telecommunication operators to look-up numbers of their subscribers. Apart from a number of such web services registered with WSMX, resolve-name web service is registered by DERI institute to look-up numbers of its staff members.

5 Voice and Data Integration Process

In this section we describe in detail the integration process of voice and data services according to the use case scenario described in section 4. According to the sequence diagram depicted in figure 3, this process can be divided into the following phases: (1) Dialing, (2) Transforming Desire to Goal, (3) Achieving Goal, (4) Achieving Desire.

5.1 Dialing

Jana is registered and connected over the Internet to the Asterisk of the VoIP Hub running at the address *voip-hub.ie* using her favorite SIP phone and her credentials (Jana is the VoIP Hub subscriber). This connection is defined in the SIP phone configuration file. Using her phone, she now wants to express and send her desire "make a cheapest call with Tomas who works with DERI". To do that, the desire must be first formalized and second it must be sent to the Asterisk using SIP protocol. There are several ways how this can be done.

One option to formalize the desire is to use some ad-hoc grammar, for example (grammar is written as the regular expression):

```
[a-zA-Z0-9]+#[a-zA-Z0-9]+#[price|quality]{0,1}
```

where terms are divided by # character. Using this grammar, the desire would be described as a string `tomas#deri#price`. The grammar and the meaning of each term must be known to users and at the same time rules/program processing this string must implement them (see the next phase). The meaning of each term is: (1) a callee name, (2) a company name that the callee works with and (3) an optional user preference (*price* means that price for the call is preferred to the quality of the call and vice versa). Although this approach doesn't correspond with the trends of ontology engineering, it is relatively easy to use with existing SIP phones (assuming, SIP phone allows alphanumeric characters to be entered when dialing). Formalized desire is then used as a user part of the SIP address. For example, if Jana "dials" `tomas#deri#price`, the SIP phone will generate following SIP message (please note that only the first line of the message is shown):

```
INVITE sip:tomas#deri#price@voip-hub.ie SIP/2.0
```

Asterisk sends back "100 Trying" response indicating that the INVITE has been received and is now being processed. Various extensions to SIP have already been proposed to standardize similar mechanisms such as Control of Service Context using SIP Request URI [9] or SIP Caller Preferences and Callee Capabilities [20]. However, they were not aimed for the communication of a desire.

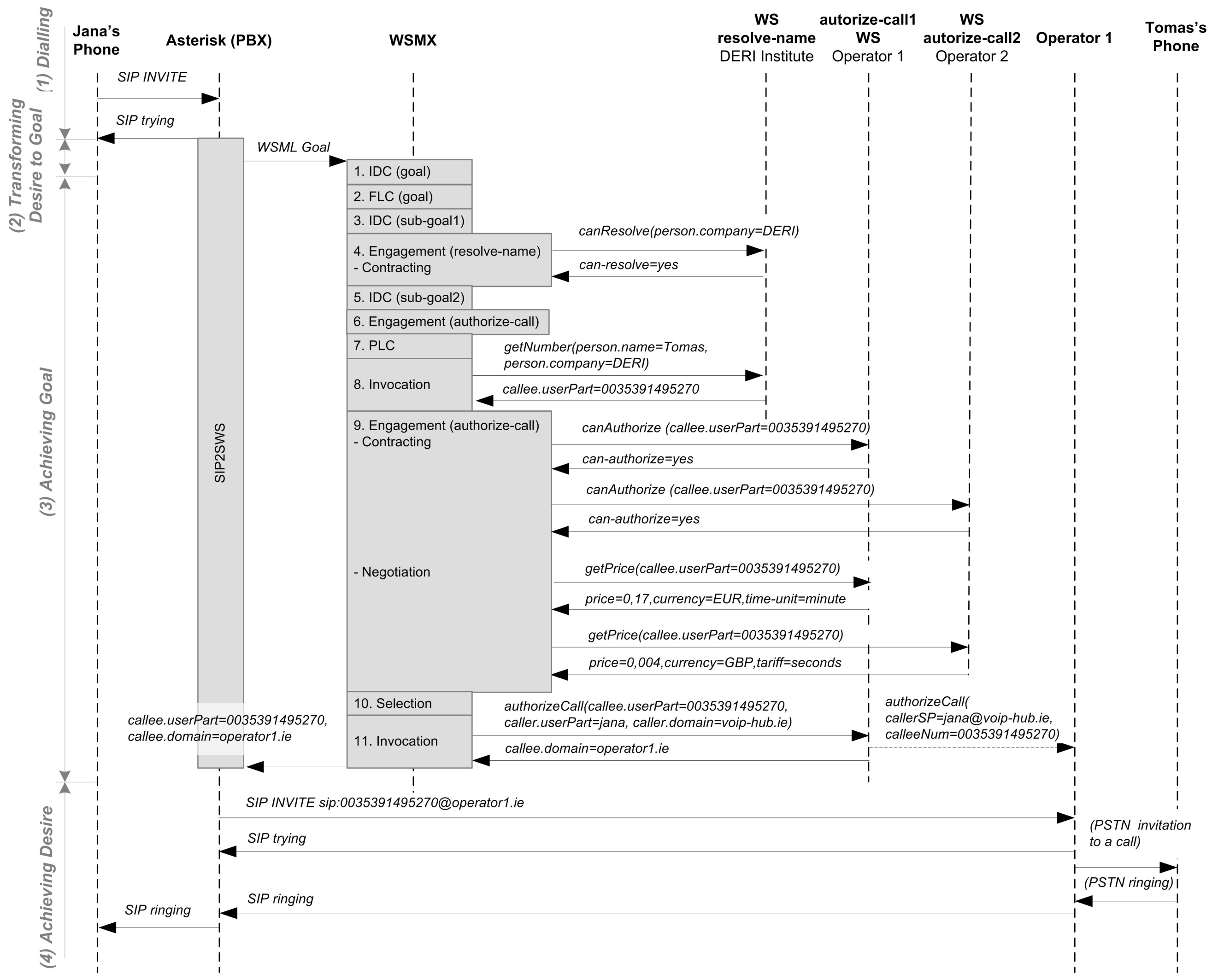
(1) Dialling
(2) Transforming Desire to Goal
(3) Achieving Goal
(4) Achieving Desire
Jana's Phone
Asterisk (PBX)
WSMX
WS resolve-name DERI Institute
autorize-call1 WS Operator 1
WS autorize-call2 Operator 2
Operator 1
Tomas's Phone
SIP INVITE
SIP trying
WSML Goal
SIP2SWS
1. IDC (goal)
2. FLC (goal)
3. IDC (sub-goal1)
4. Engagement (resolve-name) - Contracting
canResolve(person.company=DERI)
can-resolve=yes
5. IDC (sub-goal2)
6. Engagement (authorize-call)
7. PLC
8. Invocation
getNumber(person.name=Tomas, person.company=DERI)
callee.userPart=0035391495270
9. Engagement (authorize-call) - Contracting
canAuthorize (callee.userPart=0035391495270)
can-authorize=yes
canAuthorize (callee.userPart=0035391495270)
can-authorize=yes
- Negotiation
getPrice(callee.userPart=0035391495270)
price=0,17,currency=EUR,time-unit=minute
getPrice(callee.userPart=0035391495270)
price=0,004,currency=GBP,tariff=seconds
10. Selection
11. Invocation
callee.userPart=0035391495270, callee.domain=operator1.ie
authorizeCall(callee.userPart=0035391495270, caller.userPart=jana, caller.domain=voip-hub.ie)
callee.domain=operator1.ie
authorizeCall(callerSP=jana@voip-hub.ie, calleeNum=0035391495270)
SIP INVITE sip:0035391495270@operator1.ie
SIP trying
(PSTN invitation to a call)
(PSTN ringing)
SIP ringing
SIP ringing

Figure 3. Voice and Data Integration Sequence Diagram

Another option to formalize the desire would be to use some standard or ad-hoc ontology with a standard ontology language. Preferably, this could be done using the WSMO goal concept and the WSML language. However, to send such ontology through existing SIP phones would be more difficult without changing its functionality or user interface. Unlike in the first option, this ontology would not be suitable to send as the user part of the SIP address (due to its size or possible character sets used) but rather in the SIP message body in a MIME-compliant format.

We aim the integration of voice and data services to have minimal impact on end-user devices as well as to be transparent to users as much as possible. Thus, we will use the first option described however we admit that this approach is limited and should be more improved in practical scenarios.

5.2 Transforming Desire to Goal

After the SIP message and the desire has been sent, it is received by the Asterisk and processed within its *dial plan* [2]. In the dial plan, an application called SIP2SWS is executed.

The SIP2SWS application is the key application in the SIP and SWS integration process building a bridge between the Asterisk and the WSMX. On one side, it interfaces our SIP session using Application Gateway Interface (AGI)[2] [2] and on the other, it transforms the desire to the WSML goal, triggers SWS process by sending the WSML goal to the WSMX entry point and receives an output from it.

[2]AGI allows Asterisk to launch an external program written in any language to control a telephony channel, play audio, read DTMF digits, etc.

Following is the dial plan entry corresponding to our desire and its formal representation.

```
exten => _.#.#., AGI(sip2sws, ${EXTEN})
```

The element `_.#.#.` defines a dial plan pattern for the user part of the SIP address (here, the pattern corresponds to the regular expression defined in the previous phase), and the element `AGI(sip2sws, ${EXTEN})` defines AGI command to execute SIP2SWS application on the channel. The argument `${EXTEN}` is a placeholder for current extension (user part, in our case formalized desire) sent by the client.

SIP2SWS application is an external application written in Java which transforms Jana's desire "make a cheapest call with Tomas who works with DERI" formalized as `tomas#deri#price` to the goal "authorize a cheapest call between a caller Jana and a person Tomas who works with DERI". We formally describe this goal as well as its ontology in the section 5.2.1. The reason why Jana's desire differs from the goal is such that Asterisk redirects already created channel by Jana to a SIP proxy to further establish the call, whereas WSMX finds the cheapest operator for the call, authorize this call with this operator and return necessary information for redirecting the call (Tomas's number and operator's SIP proxy domain name). If the goal sent to the WSMX was "make a call" instead of "authorize a call", there would be a problem with our approach to put the SIP context of existing channel through the WSMX to another SIP proxy (in this case, Parlay X Web Services interface [10] for telecommunications systems could be used however we haven't investigated such approach in our research). In our use case, we rather combine existing SIP proxy functionality of Asterisk used for the call redirection with the WSMX functionality used for the optimal provisioning of web services. In other words, Asterisk and WSMX share tasks within the VoIP Hub to achieve Jana's desire.

In our use case, we use a simple request-response scenario operating in the asynchronous mode as depicted in the figure 4. First, SIP2SWS application calls the following WSMX system entry point defined as part of the WSMX Integration API [24].

```
Context achieveGoal(WSMLDocument wsmlDocument)
```

The attribute `wsmlDocument` contains data sent to the WSMX, i.e. the WSML goal and the WSML ontology. SIP2SWS application receives a `Context` on return to identify already established session for subsequent calls with WSMX. Result data are then sent back asynchronously from the WSMX by calling a SIP2SWS *endpoint* (we assume this endpoint is registered with WSMX). Such scenario corresponds with the fact that no interactions between Jana and the SWS execution process are required. In more elaborated scenarios, Jana would want to provide additional

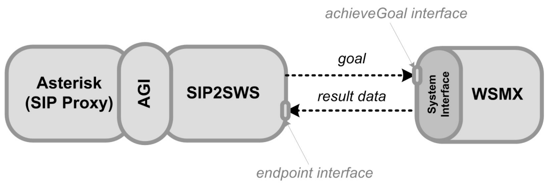

Figure 4. SIP2SWS and WSMX Interaction

information for the selection phase of the SWS execution process by approving for example terms and conditions of discovered services. Such scenarios will be investigated in our future work.

5.2.1 Goal and Ontology Representation in WSML

Simple VoIP ontology in WSML which describes concepts used in our scenario is shown in the listing 1. First, namespaces to distinguish elements of multiple resources are defined, such as for XMLSchema (*xsd*), and Dublin Core (*dc*).

```
namespace {
  xsd _"http :// www.w3.org/2001/XMLSchema#",
  dc _"http :// purl.org/dc/elements/1.1#"}

ontology _"http :// www.example.org/DERI/VoIPont"

  nonFunctionalProperties
     dc#title  hasValue "VoIP Ontology"
    dc#description hasValue "Ontology of VoIP concepts"
  endNonFunctionalProperties

  concept CallParticipant
    nonFunctionalProperties
      dc#description hasValue "Concept of a participant in  a  call "
    endNonFunctionalProperties
    userPart ofType xsd#string
    domain ofType xsd#string

  concept Person
    nonFunctionalProperties
      dc#description hasValue "Concept of a person"
    endNonFunctionalProperties
    name ofType xsd#string
    company ofType xsd#string

  relation  CallAuthorized(ofType CallParticipant,  ofType
          CallParticipant)
    nonFunctionalProperties
        dc#description hasValue "Relation that holds between two
              call participants  when call  is  authorized."
    endNonFunctionalProperties
```

Listing 1. VoIP Ontology

The VoIP ontology contains two concepts and one relation. In particular, the *CallParticipant* concept defines a *userPart* and a *domain* attributes forming a SIP address of a call participant, the *Person* concept defines a *name* of a person and a *company* where the person works and the *CallAuthorized* defines the relation which holds when a call between two call participants is authorized.

Concepts defined in the VoIP ontology are used in the goal which definition is shown in the listing 2. This goal which also includes input data values (*person.name=“Tomas”*, *person.company=“DERI”*, *caller.userPart=“Jana”*, *caller.domain=“voip-hub.ie”*) as well as preference (*preference=“price”*) is generated from the previously described desire and sent to the WSMX by calling the WSMX system entry point.

```
namespace {
  voip _"http :// www.example.org/DERI/VoIPont",
  xsd _"http :// www.w3.org/2001/XMLSchema#",
  dc _"http :// purl.org/dc/elements/1.1#"}

goal _"http :// www.example.org/DERI/SIP2SWSgoal"

  nonFunctionalProperties
     dc#title  hasValue "Authorize Call"
    dc#description hasValue "Authorize a call between a caller and a
           person from a company"
    voip#preference hasValue "price"
  endNonFunctionalProperties

  capability

    sharedVariables { ?caller, ?callee }

    precondition
      definedBy
        exists { ?person, ?caller } (
          ?person[
            name hasValue "Tomas",
            company hasValue "DERI"
          ] memberOf voip#Person and
          ?caller[
            userPart hasValue "Jana",
            domain hasValue "voip−hub.ie"
          ] memberOf voip#CallParticipant
        ).

    postcondition
      definedBy
        exists { ?callee, ?calleeUserPart, ?calleeDomain } (
          ?callee[
            userPart hasValue ?calleeUserPart,
            domain hasValue ?calleeDomain
          ] memberOf voip#CallParticipant
        ).

    effect
      definedBy
        exists { ?caller, ?callee } (
          CallAuthorized(?caller, ?callee)
        ).
```

Listing 2. Goal Authorize Call

In our use case, we use a *preference* keyword as a non-functional property for description of Jana’s preference (we assume this property is defined within the *voip* namespace). Although WSMO defines non-functional properties such as *financial* or *quality of service* [19], we don’t use them as there were created for description of web services rather then for users’ preferences. Proper use of non-functional properties for description of user’s preferences as part of the WSML goal specification will be the subject of the future research within the WSMO and the WSML Working Groups. Apart from the definition of namespaces and other non-functional properties which are similar to those in the VoIP ontology, the goal also contains a *capability* which is requested to result from the semantic web services execution process (see the next section 5.3).

The *capability* description is composed of several blocks, i.e. *sharedVariables*, *precondition*, *postcondition*, and *effect*. The *sharedVariables* block is used to indicate the variables which are shared across the goal definition in all other blocks. In the *precondition* block, requested inputs are specified, i.e. a *person* and a *caller* having assigned the input values. In the *postcondition* block, requested outputs which must result from the processing of the goal are defined. In our example, we request information regarding a *callee* – his/her *userPart* and a *domain* to which a call should be redirected. Similarly, the *effect* block defines requested output corresponding to the state-of-the world, in our case authorized call between a caller and a callee.

The WSML goal as described in the listing 2 is sent to the WSMX according to the scenario depicted in the figure 4. In this case, there is no need for a goal interface to be specified as all data values are sent as part of the goal definition and at the same time the output is sent back by calling the SIP2SWS endpoint. Such scenario corresponds to the simple request-response interaction and reflects the latest WSMX implementation. In more elaborated scenarios requiring more interactions between a service requester and service providers during the SWS execution process, this process will be initiated by sending an abstract goal[3] including requested choreography. Requested choreography defines all requested interactions between service requester and service providers including definitions of input data and their “feeding” to the SWS execution process as well as definitions of requested output data at different stages of communication. Such scenarios will be investigated in our future work.

5.3 Achieving Goal

In this section we describe the WSMX behavior according to the goal received and show the sequence of interactions between the WSMX components. The goal as shown in the listing 2 captures the information that requester wants to receive (requested outputs) as well as effects of the state of the world that requester wants to achieve (requested effects). WSMX Repository contains a number of registered web services; among others also *resolve-name* (by DERI), *authorize-call1* (by operator1), and *authorize-call2* (by operator2). We simplify the semantic description of the goal and the web services as follows.

1. **Goal:** Lookup a *callee.userPart* for a *person.name* that works in a *person.company* and allow call from the *caller.userPart@caller.domain* to the *callee.userPart* through the cheapest operator at a *callee.domain*.

[3]We refer to the abstract goal as a goal containing no instance data (input values)

- **Requested outputs:** *callee.userPart*, *callee.domain*
- **Requested effect:** *CallAuthorized(caller, callee)*
- **Provided inputs values:** *person.name="Tomas"*, *person.company="DERI"*, *caller.userPart="Jana"*, *caller.domain="voip-hub.ie"*
- **Preference:** *price*

2. **Web service:** *resolve-name*
   - **Requested inputs:** *person.name*, *person.company*
   - **Provided outputs:** *callee.userPart*
3. **Web service:** *authorize-call1*
   - **Requested inputs:** *callee.userPart*, *caller.userPart*, *caller.domain*
   - **Provided outputs:** *callee.domain*
   - **Provided effect:** *CallAuthorized(caller, callee)*
4. **Web service:** *authorize-call2*
   - **Requested inputs:** *callee.userPart*, *caller.userPart*, *caller.domain*
   - **Provided outputs:** *callee.domain*
   - **Provided effect:** *CallAuthorized(caller, callee)*

5.3.1 Execution Process

Following is the description of actions of particular components within WSMX to achieve the goal, namely Discovery and Composition, Engagement, Selection and Invocation. For discovery and composition we use integrated approach proposed in [18]. Discovery of web services is called within the composition process every time when a web service need to be discovered. In this approach, following components are identified:

- Integrated Discovery and Composition (IDC).
- Functional Level Composition (FLC) is used when there is no single service that can satisfy the goal. FLC refines the goal into sub-goals in order to find a subset of the existing web services which can be combined into composite service fulfilling the goal.
- Process Level Composition (PLC) creates a workflow for composite service with respect to choreography and orchestration constraints.

1. **IDC(goal)**
   - **Action:** Discovery component is called to match the goal with web service descriptions in the WSMX repository.
   - **Result:** Discovery was not able to find a single WS covering all requirements of the goal.
2. **FLC(goal)**
   - **Action:** The goal is refined into two sub-goals. This refinement is based on predefined generic goals stored in the goals repository.
   - **Result:** sub-goal1: Lookup a *callee.userPart* for a *person.name* that works in a *person.company*
     - **requested outputs:** *callee.userPart*,
     - **provided inputs:** *person.name*, *person.company*, *caller.userPart*, *caller.domain*
     - **provided inputs values:** *person.name="Tomas"*, *person.company="DERI"*, *caller.userPart="'Jana"*, *caller.domain="voip-hub.ie"*
     - **preference:** *price*
   - **Result:** sub-goal2: Allow call from *caller.userPart@caller.domain* to *callee.userPart* through the cheapest *callee.domain*.
     - **requested outputs:** *callee.domain*,
     - **requested effects:** *CallAuthorized(caller, callee)*,
     - **provided inputs:** *person.name*, *person.company*, *caller.userPart*, *caller.domain*
     - **provided inputs values:** *person.name="Tomas"*, *person.company="DERI"*, *caller.userPart="Jana"*, *caller.domain="voip-hub.ie"*
     - **preference:** *price*
3. **IDC(sub-goal1)**
   - **Action:** Discovery component is called to match the sub-goal1 with service descriptions in the WSMX repository.
   - **Result:** Discovery has found *resolve-name* web service with requested capabilities.

- **Action:** So far, we have only considered the outputs and effects in the service descriptions independently of input information that has to be provided to the web service. These inputs determine whether the web service can be actually used. Therefore *resolve-name* web service will be checked for available inputs. Up to now the actions were completely internal to WSMX and they were based on abstract descriptions of goal/sub-goal and web services. There was no interaction with the requester or service providers. For a complete guarantee that discovered web services can provide concrete requested service, further communication with this web service is necessary. Such communication is called engagement and must be called at this stage of discovery and composition.

4. **Engagement(resolve-name)**
   - **Action:** Contracting for *resolve-name* web service; we need to find out whether this web service is able to provide name resolution for DERI staff, thus *canResolve(person.company="DERI")* is called.
   - **Result:** Web service *resolve-name* can provide name resolution for DERI. The second phase of engagement is negotiation however there is no need for it at this stage. Before proceeding to IDC of the next sub-goal2, available inputs for sub-goal2 are extended by outputs of already discovered and contracted web service *resolve-name*, in this case *callee.userPart*.
5. **IDC(sub-goal2)**
   - **Action:** Discovery component is called to match the sub-goal2 with web service descriptions in the WSMX repository.
   - **Result:** Discovery has found 2 web services both satisfying the sub-goal2: *authorize-call1* and *authorize-call2*.
   - **Action:** Each discovered web service will be checked for available inputs.
   - **Result:** Both discovered web services need as input *callee.userPart* which is not provided by the requester. However, thanks to the previous discovered *resolve-name* service which returns *callee.userPart* as its output, *authorize-call1* and *authorize-call2* can go for further processing with restriction that they have to be invoked after the *resolve-name* service.
6. **Engagement**
   - **Action:** Contracting for *authorize-call1* and *authorize-call2* to find out whether each of these web services is able to provide the authorisation for a concrete *callee.userPart* value.
   - **Result:** Contracting is not possible at this stage as all required input values are not known (*callee.userPart* value in particular). In order to contract these web services, the *resolve-name* web service has to be first invoked.
7. **PLC** creates the workflow based on discovered services, *resolve-name* and *authorize-call*.
8. **Invocation**
   - **Action:** *resolve-name* is invoked, *getNumber(person.name="Tomas", person.company="DERI")*
   - **Result:** The web service returns *callee.userPart="0035391495270"*. Available inputs for sub-goal2 are extended by this output.
9. **Engagement**
   - **Action:** Contracting for *authorize-call1* and *authorize-call2* to find out whether each of these web services is able to provide the authorisation for a concrete *callee.userPart* value: *canAuthorize(callee.userPart="0035391495270")*.
   - **Result:** Both services can provide the authorization for the requested *callee.userPart*, both services have available inputs and have been contracted. As for the sub-goal2 we have discovered two possibilities, therefore selection will be performed based on requester's preference (price). However, in order to do that, simple negotiation must be performed beforehand to find out the price for calls offered by both web services.
   - **Action:** Negotiation with *authorize-call1* and *authorize-call2* about the price for a call to *callee.userPart*: *getPrice(callee.userPart="0035391495270")*.
   - **Result:**
     - **authorize-call1 returns:** *price="0.17"*, *currency="EUR"*, *time-unit="minute"*.
     - **authorize-call2 returns:** *price="0.004"*, *currency="GBP"*, *tariff="second"*.

10. **Selection**

   - **Action:** Selection will chose between services *authorize-call1* and *authorize-call2* based on the requester's preference. Since different ontologies are used by these services (e.g. different currencies, different concepts *time-unit* and *tariff*), data mediation will be called at this stage (please note, that mapping rules with conversions between both ontologies must be available before hand to process this data mediation– such information is obtained during the design-time stage of the mediation process).
   - **Result:** *authorize-call1* web service was selected.

11. **Invocation**

   - **Action:** *authorize-call1* web service is invoked: *authorize-call1 (callee.userPart="0035391495270", caller.userPart="Jana", caller.domain="voip-hub.ie").*
   - **Result:** *callee.domain="operator1.ie"* as well as call authorization effect through operator's 1 gateway.

5.4 Achieving Desire

After the goal has been accomplished by the WSMX, SIP2SWS application receives information about Tomas's number as well as the operator's SIP proxy (domain name of the SIP proxy) at which the call between Jana and Tomas has been authorized. Thus, SIP2SWS application can now process further to achieve Jana's original desire. To do that, following SIP message is forwarded from Asterisk to the operator's gateway (only the first line of the message is shown).

```
INVITE sip:0035391495270@operator1.ie SIP/2.0
```

Asterisk receives back 100 Trying response from the operator's gateway indicating that the INVITE has been received and is now being processed. We consider the operator 1 is a VoIP operator having connection with the operator 2 through some SIP/PSTN gateway. As Tomas with number `0035391495270` as a subscriber of operator 2 (hence the operator 2 is a land line operator), the SIP signaling is transformed at the SIP/PSTN gateway into the PSTN signaling in order to reach Tomas. Next, Tomas's phone starts ringing and Jana is notified by the ringing tone.

As we already mentioned earlier, thanks to the liberalization and regulation of transit and termination charges for calls, the alternative operators can compete with legacy operators by offered services and prices. This was also the case in our example.

6 Related Work

Integration of voice and data services is a long term effort addressed within a number of research projects. New opportunities in this area have emerged by introducing web services interfaces for telecommunications systems, namely Parlay X [10]. Parlay X defines a set of high-level interfaces that allow external parties to access the key network capabilities such as short and multimedia messaging or user location and status. These capabilities can be then easily "added" to companies' systems. Good examples are prototypes of Telecom Italia Laboratories (TiLab). Based on Parlay X service architecture, they developed a service called "ClickToSms" which allows to send an e-mail as SMS within Microsoft Outlook with Parlay X interface invoked through Microsoft .Net web service technology. There are of course many other static solutions based on other IT technologies. They usually make use of the architecture of VoIP proxies enhanced with possibility to define service logic that further directs the behavior of proxies. As for examples, Asterisk or StarSIP Tilab proxy and application server [5] based on the servlet technology can be used for any static integration using proprietary scripting languages. They all however lack the features of the Semantic Web Services approach.

Integration of the SWS technology and telecommunications systems is currently the subject of intensive research through the European funded projects DIP [3] and ASG [1]. However, such integration is currently focused only on the telecommunications management. As an example, British Telecom (BT) has explored the use of SWS within its Operational Support System (OSS) [13]. OSS ensures the visibility of the service delivery and assurance for end customers. As multiple other network operators as well as service and content providers can interact with BT during delivery of a service, BT has to allow them to easily integrate with its systems. This approach deals however more with the dynamic B2B integration of data services rather than voice and data services integration. Although it is a clear sign that telecommunication companies are showing the interest in SWS technologies, voice and data integration with use of SWS technologies has not been explored so far.

Apart from the WSMX as the reference implementation for WSMO, other SWS implementations also exist such as IRS [16, 8], OWL-S [8] or METEORS [22, 8]. Some of these implementations are based on their own concepts or on the concepts defined by WSMO. It is the intention of WSMX to be interoperable with such systems where possible. In contrast with other systems, one of the biggest advantages of WSMO, WSML and WSMX is that it addresses complete aspects of SWS including the concepts, the ontology language for SWS and the execution environment.

7 Conclusion and Future Work

In this paper we showed how voice and data services can be seamlessly integrated using existing client as well as server VoIP systems based on widely used SIP signaling protocol and newly emerging technologies of Semantic Web Services. Within particular use case scenario, we described all phases of the whole process of making the cheapest call between two call participants each using different technologies for communication (VoIP and legacy PSTN). We showed, how SWS technology can facilitate dynamic and optimal integration of voice and data services with specific characteristics conforming users needs and preferences.

The implementation of comprehensive, efficient, and scalable SWS technology of industry-strength is still in its infancy. Use case scenario described in this paper is ideal testbed on which SWS concepts and prototypes as well as integration scenarios of voice and data services can be easily tested. In the future we plan to enhance this testbed with some more complex interactions including communication with end users during dynamic integration process in order improve selection of services based directly on user's approvals, e.g. offering suggestions for selection of services to a user through so called early media. Our ultimate goal for future research is to integrate Semantic Web Services with technologies of the Next Generation Networks (NGN) [23] to enhance the NGN call control servers with the semantic technology and to promote this way the provisioning of voice, data and video services for end-users.

Acknowledgment: This paper and works upon which this paper is based is supported by the Science Foundation Ireland under Grant No. 02/CE1/I131.

References

[1] Asg - adaptive services grid, http://asg-platform.org/.
[2] Asterisk - the open source linux public branch exchange, http://www.asterisk.org/.
[3] Dip: Data, information, and process integration with semantic web services, http://dip.semanticweb.org/.
[4] Sip express router, http://www.voip-info.org/tiki-index.php?page=sip+express+router.
[5] Telecom italia lab, http://starsip.telecomitalialab.com/.
[6] Itu-t recommendation h.323, 1999.
[7] S. A. Baset and H. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. Technical report, Columbia University, 2004.
[8] L. Cabral, J. Domingue, E. Motta, T. R. Payne, and F. Hakimpour. Approaches to semantic web services: an overview and comparisons. In *ESWS*, pages 225–239, 2004.
[9] B. Campbell and R. Sparks. Control of service context using sip request-uri (rfc 3087), Apr 2001.
[10] G. D. Caprio and E. Morello. Osa/parlay lab and experiments. In *Parlay Conference*, 2003.
[11] J. Davies, D. Fensel, and F. van Harmelen. *Towards The Semantic Web: Ontology-Driven Knowledge Management*. Wiley, 2002.
[12] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. D16.1v0.2 the web service modeling language wsml. Technical report, DERI Austria, 2005.
[13] A. Duke, M. Richardson, S. Watkins, and M. Roberts. Towards b2b integration in telecommunications with semantic web services. In *ESWC*, pages 710–724, 2005.
[14] J. R. et al. Sip: Session initiation protocol (rfc 3261), Jun 2002.
[15] D. Fensel and C. Bussler. The web service modeling framework wsmf. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.
[16] E. Motta, J. Domingue, L. Cabral, and M. Gaspari. Irs-ii: A framework and infrastructure for semantic web services. In *International Semantic Web Conference*, pages 306–318, 2003.
[17] E. Oren, M. Zaremba, M. Moran, and T. Vitvar. D13.0 overview and scope of wmsx. Technical report, DERI Ireland, 2004.
[18] M. Pistore and P. Roberti. D2.4.6 a - theoretical integration of web service discovery and composition. Technical report, University of Trento, Italy, Jun 2005.
[19] D. Roman, H. Lausen, and U. Keller. D2v1.0: Web service modeling ontology (wsmo). Technical report, DERI Austria, 2004.
[20] H. Schulzrine and J. Rosenberg. Sip caller preferences and callee capabilities (internet draft), Nov 2000.
[21] H. Schulzrinne and J. Rosenberg. Internet telephony: Architecture and protocols - an ietf perspective. *Computer Networks*, 31(3):237–255, 1999.
[22] K. Verma, K. Gomadam, A. P. Sheth, J. A. Miller, and Z. Wu. The meteor-s approach for configuring and executing dynamic web processes. Technical report, LSDIS, University of Georgia, USA, Jul 2005.
[23] N. Wilkinson. *Next Generation Network Services: Technologies and Strategies*. Wiley, 2002.
[24] M. Zaremba, M. Moran, and T. Haselwanter. D13.4v0.2 wsmx architecture. Technical report, DERI Galway, 2005.