Knowl Inf Syst (2007)
DOI 10.1007/s10115-006-0064-0

Knowledge and Information Systems

REGULAR PAPER

Ramón F. Brena · José L. Aguirre · Carlos Chesñevar · Eduardo H. Ramírez · Leonardo Garrido

Knowledge and information distribution leveraged by intelligent agents

Received: 7 February 2006 / Revised: 9 November 2006 / Accepted: 25 November 2006

Abstract Knowledge and Information distribution is indeed one of the main processes in Knowledge Management. Today, most Information Technology tools for supporting this distribution are based on repositories accessed through Web-based systems. This approach has, however, many practical limitations, mainly due to the strain they put on the user, who is responsible of accessing the right Knowledge and Information at the right moments. As a solution for this problem, we have proposed an alternative approach which is based on the notion of *delegation* of distribution tasks to synthetic agents, which become responsible of taking care of the organization's as well as the individuals' interests. In this way, many Knowledge and Information distribution tasks can be performed on the background, and the agents can recognize relevant events as triggers for distributing the right information to the right users at the right time.

In this paper, we present the JITIK approach to model knowledge and information distribution, giving a high-level account of the research made around this project, emphasizing two particular aspects: a sophisticated argument-based mechanism for deciding among conflicting distribution policies, and the embedding of JITIK agents in enterprises using the service-oriented architecture paradigm. It must be remarked that a JITIK-based application is currently being implemented for one of the leading industries in Mexico.

Keywords Knowledge management · Multiagent systems · Defeasible argumentation

R. F. Brena (✉) · J. L. Aguirre · E. H. Ramírez · L. Garrido
Centro de Sistemas Inteligentes, Tecnológico de Monterrey, 64849 Monterrey, N.L., México
e-mail: {ramon.brena, jlaguirre, eduardo.ramirez, leonardo.garrido}@itesm.mx

C. I. Chesñevar
Department of Computer Science and Engineering, Universidad Nacional del Sur, B8000CPB Bahía Blanca, Argentina
e-mail: cic@cs.uns.edu.ar

1 Introduction

Information and Knowledge (IK) are each day more valuable assets in modern organizations [3, 13, 32]. Indeed, a central concern in Knowledge Management (KM) [23, 33] is to facilitate *knowledge flow*, either within an organization or from/to other relevant actors. IK distribution systems could be visualized as a kind of *information switch*, which finds adequate routing paths for IK from sources to consumers—the latter being normally humans and members of the given organization (employees, partners, etc.).

Many technologies have been applied to distribute information to users, but they can be classified in "push" and "pull" modalities [18]. Push modality means that a system delivers information to a user without the user initiative (like the email delivery) whereas pull modality means that a user actively gets information from an electronic resource, typically the Web. The so-called "push technologies" (also called "Webcasting", "Netcasting" and other names) were proposed in the 90s as an alternative way to distribute IK to users [18]. In this type of systems, the user selects information "channels," and/or fills a "user profile" and gets updated information from these selected channels. Push systems sounded promising, because the user could be relieved from getting information updated, which is for sure a boring and repetitive task. Advocators of push systems claimed that users would be poured with a wealth of useful information by just staying "plugged" to their information sources. Though push systems have made their way since the 90's [22], their application as a general information distribution paradigm is nowadays fairly restricted [8], mainly due to problems related to the bandwidth efficiency, and the flooding of information to users.

An alternative user interaction paradigm (more general than push systems) has been proposed by Nicholas Negroponte [36]. His proposal consists of a *delegation paradigm* for user interaction with the computer, aiming to gradually replace the dominating "point-and-click" interaction paradigm. In the latter, the user asks directly for an action to be executed on the computer (for example, double-clicking an item), and an immediate answer is expected from the computer. In contrast, in the delegation paradigm, the user delegates some more or less permanent tasks to the computer, which is expected to execute the "appropriate" actions at the "right" times, instead of immediately.

In the last years Negroponte's ideas have been materialized in technological terms through a number of the emerging *agent technologies* [7, 48]. Agent technologies are by no means monolithic pieces of knowledge, but rather encompass many aspects which promote the delegation paradigm (such as communication, coordination techniques, cooperation, autonomy, competition, distribution, mobility, among other issues). Agents are long-life autonomous computational processes which thrive to achieve their goals (eventually goals delegated by a user or another agent), and which interact with the external world or with other agents [48]. Intelligent Agents offer the promise of much more modularity, flexibility, and adaptability when building complex systems, particularly *multiagent systems* [48]. This requires the development of specialized methodologies and algorithms which allow agents to communicate and interact by coordinating, cooperating, competing with other agents, etc.

In the last years, we have developed the JITIK multiagent framework [1, 9, 11, 14, 15], a knowledge and information distribution approach based on the idea of of *delegation* of distribution tasks to synthetic agents, which became responsible for taking care of the organization's as well as the individuals' interests. In this way, many Knowledge and Information distribution tasks are performed on the background, and the agents can recognize relevant events as triggers for distributing the right information to the right users at the right time. In our approach, users are represented by *personal agents*, and a community of specialized agents takes care of different aspects of information distribution. Currently, we are implementing a particular version of our framework for one of the leading industries in Mexico.[1]

In this paper, we present the JITIK approach to model knowledge and information distribution among the members of a large or distributed organization, giving a high-level account of the research made around this project. Our presentation will emphasize two particular aspects: the definition of a sophisticated argument-based mechanism for deciding among conflicting distribution policies, and the embedding of JITIK agents in enterprises using the service-oriented architecture paradigm. The rest of this paper is organized as follows: Sect. 2 summarizes the main aspects of the JITIK framework. We will describe how ontologies are handled within JITIK, and which kind of services are available. In particular, we will describe a formalization of information distribution which will be useful for modeling the argument-based mechanism for dealing with conflicts among distribution policies. Next, in Sect. 3, we present a case study, similar to an application we are currently implementing in an enterprise, which will serve as an example for specific technologies presented in later sections of the paper. Section 4 presents an argument-based formalization which allows to cope with such conflicts. We provide a detailed example, based on the case study, along with a description of the underlying argument-based formalism. Section 5 analyzes how JITIK can be embedded in enterprise systems by means of Web Services under the Service-Oriented Architecture paradigm. Section 6 discusses the main contributions of the proposed approach as well as some related work in the area. Finally, Sect. 7 summarizes the most important conclusions that have been obtained.

2 The JITIK approach

JITIK [1, 9, 10] is a multiagent-based system for disseminating pieces of IK among the members of a large or distributed organization, thus supporting a Knowledge-management function. It is aimed to deliver the right IK to the adequate people just-in-time.

The JITIK agent model is shown in Fig. 1. *Personal Agents* work on behalf of the members of the organization. They filter and deliver useful content according to user preferences. The Site Agent provides IK to the Personal Agents, acting as a broker between them and Service agents. *Service agents* collect and detect IK pieces that are supposed to be relevant for someone in the organization. Examples of service agents are the Web Service agents, which receive and process external requests, as well as monitor agents which are continuously monitoring sources of

[1] For the sake of confidentiality, we generalize this application in the following, omitting names.

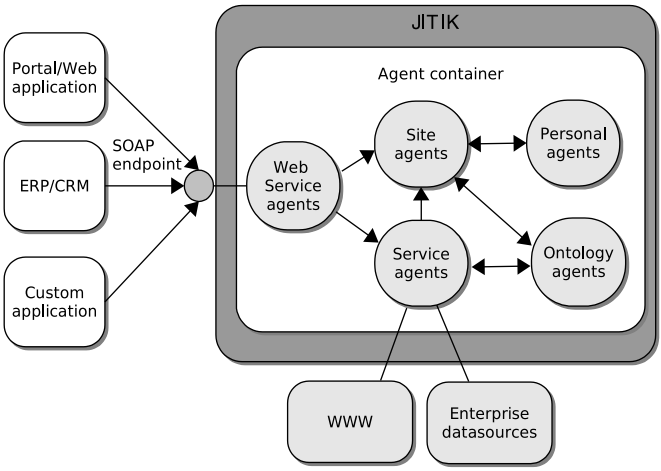

Fig. 1 The JITIK agent model

IK (Web pages, databases, etc.). Other Service agents monitor at time intervals the state of an IK resource, like a Web page, data in an enterprise's database, etc.

The *Ontology agent* contains knowledge about the interest areas for the members of the organization and about its structure [12]. That knowledge is hierarchically described in the form of taxonomies, usually one for interest areas and one describing the structure of the organization. For example, in an academic institution, the interest areas could be the science domains in which the institution is specialized, and the organizational chart of the institution gives the structure of the organization. This is going to be further described in the following sections.

Site agents are the heart of a "cluster" composed by one site agent and several personal agents served by the former. In an organization, clusters would be associated to departments, divisions, etc., depending on their size. Networks can be made up connecting several site agents. Distributed organizations like multinational companies would have a web of many connected site agents.

2.1 Ontology handling in JITIK

In JITIK, we declared users and information attributes through the use of the so-called *ontologies* [5]. Ontologies are structured representation of concepts, classes, individuals, properties, and other categories. We used open standards like DAML-OIL [24], which allow to publish in the Internet ontological knowledge in a way understandable both by humans and machines.

The way we stored ontologies is innovative in that it combines centralized and distributed ontology storage. We call this method "hybrid" local-global This allows to fine-tune the percentage of ontologies that is stored locally at each personal agent, and the remaining ontologies are going to be stored in a centralized, Internet-accessible ontology agent (OA). Personal agents are endowed with a "client ontology component" (COC) which gives it basic ontology handling capabilities. This arrangement works in the following way:

- Standard agents start with a subset of a common ontology, which is loaded at startup from an Internet resource. They use their local ontologies, handled by the COC, as long as the local knowledge suffices for the agent's activity.
- When further knowledge is required—for instance, an unrecognized term arrives from the other agent—the COC queries the OA, and receives a tailored addition to the basic ontology, that allows the agent to continue working. The COC stores locally the ontology addition so it could be used later.

Client agents try to fulfill their ontology knowledge needs using the knowledge in the COC. If necessary, the COC makes a query to the OA, and interprets and uses the answer, and eventually incorporates it into the local knowledge. In [16], we provide empirical evidence that shows that our hybrid local-global approach can be more efficient than both a completely centralized and a completely distributed approach.

2.2 JITIK services

Among the services provided by JITIK, we have the following.

Recommendation services: A user's profile is represented by a set of points in the taxonomies, as each user could have many interests and could be located at different parts of the organizational structure. As JITIK keeps track of user interests and preferences, it is able to recommend content to users on demand. Recommended content may be used in Portals or Web applications.

Subscription services: JITIK allows users to subscribe to changes in specific areas. Also, users may customize the media and frequency of JITIK notifications using simple Web-based interfaces. Rules may be defined so that messages relative to certain topics are handled with higher priorities. A rule may state that several alerts should be sent to their cell-phone via SMS, and also define that interest-area messages be sent in a weekly summary via email. Organization managers may set high-level distribution rules.

Content distribution services: Enterprise applications can deliver content to the system using its semantic-based content distribution services. When new content is received, it is classified and distributed to those users who could be interested. Users receive the notifications of new content as specified by their own rules.

2.3 Formalizing IK distribution in JITIK

As explained earlier, JITIK aims at disseminating pieces of IK among the members of a large or distributed organization, thus supporting a Knowledge-management function. The Site Agent is in charge of providing IK to the Personal Agents, acting as a broker between them and Service agents. Clearly, in large or distributed organizations, there are usually complex decision-making situations regarding IK distribution, specially in the presence of *potentially incomplete information* concerning metadata and user profiles, as well as *competing policies*, which may be complicated and could include several exceptions. Therefore, it is important that

Site agents are provided with appropriate knowledge representation and inference capabilities to solve such problems.

Next, we will provide a basic formalization for the main problem a JITIK Site Agent has to solve, namely *distributing items among users according to possibly conflicting policies*. Consider a set $I = \{i_1, i_2, \ldots, i_k\}$ of information items, which has to be distributed among a set $U = \{u_1, \ldots, u_s\}$ of users. Every item $i \in I$ should be delivered to only a distinguished subset $User \subseteq U$. To accomplish this task, the Site Agent will apply a distribution policy p that can be formally defined as a mapping $p : I \rightarrow \wp(U)$. Distributing an item i to a user u is *compliant* with a policy p when $(i, \{\ldots, u, \ldots\}) \in p$.

In any real-world organization, it is clear that policies are not formulated in this way, but instead they are specified by a number of *constraints* enforced by the organization (e.g., access rights). If P is a set of possible policies in the organization, given two policies $p_1, p_2 \in P$, we say they are in *conflict* whenever $(i, \{\ldots, u, \ldots\}) \in p_1$ but $(i, \{\ldots, u, \ldots\}) \notin p_2$, or viceversa. A conflict means that an information item i cannot be compliant with two policies p_1 and p_2 at the same time. We can define a *dominance* partial order $\prec$ among possible policies in P, writing $p_1 \prec p_2$ to indicate that a policy p_2 is preferred over policy p_1 in case they are in conflict. In this setting, the "information distribution problem" to be solved by a JITIK Site Agent could then be recast as follows: *Send every information item $i \in I$ to a user $u \in U$ following a distribution p iff p is compliant with every nondominated policy $p' \in P$.*[2]

Note that dominance characterizes a transitive ordering among policies, with the following particular feature: a policy p_i can be dominated by another policy p_j, making p_i not valid. However, p_j can on its turn be dominated by another policy p_k. If that is the case, the policy p_i may be perhaps valid again (as the policy p_j was "defeated" by policy p_k). Such a situation can be recast in an argument-based system for defeasible reasoning [17], in which *defeasible* rules (i.e., rules supporting conclusions that may be no longer valid when new information is available) are allowed. By chaining defeasible rules to reach a conclusion, we have *arguments* instead of proofs. Arguments may compete, rebutting each other, so a *process* of argumentation is a natural result of the search for arguments. Adjudication of competing arguments must be performed, comparing arguments in order to determine what beliefs are ultimately accepted as *warranted* or *justified*. Preference among conflicting arguments is defined in terms of a *preference criterion* which establishes a relation " $\preceq$ " among possible arguments.

3 A case study

In this section, we will illustrate how JITIK Agents work in the context of a real-world problem. To do so, we will describe a generic application similar to the one we are currently implementing for one of the leading industries in Mexico.[3] Consider a "news center" in a big enterprise, where a flow of news is originated, both

[2] Note that characterizing p depends on the specific sets U, I, and P under consideration. Here, we do not discuss the problem of finding out whether such a mapping actually exists, but rather focus on enforcing dominance on conflicting policies.

[3] For the sake of confidentiality, some relevant details are generalized and names are intentionally omitted.

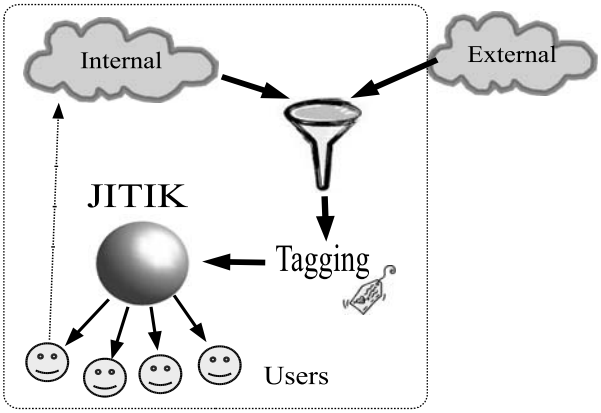

Fig. 2 Information flow in a News Center (case study)

from internal and external news. News are "tagged" both by human employees and by automated text classifiers. Once the information is tagged, the problem is to deliver those news to the right persons in the company by considering their information needs as well as their preferences and roles in the company. Information flow of this case study is depicted in Fig. 2.

Individual interests and some information needs are taken into account by means of news classification into a set of categories which are organized in a tree. In this tree, a broad area like "marketing" has subareas like "TV advertising," which, in turn, may have subareas. Users select a collection of nodes in this tree in order to describe their information interests. Some of the information needs of users are determined "a priori" by the company without user intervention, as, for instance, when an information area matches an employee's department or role (for example, anybody in the marketing department is likely to receive useful information on marketing). Whether the individual policies prevail over corporate ones when they are in conflict, is up to the particular enterprise. It is also the case that some information items may correspond to sensitive information, so that they are not delivered to anybody (so that a level of authorization or permission is required).

Such distribution policies can be formulated in terms of the formalization we provided earlier for information distribution. For example, consider the problem of distributing information according to the hierarchical classification tree of topics. Suppose a user $u \in U$ has selected as his/her interests the set of topics $T = \{t_1, \ldots, t_n\}$ which are nodes from the topic classification tree. We are going to represent the "topic tag" of an information i by $\tau(i)$ and the "subarea" relation by ς. Then, we are defining a policy where u is going to receive any information i such that $(\tau(i), t_k) \in \varsigma^*$, where $t_k \in T$ and ς^* is the transitive closure of ς, meaning that if a user selects a topic t_k he/she is automatically interested in any subtopic of t_k.

In the next section, we will analyze how to model reasoning capabilities in JITIK Site Agents in terms of Defeasible Logic Programming [21], an argument-based logic programming formalism. As a basis for our discussion, we will consider a particular case study related to a news center.

4 Modeling reasoning in JITIK site agents via argumentation

As explained in Sect. 2.3, Site agents will be in charge of applying such distribution policies, and consequently they should be provided with appropriate knowledge representation and inference capabilities to solve such problems. In order to provide a rationally justified procedure for deciding which IK item goes to which user, we will rely on an argument-based logic programming language called Defeasible Logic Programming (DeLP).

Logical models of defeasible argumentation [17] have evolved in the last decade as a successful approach to formalize defeasible, commonsense reasoning. Recent research has shown that argumentation can be integrated in a growing number of real-world applications in a broad scope of areas such as legal reasoning, natural language processing, clustering, analysis of news reports [25], and others.

4.1 Defeasible logic programming: formalizing argumentation in JITIK

In what follows, we will summarize the fundamentals of Defeasible logic programming (DeLP) [21] in a particular general-purpose defeasible argumentation formalism based on logic programming.[4]

A defeasible logic program in a set $K = (\Pi, \Delta)$ of Horn-like clauses, where Π and Δ stand for sets of strict and defeasible knowledge, respectively. The set Π of strict knowledge involves *strict rules* of the form $p \leftarrow q_1, \ldots, q_k$ and *facts* (strict rules with empty body), and it is assumed to be *noncontradictory* (i.e., an atom P and its negation cannot be derived from Π). The set Δ of defeasible knowledge involves *defeasible rules* of the form $p \leftarrowtail q_1, \ldots, q_k$, which stands for "$q_1, \ldots q_k$ provide a *tentative reason* to believe p." The underlying logical language is that of extended logic programming, enriched with a special symbol "$\leftarrowtail$ " to denote defeasible rules. Both default and classical negation are allowed (denoted `not` and $\sim$, resp.). Syntactically, the symbol "$\leftarrowtail$ " is all that distinguishes a *defeasible* rule $p \leftarrowtail q_1, \ldots q_k$ from a *strict* (nondefeasible) rule $p \leftarrow q_1, \ldots, q_k$. DeLP rules are thus Horn-like clauses to be thought of as *inference rules* rather than implications in the object language. Deriving literals in DeLP results in the construction of *arguments*.

Definition 4.1 (Argument) *Given a DeLP program $\mathcal{P}$, an* argument $\mathcal{A}$ *for a query q, denoted $\langle \mathcal{A}, q \rangle$, is a subset of ground instances of defeasible rules in $\mathcal{P}$ and a (possibly empty) set of default ground literals "*`not` *L," such that: 1) there exists a* defeasible derivation *for q from $\Pi \cup \mathcal{A}$; 2) $\Pi \cup \mathcal{A}$ is noncontradictory (i.e, $\Pi \cup \mathcal{A}$ does not entail two complementary literals p and $\sim p$ (or p and* `not` *p)), and 3) $\mathcal{A}$ is minimal with respect to set inclusion.*

An argument $\langle \mathcal{A}_1, Q_1 \rangle$ is a subargument *of another argument $\langle \mathcal{A}_2, Q_2 \rangle$ if $\mathcal{A}_1 \subseteq \mathcal{A}_2$. Given a DeLP program $\mathcal{P}$, $Args(\mathcal{P})$ denotes the set of all possible arguments that can be derived from $\mathcal{P}$.*

The notion of defeasible derivation corresponds to the usual query-driven derivation used in logic programming, performed by backward chaining on both

[4] For an in-depth analysis, the reader is referred to ref. [21].

strict and defeasible rules; in this context a negated literal $\sim p$ is treated just as a new predicate name no_p. Minimality imposes a kind of "Occam's razor principle" [42] on arguments. The noncontradiction requirement forbids the use of (ground instances of) defeasible rules in an argument $\mathcal{A}$ whenever $\Pi \cup \mathcal{A}$ entails two complementary literals.

Definition 4.2 (Counterargument—Defeat) *An argument* $\langle \mathcal{A}_1, q_1 \rangle$ *is a* counterargument *for an argument* $\langle \mathcal{A}_2, q_2 \rangle$ *iff*

1. *There is an subargument* $\langle \mathcal{A}, q \rangle$ *of* $\langle \mathcal{A}_2, q_2 \rangle$ *such that the set* $\Pi \cup \{q_1, q\}$ *is contradictory.*
2. *A literal* not q_1 *is present in some rule in* $\mathcal{A}_1$.

A partial order $\preceq \subseteq Args(\mathcal{P}) \times Args(\mathcal{P})$ *will be used as a* preference criterion *among conflicting arguments. An argument* $\langle \mathcal{A}_1, q_1 \rangle$ *is a* defeater *for an argument* $\langle \mathcal{A}_2, q_2 \rangle$ *if* $\langle \mathcal{A}_1, q_1 \rangle$ *counterargues* $\langle \mathcal{A}_2, q_2 \rangle$*, and* $\langle \mathcal{A}_1, q_1 \rangle$ *is preferred over* $\langle \mathcal{A}_2, q_2 \rangle$ *wrt* $\preceq$*. For cases (1) and (2) earlier, we distinguish between proper* and *blocking defeaters* as follows:

- *In case 1, the argument* $\langle \mathcal{A}_1, q_1 \rangle$ *will be called a* proper defeater *for* $\langle \mathcal{A}_2, q_2 \rangle$ *iff* $\langle \mathcal{A}_1, q_1 \rangle$ *is strictly preferred over* $\langle \mathcal{A}, q \rangle$ *wrt* $\preceq$.
- *In case 1, if* $\langle \mathcal{A}_1, q_1 \rangle$ *and* $\langle \mathcal{A}, q \rangle$ *are unrelated to each other, or in case 2,* $\langle \mathcal{A}_1, q_1 \rangle$ *will be called a* blocking defeater *for* $\langle \mathcal{A}_2, q_2 \rangle$.

Specificity [42] is used in DeLP as a syntax-based criterion among conflicting arguments, preferring those arguments which are *more informed* or *more direct* [42, 43]. However, other alternative partial orders could also be used.

As defeaters are arguments, they can on their turn be defeated by other arguments. This leads to a recursive analysis, in which many alternative *argumentation lines* can be computed. An *argumentation line* starting in an argument $\langle \mathcal{A}_0, Q_0 \rangle$ (denoted $\lambda^{\langle \mathcal{A}_0, q_0 \rangle}$) is a sequence $[\langle \mathcal{A}_0, Q_0 \rangle, \langle \mathcal{A}_1, Q_1 \rangle, \langle \mathcal{A}_2, Q_2 \rangle, \ldots, \langle \mathcal{A}_n, Q_n \rangle \ldots]$ that can be thought of as an exchange of arguments between two parties, a *proponent* (evenly-indexed arguments) and an *opponent* (oddly-indexed arguments). Each $\langle \mathcal{A}_i, Q_i \rangle$ is a defeater for the previous argument $\langle \mathcal{A}_{i-1}, Q_{i-1} \rangle$ in the sequence, $i > 0$. In the same way as in an exchange of arguments between human contenders, an argumentation line $\lambda^{\langle \mathcal{A}_0, q_0 \rangle}$ is *won* by the proponent if he/she presents the last argument (i.e., $\lambda^{\langle \mathcal{A}_0, q_0 \rangle}$ has odd length $k = 2p + 1$, $p >= 0$); otherwise, the line is *lost*. An argument $\langle \mathcal{A}_0, q_0 \rangle$ is *warranted* iff all argumentation lines rooted in $\langle \mathcal{A}_0, q_0 \rangle$ are won. In order to avoid *fallacious* reasoning, there are additional constraints (viz., disallowing circular argumentation, enforcing the use of proper defeaters to defeat blocking defeaters, etc.[5]) on such an argument exchange to be considered rationally valid. On the basis of such constraints, argumentation lines can be proven to be finite. Given a DeLP program $\mathcal{P}$, solving a query q_0 wrt $\mathcal{P}$ can result in three different answers: a) YES (there is some *warranted* argument $\langle \mathcal{A}_0, q_0 \rangle$); b) NO (there is some *warranted* argument $\langle \mathcal{A}_0, \sim q_0 \rangle$); c) UNDECIDED (neither a) nor b) hold). An efficient inference engine based on an extension of Warren's Abstract machine for PROLOG is available to solve queries in DeLP.[6]

[5] For an in-depth treatment of dialectical constraints in DeLP, the reader is referred to ref. [21].

[6] See http://lidia.cs.uns.edu.ar/DeLP

4.2 Reasoning in JITIK under DeLP: proposed approach

The JITIK framework, as it stands, can take into consideration hierarchies for users and content classification for determining how distribution rules are to be applied. In the case of policies with exceptions, or competing policies, specialized criteria have to be explicitly encoded in both Site and Personal agents. In many respects, such an approach is undesirable. On the one hand, such changes involve modifying the underlying decision algorithm. The correctness of such changes may be difficult to test, as unexpected side-effects might arise for new future cases. On the other hand, the knowledge engineer should be able to encode knowledge as declaratively as possible, including the possibility of representing competing policies. Such knowledge should be independent of the rational procedure for determining which is the winning policy when conflicting situations arise.

Our proposal consists of integrating the JITIK framework with DeLP, incorporating distribution policies for Site Agents explicitly in terms of defeasible logic programs. As explained in Sect. 2, a JITIK Site Agent Ag_S is responsible for distributing IK among different Personal Agents $Ag_1, \ldots, Ag_n$. We will use DeLP programs to represent the knowledge of these agents. Thus, preferences of different Personal Agents $Ag_1, \ldots, Ag_n$ will be represented as DeLP programs $\mathcal{P}_{Ag_1}, \ldots, \mathcal{P}_{Ag_n}$. Distribution policies and preferences of the Site Agent Ag_S will be represented by another DeLP program $\mathcal{P}_S$. In contrast with the programs associated with Personal Agents, this program $\mathcal{P}_S$ will contain corporate rules defining hierarchies and (possibly conflicting) policies for IK distribution among personal agents.

Given a list $L = [Item_1, \ldots, Item_i]$ of IK items to be distributed by the Site Agent Ag_S among different Personal Agents $Ag_1, \ldots, Ag_n$, a distinguished predicate $distribute(I, User)$ will be used to determine whether a particular IK item $I \in L$ is intended to be delivered to a specific user $User$. This query will be solved by the DeLP inference engine on the basis of a program $\mathcal{P}$ which will take into account the Site Agent's knowledge, the metadata corresponding to the incoming items to be distributed and the personal preferences of the different users involved. This is made explicit in an algorithm shown in Fig. 3. Solving queries based on the *distribute* predicate wrt the DeLP inference engine will automate the decision making process for Site Agents, providing a rationally justified decision even for very complex cases, as we will see in the next section.

4.3 An example: distributing items at the news center

In this section, we take the case study of Sect. 3 as an example of how DeLP is integrated into the JITIK system to distribute IK items to users using the described argumentation framework. So, we are assuming a "News Center" that distributes news to users according to their interests as well as the corporate policies. In such a corporate environment, there would be people with different rights and responsibilities (CEO, managers, supervisors, etc.). These people (users) will belong to different areas of the organization (production, marketing, etc.), and will have different personal interests and preferences which are characterized by their cor-

ALGORITHM DistributeItems
{*Executed by Site Agent* Ag_S *to decide distribution of items in* L}
INPUT: List $L = [item_1, \ldots, item_k]$ of incoming items
DeLP program $\mathcal{P}_S$ for Site Agent Ag_S
DeLP programs $\mathcal{P}_1, \ldots, \mathcal{P}_n$ for Personal Agents depending from Ag_S
OUTPUT: Item distribution to Personal Agents
according to policies and user preferences
BEGIN
$\mathcal{P}'_S := \mathcal{P}_S \cup \{info(item_1), \ldots, info(item_k)\}$
{*Encode incoming items as new facts for Site Agent*}
FOR every item $I \in L$
FOR every Personal Agent Ag_i supervised by Ag_S
Let $\mathcal{P} = \mathcal{P}'_S \cup \mathcal{P}_{Ag_i}$
Using program $\mathcal{P}$, solve query $distribute(Item, Ag_i)$
IF $distribute(Item, Ag_i)$ is warranted
THEN
Send message I to agent Ag_i
END

Fig. 3 Algorithm for knowledge distribution using DeLP in a JITIK site agent

responding Personal Agents. In our example, IK items will correspond to memos, which have to be delivered by a Site Agent to different users according to the organization policies. Personal interests and preferences of the different users involved should also be taken into account.

Within the case study organization, areas and topics are organized in *hierarchies*. Thus, for example, a hierarchy of topics for news could be "computers—hardware—processors." The Site Agent is required to take this into account, performing inheritance reasoning to infer consequences related to subareas: if a user is not interested in news related to hardware, he will not be interested in news related to processors either. Note that other organization policies could add *exceptions* to such hierarchies, e.g. by stipulating that a certain news item is mandatory, and should be delivered without consulting the user preferences.

In our example, IK items made available from the organization to the Site Agent will correspond to different news, which will be encoded with a predicate $info\ (Id, A, L, M, T, S)$, meaning that the news item with unique identifier Id is about area A and it can be accessed by users of at least level L. Other attributes associated with the news item are whether it is mandatory ($M = 1$) or optional ($M = 0$), top secret ($T = 1$) or not ($T = 0$), and is originated at source S. Thus, the fact

$$info(id_3, computers, manager, 0, 0, marketing) \quad \leftarrow$$

indicates that the news item id_3 is about $computers$, it is intended at least for managers, it is not mandatory nor secret, and it has been produced by the department of marketing.

4.4 Characterizing organization knowledge in site and personal agents

Figure 4 shows a sample DeLP code associated with a Site and a Personal agent in our organization.[7] Strict rules s_1–s_9 characterize permissions and extract informa-

[7] Note that we distinguish strict rules, defeasible rules, and facts by using s_i, d_i and f_i as clause identifiers, respectively.

Site Agent Knowledge

Strict rules

$s_1)$ $allowed(I, U) \leftarrow info(I, A, L, M, T, S), permissions(U, L).$
$s_2)$ $isAbout(I, A) \leftarrow info(I, A, L, M, T, S)$
$s_3)$ $isAbout(I, A) \leftarrow subField(SuperA, A), isAbout(I, SuperA).$
$s_4)$ $permissions(U, X) \leftarrow depends(X, Y), permissions(U, Y).$
$s_5)$ $depends(X, Y) \leftarrow subordinate(X, Y).$
$s_6)$ $depends(X, Z) \leftarrow subordinate(Y, Z), depends(X, Y).$
$s_7)$ $source(I, S) \leftarrow info(I, \_, \_, \_, \_, S).$
$s_8)$ $mandatory(I) \leftarrow info(I, \_, \_, 1, \_, \_).$
$s_9)$ $topsecret(I) \leftarrow info(I, \_, \_, \_, 1, \_).$

Defeasible rules

$d_1)$ $interest(I, U) \prec isAbout(I, A), interestField(A, U).$
$d_2)$ $distribute(I, U) \prec allowed(I, U), mandatory(I, U).$
$d_3)$ $\sim mandatory(I, U) \prec permissions(U, manager), \sim interest(I, U),$ not $topsecret(I).$
$d_4)$ $distribute(I, U) \prec allowed(I, U), interest(I, U).$

Facts

Granted Permissions within the organization

$f_1)$ $permissions(joe, manager) \leftarrow$
$f_2)$ $permissions(peter, everybody) \leftarrow$
$f_3)$ $permissions(dana, ceo) \leftarrow$

People Hierarchy

$f_4)$ $subordinate(everybody, manager) \leftarrow$
$f_5)$ $subordinate(manager, ceo) \leftarrow$

Field Hierarchy

$f_6)$ $subField(hardware, computers) \leftarrow$
$f_7)$ $subField(processors, hardware) \leftarrow$

Information Items as facts

$f_8)$ $info(id_1, computers, everybody, 0, 0, external) \leftarrow$
$f_9)$ $info(id_2, computers, everybody, 0, 0, techdept) \leftarrow$
$f_{10})$ $info(id_5, processors, manager, 1, 1, techdept) \leftarrow$

Personal Agent Knowledge

Defeasible rules

$d'_1)$ $\sim interest(I, joe) \prec isAbout(I, A), interestField(A, joe).$ $source(I, S), \sim relies(joe, S).$
$d'_2)$ $\sim interest(I, joe) \prec isAbout(I, A), interestField(A, joe),$ $isAbout(I, SuperA), \sim interestField(SuperA, joe).$

Facts

User Preferences

$f'_1)$ $interestField(computers, joe) \leftarrow$
$f'_2)$ $\sim interestField(hardware, joe) \leftarrow$
$f'_3)$ $relies(joe, techdept) \leftarrow$
$f'_4)$ $\sim relies(joe, external) \leftarrow$

Fig. 4 DeLP code for a site agent and a personal agent in JITIK

tion from news. Rule s_1 defines that a user P is *allowed* access to item I if he/she has the required *permissions*. Granted permissions are given as facts (f_1, f_2 and f_3). Permissions are also propagated using the strict rules s_4, s_5 and s_6, where the binary predicate *depends* establishes the organization hierarchy, stating that the first argument person is (transitively) subordinated to the second one. This predicate is calculated as the transitive closure of a basic predicate *subordinate* (defined by facts f_4 and f_5), which establishes subordinate relationships pairwise. Thus, having, e.g., granted permissions as CEO allows the CEO to have access

to every memo corresponding to lower level permissions. Note that the predicate *subordinate* uses generic *roles* as arguments, not specific person identifiers.

Rule s_2 and s_3 define the predicate $isAbout(I, A)$ as an information hierarchy among subfields. The basic case corresponds to a subfield for which specific information is available (rule s_2). Note that in our particular example facts f_6 and f_7 define the basic relationships in this hierarchy. Finally, rules $s_7 - s_9$ define auxiliary predicates *source*, *mandatory* (yes/no) and *topsecret* (yes/no) which allow to extract these particular attributes from the news to be distributed that just extract information from *info*, facts, simplifying the subsequent analysis.

Let us now consider the defeasible rules for our Site Agent. Rule d_1 defines when an item I is usually of interest for a specific user U, on the basis of the user's personal preferences. Rule d_2 and d_4 define a policy for news distribution in our organization: a) an item (memo) I should be delivered to a user U if he is allowed to read this news item, and it is mandatory for him to read it; b) an item I should be delivered to a user U if he is allowed to read it, and it is interesting for him. Rule d_3 provides an exception for mandatory news: users which have at least permission as managers are not forced to read news they are not interested in, *unless* they are top secret ones.[8]

Finally, let us consider the DeLP program associated with a particular Personal Agent (e.g., Joe). A number of facts represent Joe's preferences: which are his interest fields, and his personal belief about other parts of the organization (e.g., reliability with respect to the source of incoming memo).[9] Joe can provide also a number of defeasible rules associated with his preferences. Rule d_1' establishes that Joe is not interested in a news item coming from an unreliable source. Rule d_2' defines how to handle "negative inheritance" within the hierarchy of interests: Joe is not interested in any area A which is a subarea of another area $SuperA$, such that $SuperA$ is not interesting for him (e.g., if he is interested in computers but not interested in hardware, he will not be interested in a news item about processors, as processors are a subarea of hardware).

4.5 Solving conflicts for information distribution as DeLP queries

Let us assume that there is a list of information items $[News_1, News_2, News_5]$ corresponding to news items to be distributed by our Site Agent, which encodes organization policies as a DeLP program $\mathcal{P}_S$. By applying the algorithm given in Fig. 3, these items will be encoded temporarily as a set $\mathcal{P}_{items} = \{info(News_1), info(News_2), info(News_5)\}$ (see Fig. 4).

For the sake of simplicity, we will assume that there is only one single Personal Agent involved, associated with a specific user joe, whose role is $manager$. Joe's Personal Agent mirrors his preferences in terms of a DeLP program $\mathcal{P}_{joe} = \{d_1', d_2', f_1', f_2', f_3', f_4'\}$, which together with $\mathcal{P}_S$ and $\mathcal{P}_{items}$ will provide the knowledge necessary to decide which IK items should be delivered to this specific user. Following the algorithm in Fig. 3, the Site Agent will have to solve the queries $distribute(id_1, joe)$, $distribute(id_2, joe)$ and $distribute(id_5, joe)$ wrt the DeLP program $\mathcal{P}_S \cup \mathcal{P}_{items} \cup \mathcal{P}_{joe}$. We will show next how every one of these

[8] Note how this last condition is expressed in terms of default negation not in rule d_3.

[9] Note the use of explicit negation in these predicates.

queries is solved in different examples that show how DeLP deals with conflicts among organization policies and user preferences.

Example 4.1 Consider the query $distribute(id_1, joe)$. In this case the DeLP inference engine will find the argument $\langle \mathcal{A}_1, distribute(id_1, joe)\rangle$, with[10] $\mathcal{A}_1 =$

$$\begin{aligned} \{distribute(id_1, joe) \prec\ & allowed(id_1, joe), \\ & interest(id_1, joe); \\ interest(id_1, joe) \prec\ & isAbout(id_1, computers), \\ & interestField(computers, joe)\} \end{aligned}$$

However, in this case, a defeater $\langle \mathcal{A}_2, \sim interest(id_1, joe)\rangle$ for the argument $\langle \mathcal{A}_1, distribute(id_1, joe)\rangle$ will be found, with $\mathcal{A}_2 =$

$$\begin{aligned} \{\sim interest(id_1, joe) \prec\ & isAbout(id_1, computers) \\ & interestField(computers, joe) \\ & source(id_1, external), \\ & \sim relies(joe, external).\} \end{aligned}$$

Note that in this case, id_1 comes from an external source, and according to joe's preference criteria, external sources are unreliable. Hence, the Site Agent will not deliver this information item to him. There is a single argumentation line with two nodes (even length) $[\langle \mathcal{A}_1, distribute(id_1, joe)\rangle, \langle \mathcal{A}_2, \sim interest(id_1, joe)\rangle]$. There are no other arguments to consider, and $\langle \mathcal{A}_1, distribute(id_1, joe)\rangle$ is not warranted.

Example 4.2 Consider now the query $distribute(id_2, joe)$. There is an argument $\langle \mathcal{B}_1, distribute(id_1, joe)\rangle$, with $\mathcal{B}_1 =$

$$\begin{aligned} \{distribute(id_2, joe) \prec\ & allowed(id_2, joe), \\ & interest(id_2, joe); \\ interest(id_2, joe) \prec\ & isAbout(id_2, computers), \\ & interestField(computers, joe)\} \end{aligned}$$

This argument has no defeaters, and hence only one associated argumentation line $[\langle \mathcal{B}_1, distribute(id_1, joe)\rangle]$. The original argument is therefore warranted.

Example 4.3 Finally consider the query $distribute(id_5, joe)$. There is an argument $\langle \mathcal{C}_1, distribute(id_1, joe)\rangle$, with $\mathcal{C}_1 =$

$$\begin{aligned} \{distribute(id_5, joe) \prec\ & allowed(id_5, joe), interest(id_5, joe); \\ interest(id_5, joe) \prec\ & isAbout(id_5, computers), \\ & interestField(computers, joe)\} \end{aligned}$$

[10] For the sake of clarity, we use semicolons to separate elements in an argument $\mathcal{A} = \{e_1 ; e_2 ; \ldots ; e_k \}$.

However, in this case, a defeater $\langle \mathcal{C}_2, \sim interest(id_5, joe)\rangle$ for the argument $\langle \mathcal{C}_1, distribute(id_5, joe)\rangle$ can be found, with $\mathcal{C}_2 =$

$$\begin{aligned}\{\sim interest(id_5, joe) \prec\ & isAbout(id_5, computers),\\ & interestField(computers, joe),\\ & isAbout(id_5, hardware),\\ & \sim interestField(hardware, joe).\}\end{aligned}$$

As in Example 4.1, the argument $\langle \mathcal{C}_1, distribute(id_1, joe)\rangle$ is not warranted. The DeLP inference engine searches then for alternative arguments for $distribute(id_5, joe)$. There is another one, namely $\langle \mathcal{D}_1, distribute(id_5, joe)\rangle$, with $\mathcal{D}_1$=

$$\{distribute(id_2, joe) \prec allowed(id_2, joe), mandatory(id_5, joe)\}$$

which in this case is defeated by another argument $\langle \mathcal{D}_2, \sim mandatory(id_5, joe)\rangle$, with $\mathcal{D}_2 =$

$$\begin{aligned}\{\sim mandatory(id_5, joe) \prec\ & permissions(joe, manager),\\ & \sim interest(id_5, joe),\\ & \text{not } topsecret(id_5);\\ \sim interest(id_5, joe) \prec\ & isAbout(id_5, computers),\\ & interestField(computers, joe),\\ & isAbout(id_5, hardware),\\ & \sim interestField(hardware, joe)\}\end{aligned}$$

which is on its turn defeated by a third, empty argument $\langle \mathcal{D}_3, topsecret(id_5)\rangle$, with $\mathcal{D}_3 = \emptyset$ (note that $topsecret(id_5)$ is logically entailed by the strict knowledge of the Site Agent, and hence no defeasible information is needed). Argument $\langle \mathcal{D}_3, topsecret(id_5)\rangle$ defeats $\langle \mathcal{D}_2, \sim mandatory(id_5, joe)\rangle$, reinstating the argument $\langle \mathcal{D}_1, distribute(id_5, joe)\rangle$. Thus an argumentation line of three arguments is associated with $\langle \mathcal{C}_1, distribute(id_1, joe)\rangle$, so that in this particular case the argument $\langle \mathcal{D}_1, distribute(id_5, joe)\rangle$ is warranted.

After solving the different queries as shown in the previous examples, the Site Agent will proceed to deliver only news items id_2 and id_5 to joe's Personal Agent, but not news item id_1.

5 Embedding JITIK agents in an enterprise system using Web Services

As stated in Sect. 2, the JITIK framework goals demand a seamless integration of its agent components with other technologies like Web servers, databases, etc. Indeed, such lack of integration in other agent-based systems has been one of the main factors hindering their widespread use [48]. Thus, we have proposed an alternative approach to integrating agents with conventional Internet-based software. Our proposal is by no means a radical departure from other ones, but we have shown [40] that in many contexts it is much simpler, cleaner, and efficient than competing ones.

For integrating JITIK with Internet-based open systems, we relied on the so-called "Service-Oriented approaches" (SOA) [2], which could be described as: "the architectural style that supports loosely coupled services to enable business flexibility in an interoperable, technology-agnostic manner. SOA consists of a composite set of business-aligned services that support a flexible and dynamically reconfigurable end-to-end business processes realization using interface-based service descriptions" [6].

The Service-Oriented Architecture paradigm (SOA) improves the abstraction and flexibility on which information systems may be designed and integrated when compared to code-centric previous approaches. There exists three roles or *actors* in a SOA: a *service consumer*, first locates a *service provider* in a *service registry* or broker, then binds the service description, and finally performs an invocation to the provider. The loose coupling principle suggests that the operations between actors should be carried out interchanging as little information as possible, usually through message passing.

Even though the SOA architectural style is not bound to any particular implementation technology, the Web Services standards are becoming a natural and common choice for implementing it. For the purposes of this work, we understand a Web Service as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL [51]). Other systems interact with the Web Service in a manner prescribed by its description using SOAP-messages [52], typically conveyed using HTTP with an XML [50] serialization in conjunction with other Web-related standards [53]."

Despite a continuous maturing of MultiAgent Systems technologies, they are still far from becoming mainstream in enterprise applications [49], delivering their promise of robust and flexible new-generation information systems. Some reasons for this can be traced back to methodological or cultural issues [47], but we think there are important architectural issues as well. In fact, current approaches for integrating agents and composing their function and services rely on low-level communication mechanisms [31], which have very little to do with enterprise-wide integration issues.

The SOA approach and Web Services technologies could be used to overcome the technical limitations required to integrate a MultiAgent System into an enterprise service-oriented environment. In a similar way, the SOA ideas can be translated to the application internals to allow agents to provide a stable framework for building robust Agent-based applications. A number of recent works report how the SOA approach has been applied in several AI and multiagent systems and components (e.g., [41, 54]).

In conformance with the SOA ideas, the main architectural principle for integrating a MultiAgent system into an enterprise environment consists of decoupling applications through the exposure of "coarse-grained" service interfaces. As shown in Fig. 5, the underlying metaphor used to determine the design strategy was the aim to create a "black-box" in which agents can live and perform complex knowledge-intensive tasks. Neither enterprise applications nor its developers should be aware that a service is provided by agents if the system offers a standard SOAP endpoint as interface. Several integration architectures for agent-based application exist (e.g., ref. [46]). However, a particular approach called "Embedded

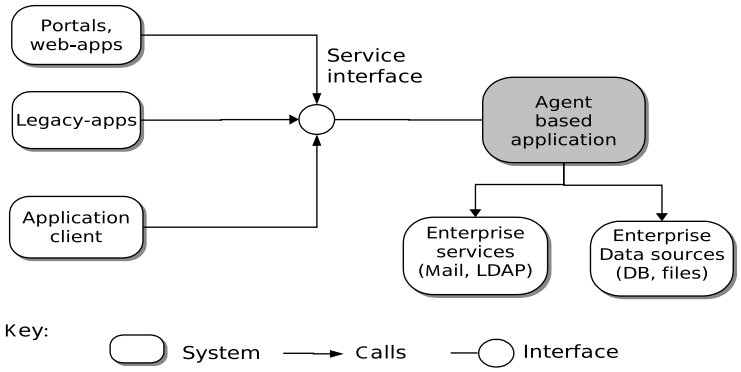

Fig. 5 Decoupled architecture (top-level view)

Web Service Architecture (EWSA)" has proven to be enough simple and effective for this purpose, with a clear focus on agent-to-application communication. The proposal's main idea suggests embedding a Web Server into an agent-based application and defining an interaction model upon which software agents and Web-components [44] may communicate in order to provide services. Details on the approach can be found in ref. [40].

Besides application integration, the service-oriented approach has gained momentum for intra-application component communication. The SOA paradigm has been reinforced by a new generation of noninvasive lightweight frameworks (e.g. [28, 29]) that leverage contemporary software design principles and allow application developers to produce enterprise-level, yet simple applications. The use of a lightweight framework like Spring [26] helps agent project teams to decompose an application in to simple components that access and reuse a collection of platform services, thus simplifying programming and allowing a greater abstraction to the specific agent development.

We have developed and implemented [15, 40] a generic application architecture (shown in Fig. 6) where some general-purpose services may be provided by

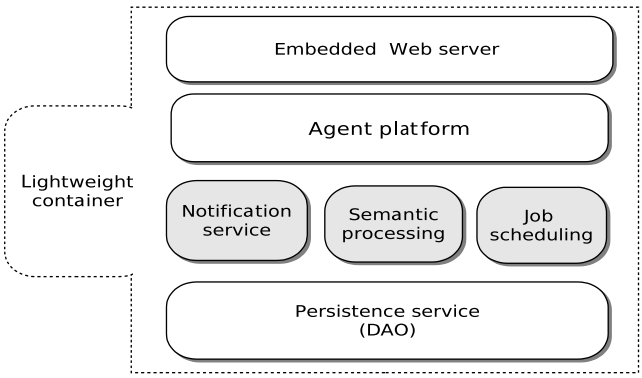

Fig. 6 Platform services framework (layered component view)

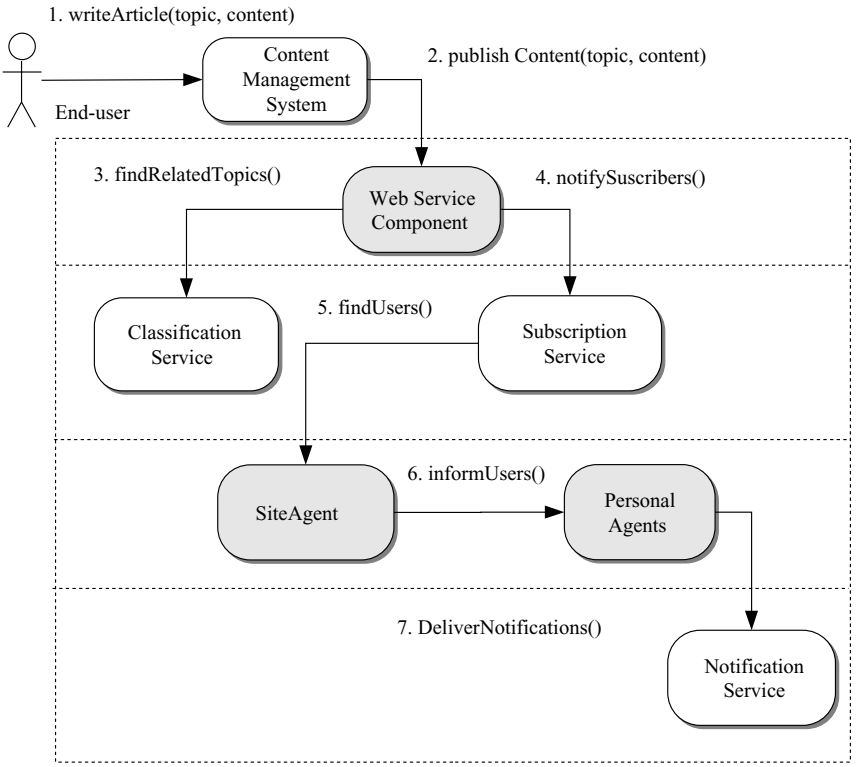

Fig. 7 Case study control flow

robust third-party tools. Each of the boxes in Fig. 6 represents a platform service that is configured and instantiated by the lightweight container; the container also fulfills the "service registry" role described earlier. Each platform service is composed of a particular kind of application component concerning a specific technical domain. The layer metaphor gives an insight about the abstraction level of each application tier, where high-level service components (i.e., Web-components and agents) consume the services of low-level components (i.e., data-access objects).

Using the service-oriented approach, the news delivery and classification case study causes the flow of events between services and agents as described in Fig. 7. The flow of events goes as follows:

- Once the user of the content management system (CMS) publishes a new article, it triggers a notification to the JITIK service. The CMS application will invoke the content-distribution service using a standard coarse-grained interface.
- After that, the classification service will attempt to match a set of topics for the article, according the contents of the ontology of the organization.
- When the content is classified, the request is passed into the agent-layer where agents determine the set of content receivers considering their interest profiles and personal preferences.

– Finally, relevant information will be delivered to users through the internal notification services, the system may leverage the use of several distribution media (Email, SMS) to alert users of new documents and other application events.

In the presented example, there are generic (Web, Agent, Persistence) and custom services (shaded) like the Scheduling and Notification services. For this specific case, services were defined and implemented as follows:

Embedded Web Server: Implemented using the Tomcat Web Server [27], it provides application with the capability of receiving HTTP requests and turning them to a particular Web-component or Servlet in conformance with the JSR-154 [44] specification.

Agent platform: The execution environment for the agents provided by the agent platform in conformity with FIPA [19] specifications. Particularly implemented with the JADE [4] platform.

Scheduling service: Allows the agents and users of the applications to schedule periodic or time-triggered execution of custom jobs using the Quartz Scheduler [45]. Job components are defined using an application-specific hierarchy of tasks and events (i.e., CheckMail, CheckAppointments, StartWorkflow, etc.).

Notification service: Allows the agents and Web-components to access the enterprise notification infrastructure, namely the mail service using the JavaMail API and several SMS systems. Notification components are the drivers for the distinct enterprise notification providers.

Semantic processing service: Provides access to a set of classification and inference tools like the Jess rule engine [20]. Allows agents to perform simplified queries to rule bases using native Java calls.

Persistence service: Implemented using the JDBC API and the components in conformance with DAO pattern, it is a low-level platform service that provides a simplified object access to enterprise datasources to the servlets, agents, jobs and other components in the application. It also offers access to directory services such as LDAP for user authentication and profile synchronization.

Following this approach, the particular agent-model of the system is designed and implemented on the Agent platform, enabling agents with the capability of invoking any of the underlying platform services. We believe that the use of such a framework becomes critical in agent projects where the agent developers do not share the same set of skills and concerns enterprise application developers have. In such cases, the use of service-oriented principles helps such heterogeneous development teams to achieve a clear division of work.

6 Discussion: related work

We contend that an agent-based approach to IK distribution could overcome the limitations of older systems based on older delegation-based systems, like push technologies. Indeed, according to ref. [18], the reasons why push technologies were not as successful as they were supposed to be were mainly (a) *excessive network load*; (b) *Excessive intrusion*; and (c) *Information overload*.

Concerning the network load, it is clear that if users are being constantly fed with information they actually do not consult, this implies a bandwidth waste. Of course, the critical aspect here is that the amount of received information outweighs by several orders of magnitude the actual information users need. The resulting information overload is indeed a major concern. The whole idea of push systems is to increase individual efficiency, and not hindering it with masses of useless information to consult. In that respect, we think that the *usefulness of information* is the one critical issue for the usability of push systems. So, the next question to solve is how to be *more selective* about the pushed information.

In the implementation project that inspired the case study we presented in Sect. 4, the enterprise representatives made clear to us that avoiding *information overflow* was a major concern. Reaching that goal in JITIK was possible mainly due to two aspects:

- In JITIK we "tailor" information to users using metadata of both of them in a two-tier way: first, the site agent makes a basic information filtering, limiting the information by means of ontology information. This reduces the network overload by avoiding indiscriminate information distribution. Then, personal agents take into account finer user preferences, both about the information itself as well as the ways users prefer to receive the information. This leads to a substantial reduction in the received useless information.
- At the site-agent level, the use of sophisticated inference mechanism, like the use of ontologies and argumentation for solving the IK distribution problem gives much greater flexibility and generality than traditional database approaches. As we have seen in Sect. 4, argumentation allows to nicely model decision making under conflicting policies. This can be efficiently automated in terms of DeLP.

To the best of our knowledge there are virtually no other efforts like JITIK for distributing knowledge and information on the basis of an argument-based approach. In ref. [35] the authors present a defeasible reasoning method for dealing with workflow processes, and it shares some goals with our work, as they also are able to deal with exceptions and imprecise information, but the application domain is quite different. A somehow related research is reported in ref. [34] about methods for helping in decision-making processes using argumentation. Besides the differences in the intended application of this system, there is also an important difference in the approach as they use static predefined argumentation schemas, whereas here we propose a general method for constructing arguments that is not restricted to a finite number of argument structures. Other works related to ours involve decision making and negotiation using argumentation among agents [37–39] In contrast, in our system, the argumentation process itself is not distributed, and it always takes place in the DeLP inference engine called by the Site Agent. Other argumentation-based decision-support proposals focus on the planning process for workgroup support, like the "dialectical planning" of Karacapilidis [30]. In none of the earlier-mentioned references, the problem of IK distribution is considered as is done in this paper.

7 Conclusion

In this paper we have presented the JITIK approach to model knowledge and information distribution, giving a high-level account of the research made around this project. We presented its basic architecture, as well as some additional technologies that gave it more flexibility, like the use of argumentation-based reasoning procedures and integration technologies for agents making use of Web Services.

We think that the JITIK effort can be integrated with a number of alternative architectures and technologies, supporting thus the claim that it is possible to build intelligent delegation-based systems to distribute knowledge and information in a flexible and efficient way. In particular, we have presented a novel argument-based approach for supporting IK distribution processes in large organizations by providing an integration of the JITIK multiagent platform with a defeasible argumentation formalism. The main advantage obtained through the use of an argumentation engine in JITIK is an increased flexibility, as it is not necessary to explicitly encode actions for every possible situation. This is particularly important in corporate environments with potentially conflicting IK distribution criteria.

The practical feasibility of our proposal is currently being validated through an implementation project of an information distribution center in one of the leading industries in Mexico. Although such project has involved integrating a large agent-based platform in a complex enterprise software environment, the results obtained so far have been successful and very promising. In the near future, we also intend to extend our approach for a distributed version of DeLP that could allow several JITIK Site Agents to perform collaborative decision making. Research in this direction is currently being pursued.

Acknowledgements This work was supported by the Monterrey Tech CAT-011 Research Chair, by Projects TIC2003-00950, TIN 2004-07933-C03-03, by Ramón y Cajal Program (MCyT, Spain), and by CONICET (Argentina).

References

1. Aguirre JL, Brena R, Cantu FJ (2001) Multiagent-based knowledge networks. Expert Syst Appl 20(1):65–75
2. Arsanjani A (2001) A domain-language approach to designing dynamic enterprise component-based architectures to support business services. In: 39th international conference and exhibition on technology of object-oriented languages and systems, August 2001. TOOLS 39, pp 130–141
3. Atkinson R, Court R, Ward J (1998) The knowledge economy: knowledge producers and knowledge users. The new economic index, http://www.webcitation.org/5Lok8rHAK
4. Bellifemine F et al (1999) JADE — A FIPA-compliant agent framework. In: Proceedings of PAAM99, London, pp 97–108
5. Berners-Lee T, Hendler J, Lassila O (2001) The semantic web. Sci Am 284(5):28–37
6. Borges B, Holley K, Arsanjani A. (2004) Service-oriented architecture. http://www.webcitation.org/5LoIY9AGX
7. Bradshaw JM (ed) (1997) Software agents. AAAI MIT Press, Menlo Park, CA
8. Brena R, Aguirre J (2004) Push technologies leveraged by intelligent agents. In: The 8th world multi-conference on systemics, cybernetics and informatics proceedings, Orlando, Florida, USA, pp 236–238

9. Brena R, Aguirre JL, Treviño AC (2001a) Just-in-time information and knowledge: agent technology for KM bussiness process. In: Proceedings of the 2001 IEEE conference on systems, man and cybernetics, IEEE press, Tucson, Arizona, USA, pp 3303–3308
10. Brena R, Aguirre JL, Treviño AC (2001b) Just-in-time knowledge flow for distributed organizations using agents technology'. In: Proceedings of the knowledge technologies 2001 conference, Austin, TX
11. Brena R, Ceballos H (2004a) Combining global and local ontology handling in a multiagent system. In: Barr V, Markov Z (eds) Proceedings of the seventeenth international Florida artificial intelligence research symposium conference, , AAAI press, Miami Beach, Florida, USA, pp 418–424
12. Brena R, Ceballos H (2004b) A hybrid local-global approach for handling ontologies in a multiagent system. In: Yager RR, Sgurev VS (eds) Proceedings of the 2004 informations systems international conference, Varna, Bulgaria, IEEE, pp 261–266
13. Carrillo J (1998) Managing knowledge-based value systems. J Knowl Manage 1(4):280–286
14. Ceballos H, Brena R (2004a) Finding compromises between local and global ontology querying in multiagent systems. In: Meersman R, Tari Z (eds) On the move to meaningful Internet systems 2004: CoopIS, DOA, and ODBASE: OTM confederated international conferences proceedings, LNCS, Springer-Verlag, Berlin, vol 3291, pp 999–1011
15. Ceballos H, Brena R (2004b) Web-enabling multiagent systems. In: Advances in artificial intelligence IBERAMIA 2004: 9th Ibero-American conference on AI, LNCS, Springer-Verlag, Berlin, vol 3315, pp 53–61
16. Ceballos H, Brena R (2005) Combining local and global access to ontologies in a multiagent system. J Adv Comput Intell Intell Inf 9(1):5–12
17. Chesñevar C, Maguitman A, Loui R (2000) Logical models of argument. ACM Comput Surv 32(4):337–383
18. Chin P (2003) Push technology: still relevant after all these years? Intranet J, http://www.webcitation.org.5LvGUSopy
19. Foundation for Intelligent Physical Agents (2002) FIPA abstract architecture specification, http://www.fipa.org/specs/fipa00001/SC00001L.html
20. Friedman-Hill E (2005) Jess, the rule engine for the Java platform. http://www.webcitation.org/5LvH1dQaJ
21. García A, Simari G (2004) Defeasible logic programming: an argumentative approach. Theory Pract Logic Program 4(1):95–138
22. Himelstein RSL (1999) PointCast: the rise and fall of an internet star. Bus Week Online, http://www.webcitation.org/5LvHCUwsi
23. Horibe F (1999) Managing knowledge workers. Wiley, New York
24. Horrocks I (2002) DAML+OIL: a description logic for the semantic Web. Bull IEEE Comput Soc Tech Comman Data Eng 25(1):4–9
25. Hunter A (2001) Hybrid argumentation systems for structured news reports. Knowl Eng Rev (16):295–329
26. Interface21 Limited (2005) The Spring Java/J2EE application framework, http://www.springframework.org/
27. Johnson R (2003) Introducind the spring framework. TheServerSide.COM
28. Johnson R (2002) Expert one-on-one J2EE design and development. Wrox, Birmingham, UK
29. Johnson R (2005) J2EE development frameworks. Computer 38(1):107–110
30. Karacapilidis N, Gordon T (1996) Dialectical planning: designing a mediating system for group decision making. Proceedings of the 10th Workshop Planen und Konfigurieren, pp 205–216
31. Labrou Y, Finin T, Peng Y (1999) Agent communication languages: the current landscape. IEEE Intell Syst 14(2):45–52
32. Liebowitz J, Beckman T (1998) Knowledge organizations. CRC Press, St. Lucie
33. Liebowitz J, Wilcox L (1997) Knowledge management. CRC, Boca Raton, FL
34. Lowrance JD, Harrison IW, Rodriguez AC (2000) Structured argumentation for analysis. In: Proceedings of the 12th international conference on systems research, informatics, and cybernetics, Baden-Baden, Germany, pp 47–57

35. Luo Z, Sheth A, Miller J, Kochut K (1998) Defeasible workflow, its computation and exception handling. In: Proceedings of the CSCW-98 workshop: towards adaptive workflow systems, Seattle, WA, http://www.webcitation.org/5LvIsPVZV
36. Negroponte N (1996) Being digital. Random House, New York
37. Parsons S, Jennings NR (1998) Argumentation and multi-agent decision making. In: Proceedings of the AAAI spring symposium on interactive and mixed-initiative decision making, Stanford, CA, pp 89–91
38. Parsons S, Sierrra C, Jennings N (1998) Agents that reason and negotiate by arguing. J Logic Comput 8:261–292
39. Rahwan I, Ramchurn SD, Jennings NR, Mcburney P, Parsons S, Sonenberg L (2003) Argumentation-based negotiation. Knowl Eng Rev 18(4):343–375
40. Ramirez E, Brena R (2005) 'Integrating agent technologies into enterprise systems using web services'. In: ICEIS 2005, proceedings of the seventh international conference on enterprise information systems, Miami, USA, pp 11–15
41. Schuschel H, Weske M (2004) Automated planning in a service-oriented architecture. In: 13th international IEEE workshops on enabling technologies: Infrastructure for collaborative enterprises, WET ICE, pp 75–78
42. Simari G, Loui R (1992) A mathematical treatment of defeasible reasoning and its implementation. Artif Intell 53:125–157
43. Stolzenburg F, García A, Chesñevar C, Simari G (2003) Computing generalized specificity. J Non-Class Logics 13(1):87–113
44. Sun Microsystems, Inc. (2003) JSR-000154 Java(TM) Servlet 2.4 Specification (Final release), http://jcp.org/aboutJava/communityprocess/final/jsr154/ index.html
45. The OpenSymphony Group (2005) Quartz enterpise job scheduler, http://www.opensymphony.com/quartz/
46. Whitestein Technologies, A.G. (2003) Web services agent integration project, http://wsai.sourceforge.net
47. Wooldridge M (1999) Multiagent systems: a modern approach to distributed artificial intelligence. MIT Press., Cambridge, MA
48. Wooldridge M (2001) An introduction to multiagent systems. Wiley, Baffins Lane,UK
49. Wooldridge M (1997) Agent-based software engineering. Softw Eng IEE Proc 144(1):26–37
50. World Wide Web Consortium (W3C) (2000) Extensible markup language (XML) 1.0, 2nd Edn, http://www.webcitation.org/5LvJRvkNn
51. World Wide Web Consortium (W3C) (2001) web services description language (WSDL) 1.1, http://www.w3c.org/TR/wsdl
52. World Wide Web Consortium (W3C) (2003a) Simple object access protocol, http://www.w3.org/TR/SOAP
53. World Wide Web Consortium (W3C) (2003b) Web services glossary, working draft, http://www.w3c.org/TR/ws-gloss/
54. Yang SJH, Lan BCW, Chung J-Y (2005) A new approach for context aware SOA. In: The 2005 IEEE international conference on e-Technology, e-Commerce and e-Service, EEE '05, Hong Kong, China, pp 438–443

Author biographies

Ramón F. Brena is Full Professor at the Center of Intelligent Systems, Tech of Monterrey, Mexico, since 1990, where he is Head of a Research Group in Distributed Knowledge and Multiagent Systems. Dr. Brena holds a Ph.D. from the INPG, Grenoble, France, where he presented a Doctoral Thesis related to knowledge in program synthesis. His current research and publication areas include intelligent agents and multiagent systems, knowledge processing and distribution, semantic web, and artificial Intelligence in general. Past research include program synthesis and software reuse, as well as automated reasoning. Dr Brena is a member of the SMIA (AI Mexican Society), the AAAI, the ACM, and is in the SNI, CONACyT.

José L. Aguirre is Associate Professor at the ITESM Campus Monterrey in the Center for Intelligent Systems since 1990. He received the bachelor degree in computer systems engineering from de ITESM Campus Quertaro in 1980 and another Bachelors degree in informatics engineering from the ENSIMAG in Grenoble, France in 1985. After that, he recieved a PH degree in Informatics with major in Artificial Intelligence from the INPG in Grenoble, France in 1989. He is now an Invited Researcher at the HELIX group of INRIA in Montbonnot, France.

Carlos I. Chesñevar is Researcher and Professor at the Universidad Nacional del Sur, Bahía Blanca, Argentina, and External Researcher at the Artificial Intelligence Research Institute (IIIA-CSIC, Bellaterra, Spain). He holds the degrees of Magister in computer science and Ph.D. in computer science (Universidad Nacional del Sur, Argentina). His recent research activities have been focused on the development of argument-based software applications in the context of different real-world problems. Part of his current research is also involved with the study of extensions of logic programming which combine vague knowledge and defeasible argumentation. His research interests cover defeasible argumentation, logic programming, intelligent systems, recommender systems, multiagent Systems, and semantic Web.

Eduardo H. Ramírez holds a M.Sc. degree in information technology from the Tech of Monterrey, Mexico, where he collaborated as a Research Assistant and Staff Engineer at the Center of Intelligent Systems. He is involved in the development and enterprise implementation of the JITIK Project. He is Cofounder and CTO of Ensitech, S.C. a high-tech startup with consultancy and research activities in distributed computing and Web technologies. His current Ph.D. research work and interests involve agent-oriented software engineering, Web services and service oriented architectures, semantic web, rich internet applications and collaborative knowledge management.

Leonardo Garrido is Professor and Researcher of the Center of Intelligent Systems at the Monterrey Institute of Technology (ITESM). He holds a M.S. in computer systems and a Ph.D. in artificial intelligence (with specialization in autonomous agents and multiagent systems). During his doctoral studies, he was a visiting Research Scholar in the Robotics Institute at Carnegie Mellon University (CMU) for 3 years, where he conducted research on multiagent meeting scheduling and learning models about other agents in multiagent environments. He has published in several international conferences specialized in Agents, multiagent systems and artificial intelligence. During the most recent years, he has served as Member of program and scientific committees of several conferences, such as the International Conference on Autonomous Agents and Multiagent Systems (AAMAS) and the Mexican International Conference on Artificial Intelligence (MICAI). He has also served as external reviewer for several top journals and conferences.