Knowledge Federation: Necessity and Required Technologies

Yuzuru Tanaka

Meme Media Laboratory, Hokkaido University
N13 W8, Sapporo 0608628, Japan
{tanaka}@meme.hokudai.ac.jp

Abstract. This paper focuses on knowledge federation from different aspects; its taxonomy, its necessity, and its required technologies. There are three different aspects of knowledge federation, namely, semantic federation, service federation, and view federation. Each knowledge federation uses at least one of these aspects. The latter half of this paper focuses on *ad hoc* knowledge federation of Web resources including Web applications. *Ad hoc* knowledge federation deals with two of the three aspects, i.e., service federation and view federation. This paper reviews the current author's R&D on meme media technologies and *ad hoc* knowledge federation based on them, and then proposes an extension of the Web toward 'the memetic Web,' in which users' collaborative reediting and redistribution of Web resources accelerates the evolution of extended Web resources. Finally, it introduces a new version of the meme media system IntelligentPad as an enabling technology for the memetic Web. This new version runs on *de facto* standard browsers empowered by Silverlight plug-in, and requires no IntelligentPad kernel running on clients.

1 Introduction

The Internet has been and still is continuously changing our lives. It is pervading the economy and the private sphere, thus boosting the pace at which modern information and communication technologies are penetrating everyone's life, every science and engineering discipline, and almost every business. Despite the fact that billions of Web resources are available world-wide and no slowdown of the Web's growth is in sight, users typically make little use of the richness of the Internet.

One well-known problem is the difficulty of finding what a user is interested in, due to a lack of a suitable schema and a query language to store and retrieve resources. A second problem is the difficulty of making mutually related Web resources work together, due to a lack of suitable interfaces or conversely a plethora of interfaces, formats, and standards. Here in this paper, let me call the first problem 'the findability problem,' and the second 'the reusability problem.'

There is a similar prominent problem in practically every large enterprise, in the larger healthcare institutions and in literally every governmental agency: enterprise application integration. The key problem is that legacy systems have

been introduced over many decades. Each department has no means to know what kinds of applications are used and how they are related with each other in another department. Most of these systems have never been designed to work together, and the potential for synergetic effects dwindles away. Both the findability and reusability problems exist there. Such situations can be commonly observed even in those enterprises which, in 70s and 80s, made large investments to fully integrate their data and applications by introducing database management systems. Such paradoxical situations were brought by rapid and *ad hoc* introduction of many different types of client applications, and also by the needs for the interoperation with Web resources.

The diversity, heterogeneity, autonomy, and openness of both resources and their usages are the inherent characteristics of current information system environments. These characteristics are inherently out-of-control, and incompatible with conventional top-down system integration methods.

It seems that human creativity will always lead to newly designed systems and solutions that go beyond earlier standards and overall pictures of whole systems, and are thus not fully compatible with the established and widely used systems. Creating and updating standards in information and communication technologies is of course needed, but there is little hope that this alone could ever solve the enterprise application integration problem, or alleviate the abovementioned problems that face individuals, enterprises and communities when dealing with the diversity of the Web.

The 'findability problem' is partially solved either by providing indices for useful resources, or by semantically restructuring resource fragments. While the first approach has developed Web search engines using resource indices, the second approach has introduced portal sites, Semantic Web technologies [1], Topic Map tools [2], Web ontologies [3], ontology mediation technologies [4], and semantic inference technologies [5]. The semantic restructuring is further classified into two different views of semantic entities, i.e., as universals and as names. The first view assumes the existence of a single universal semantic structure among entities. This first view worked well for such compiled sets of resources as databases, but not for such an ever-evolving open set of resources like the Web. The latter requires semantic restructuring based on the second view, which may independently restructure mutually overlapping resource and resource-fragment sets in different semantic schemes, and later use ontology mediators to consistently merge these semantic schemes into a virtually integrated scheme.

The 'reusability problem' addresses how to extract some fragments you want to reuse from the resources we find, how to customize each of them for fitting it to its reuse context, and how to coordinate them work together to perform a required job. Here the coordination means a huge variety of different goals, which include the construction of a single portal Web page showing these fragments, the application of some of them as functions to the others as data sets, the combination of their functions to compose a new function, and the composition of a new visual Web resource embedding the visual representations of these fragments.

Semantic restructuring of resources and resource fragments, and functional and/or visual combination of resource fragments give a partial solution respectively to the findability problem and to the reusability problem. Both of these topics share the same characteristics; the diversity, heterogeneity, autonomy, network-distribution, and openness of resources and their usages, the virtual or real extraction of useful resource fragments into structured formats, and the definition of interrelation and/or interoperation among them to treat them as a single virtual complex information and/or functional entity. We use the word 'knowledge federation' to denote such a single virtual entity defined from Web resources, or such a defining action. The word federation emphasizes the goal to form a virtual unity without restricting any of the diversity, heterogeneity, autonomy, network-distribution, and openness of both resources and their usages.

This paper focuses on knowledge federation from different aspects; its taxonomy, its necessity, and its required technologies. First, it points out that there are three different factors of knowledge federation, namely, semantic federation, service federation, and view federation. Then it focuses on *ad hoc* knowledge federation of Web resources including Web applications. *Ad hoc* knowledge federation deals with two of the three factors, i.e., service federation and view federation. It reviews the current author's R&D on *ad hoc* knowledge federation, and proposes an extension of the Web toward 'the memetic Web,' in which users' collaborative reediting and redistribution of Web resources accelerates the evolution of extended Web resources. Finally, this paper introduces our new version of the IntelligentPad system as an enabling technology for the memetic Web. This new version runs on *de facto* standard browsers empowered by Silverlight plug-in, and requires no IntelligentPad kernel running on clients.

2 Taxonomy of Knowledge Federation

The word 'knowledge' in computer science was initially used in a very limited meaning in 70s and 80s to denote what can be described in some 'knowledge representation language' for the problem solving by machines. The same word in dairy conversation denotes facts, information and skills acquired through experience or education, theoretical or practical understanding of a subject, language, etc., or awareness or familiarity gained by experience of a fact or situation. Today, we use computers to deal with all these different types of knowledge by representing them as some kinds of objects such as documents, tools and services. IO relations and/or functions of applications and services can be also considered as knowledge fragments.

The Web is expanding its capability to deal with all these different types of knowledge for their sharing in a society and/or in an enterprise, and to promote their reuse in different contexts for different purposes. Recently, some philosophers such as Davis Baird are trying to expand the definition of knowledge to denote not only linguistically expressible knowledge, but also 'thing knowledge', i.e., knowledge embodied as some artifact object [6]. Computerization of the design and manufacturing of artifacts is also accumulating their thing knowl-

edge as reusable and sharable knowledge. Only some Web documents satisfy the definition of linguistically expressible knowledge, while all the Web resources including documents, tools, and services are artifacts, and satisfy the definition of thing knowledge. Our definition of 'knowledge' in this paper includes both linguistically expressible knowledge and thing knowledge.

Knowledge resources denote sharable and reusable different types of such knowledge represented on computers. The Web basically represents knowledge resources as compound documents with embedded nonfunctional multimedia contents and/or functional contents such as application tools and services. The compound document architecture was initially proposed around mid 1980s as an extension of multimedia documents. While a multimedia document embeds in itself multimedia contents, a compound document looks like a multimedia document and can embed in itself not only multimedia contents but also any interactive visual objects with some functionality. Web documents as well as Web applications are compound documents. Compound documents treated in the same system are based on the same compound document architecture.

Knowledge resources require some media to externalize them as interactive visual objects, and to publish them for their reuse by other people. Knowledge media denote such media since they externalize knowledge resources [7]. Knowledge resources treated in the same system exploit the same knowledge media architecture. Compound document architectures are examples of knowledge media architectures. While compound documents are two dimensional visual representations of knowledge resources, some knowledge media architectures represent knowledge resources as three dimensional visual objects or spaces.

The Web provides both a standard knowledge media architecture to publish knowledge resources as Web documents, and an open repository of Web documents for publishing, sharing, and reusing them among different people distributed over the world. Web documents are compound documents.

The word 'federation' was probably first introduced to IT areas in mid 80s by Dennis Heimbigner in the context of a federated database architecture [8], and then secondarily in late 90s, by Bill Joy in a different context, namely, federation of services [9]. In a federated database architecture, a collection of independent database systems are united into a loosely coupled federation in order to share and exchange information. The term federation in this context refers to the collection of constituent databases participating in a federated database. In the context of service federation, a federation is a set of services that can work together to perform a task.

In a federated database, a federation consists of components and a single federal dictionary [8]. The components represent different databases, individual users, applications, workstations, or other components in an office information system. The federal dictionary is a specialized component that maintains the topology of the federation and oversees the entry of new components. Each component in the federation controls its interactions with other components by means of an export schema and an import schema. The export schema specifies the information that a component will share with other components, while the

import schema specifies the nonlocal information that a component wishes to manipulate. The federated architecture provides mechanisms for sharing data, for sharing transactions via message types to combine information from several components, and for coordinating activities among autonomous components via negotiation.

The idea of federating databases can be extended to federation of Web information resources, which we call 'semantic federation.' Each constituent database schema corresponds to the articulation of useful information fragments in some set of Web information resources, and the semantic structuring of these articulated fragments. Semantic Web and Topic Map technologies are used for such articulation and semantic structuring. Instead of using export and import schema to consistently merge different but partially overlapping database schemata, federation of Web information resources combines more than one semantic structure over sets of Web information resources to provide a federated single semantic view of the relationships among those resource fragments.

Federation of services, on the other hand, denotes the definition and execution of interoperation among services that are accessible either through the Internet or through peer-to-peer *ad hoc* communication. It may be classified into two types: *ad hoc* federation defined by users and autonomic federation defined by programs. *Ad hoc* service federation denotes a technological goal that enables users to define a service federation without writing a code, while autonomic federation denotes a service federation in which each service-requesting service autonomously finds a corresponding service-providing service to form a federation. Most of the service federation architectures proposed in the past focus on autonomic federation [10–13]. Only a few of them focus on *ad hoc* federation of services [14, 15].

Autonomic service federation architectures basically propose two things, i.e., a standard communication protocol with a language to use it, and a repository-and-lookup service that allows each member to register its service-providing capabilities, and to request a service that matches its demand. For each request with a specified demand, a repository-and-lookup service searches all the registered service capabilities for those satisfying the specified demand. A repository-and-lookup service matches service-requesting queries with corresponding service-providing capabilities. The origin of such an idea can be found in the original tuple space model Linda [10] and its extension Lime [11] that copes with mobile objects by providing each of them a dedicated tuple space. Linda and Lime are languages that use tuples to register service-providing capabilities and to issue service-requesting queries to a repository-and-lookup service called a tuple space. Java Space [12] and Jini [13] are Java versions of Linda and Lime architectures.

The idea of federating services can be also extended to federation of knowledge resources over the Web. Knowledge resources include not only Web services but also Web applications and Web documents. Federation of knowledge resources first needs to extract useful fragments from Web resources as services with or without visual presentations. Especially in case of a Web application without its API, it needs to extract an IO relation between some input forms

and some output-page fragments as a reusable service [14, 15]. From a Web document, it may extract a structured set of data [16]. Then it needs to combine these data sets and services so that they may work together virtually as a single application with or without a composite visual presentation. Federation of Web resources with extraction and composition of visual presentations needs to define not only federation of services but also federation of their presentation views into a single composite presentation view. We call the latter 'view federation.' We may also consider a view federation of Web applications and/or Web documents without their service federation. For example, some tools help you extract portlets from different Web resources and to combine them to construct a portal site without defining any interoperation among these portlets [17].

Knowledge federation is characterized by the following three different aspects;

1. semantic federation,
2. service federation, and
3. view federation.

Each knowledge federation uses at least one of these aspects.

Federation is becoming fundamental to IT. It is the process of combining multiple technology elements and/or heterogeneous resources into a single virtual entity. Federation is being driven by complexity, and by IT's need to make sense of technology systems that have sprawled to unprecedented scale and have become extremely expensive to maintain and manage. Federation is different from integration in the following sense. Federation assumes an open networked environment of heterogeneous, autonomous, and distributed resources, and deals with open scenarios of information processing. Integration basically targets local and centralized management and interoperation of resources in a closed environment, and deals with closed scenarios of information processing.

3 R&D Trends in Knowledge Federation

R&D on semantic federation uses either Semantic Web or Topic Maps as their basis. As to the extraction of useful Web information in a structured format, one remarkable effort can be found in DBpedia [18], which is a community effort to extract structured information from Wikipedia and to make this information available on the Web. DBpedia provides us with a semantic schema in which extracted information resources are structured, and allows us to access these resources through this schema by writing a query in a query language SPAQL. Soon in a future, we will have lots of such semantically structured Web information extracted from various different sets of Web resources. We will need a semantic federation of them to merge their schemata into a single unified schema so that we may issue a query through this unified schema.

R&D on service federation of Web services has three mutually overlapping technology trends. The first trend uses a tuple space or an XML space to federate Web services [19]. The second trend uses a GRID computing architecture and its workflow/orchestration tools to federate Web services [20, 21]. The third trend

uses Semantic Web description for semantically federating Web services [22]. This technology is called Semantic Web Service. Semantic Web Service has two aspects of knowledge federation; service federation and semantic federation.

R&D on service federation of Web resources including Web applications has two technology trends. One of them is represented by Web mashup tools. The other is the current author's knowledge federation framework [14, 15] based on meme media technologies [23]. Since service federation of Web resources including Web applications needs to deal with page presentations of Web applications, it requires user's intervention for specifying which portions to extract and how to arrange them to compose a target Web page. Service federation of Web resources including Web applications also requires view federation.

Web mashup denotes the user-driven micro-integration of Web data, where micro integration means data merging, data feeding, data joining, data filtering, and data annotating of Web data. Web mashup tools range from the feeding of Web service output data to the API of a specified Web application such as Google Maps [24] and Google Chart, to advanced mashup editors such as Intel Mash Maker [25] and Lotus Mashup. Web mashup tools in the first generation used some specific Web application such as Google Maps as the base application, and used its API to add, on its page, some graphical objects that represent the data retrieved from other Web services. The server program of this base Web application needs to provide such a function through its API. The addition of graphical objects corresponding to the external data is performed by the server program of this Web application. Traditional mashup tools are based on workflow composition of Web services to construct a new Web application or service. The I/O data transfer in such a composition depends on SOAP, REST and/or RSS/ATOM feeds. To include a Web application, users must derive a proxy Web service by using some extraction tool. These mashup tools cannot deal with Web applications as source resources.

Recently, technologies are moving towards on-screen visual composition of new resources, including widgets extracted from Web applications. Mash Maker supports augmentation of Web pages with data from other sources. Users can define an extractor for the data within an HTML page, and paste registered widgets onto such pages, connecting them through the relevant extractor. The widgets are portlets encapsulating Web-application behavior, created by writing code. Lotus Mashup supports composition within a Web-based workspace. Tools allow users to create widgets that capture Web application behavior. Widgets are combined in the workspace. There is a good paper comparing two cultures; mashing up Web 2.0 and the Semantic Web [26].

In the sequel of this paper, we will focus on *ad hoc* knowledge federation of Web resources including Web applications. *Ad hoc* knowledge federation deals with both service federation and view federation. This type of knowledge federation is the one the current author proposed its concepts and architectures in 2004 [27] based on our R&D on 2D and 3D meme media since 1987 [23].

4 Ad hoc Knowledge Federation of Web Resources Including Web Applications

4.1 Its requirements

The Web is becoming an open repository of a huge variety of knowledge resources. The problem we face today is how to make it easy to select some set of resources, customize them, and make them work together to meet our ever-changing demands for new knowledge resources. Commercially available drug design software systems are a good example of the complex integrated application systems available today. Each system provides an integrated environment comprising modules such as a protein database, a homology search system, a ligand structure prediction system, and a docking simulator. While such integrated systems are very expensive, for each module you can often find a free Web service, or a downloadable freeware program. Similar situations are frequently observed in a range of academic areas. Nonetheless, finding appropriate data sources and applications from the Web, customizing them, and making them work together is as difficult and expensive as developing a commercial product with similar complexity. If there were an easy, generic way for users to edit these open resources instantaneously to compose the desired complex applications, we would be able to accelerate substantially the evolution of the knowledge resources shared by our societies. The existing proposals for federation, such as Jini, provide no support for instantaneous extraction, customization, or composition of knowledge resources.

Recently a new research methodology has been developed for the advanced automation of scientific data acquisition tools, based on collecting data for all cases without any specific purposes, then later retrieving and analyzing appropriate data sets to support a specific hypothesis or conjecture. This new methodology is referred to as 'data-based science', or 'data-centric science'. Data-based science is becoming more and more popular in scientific areas such as genomics, proteomics, brain science, clinical trials, nuclear physics, astrophysics, material science, meteorology, and seismology. Data-based science makes data acquisition independent from data analysis; it leads to large numbers of huge, independent accumulations of data, and a great variety of data analysis tools. Many such data accumulations and analysis tools have been made available over the Web, to encourage their reuse. Advanced utilization of such resources requires both flexible extraction of appropriate data sets from these diverse sources, and flexible application of appropriate analysis tools.

Such an approach, although currently performed with lots of manual operations and programming, has enabled the comparative analysis of seemingly unrelated large data sets, leading to the discovery of interesting correlations such as those between the El Niño-Southern Oscillation and the weather in Europe, volcanic eruptions, and diseases. These studies required extraction of appropriate data sets from multiple seemingly unrelated sources, their compilation or customization, and the application of data analysis and/or data visualization tool resources to these data sets. Such operations are inherently performed in an

ad hoc manner, and should be executable rapidly enough to follow up on fleeting ideas.

The creation of information systems that may work as a science infrastructure to accelerate further development of science and technology requires a new generic technology for 'knowledge federation', as well as the development and extension of databases, simulators, and analysis tools in each research field. For the routine jobs that require interoperation of multiple knowledge resources, we may use work-flow and/or resource-orchestration technologies based on GRID computing. These technologies were extensively studied and developed during the last decade in the US, Europe and Japan, and are now becoming off-the-shelf technologies. But in R&D activities, where repetitive trial-and-error operations are a vital part of creative thinking, we need support for *ad hoc*, instantaneous implementation of passing ideas. By contrast, most of the preceding e-science projects were based on GRID computing technologies that are suited only to routine work.

Ad hoc knowledge federation is required in strategic intellectual activities including not only scientific research activities, but also strategic analysis and planning such as financial analysis and planning, and disaster contingency analysis and planning.

Recently, the Web works not only a world-wide repository of documents, but also an infrastructure to make mobile phones, sensor/actuator networks, home electronic appliances, and expensive scientific equipments accessible to and from Web resources. Users can access and/or control these objects through the Web from anywhere, and these objects can get useful information from anywhere through the Web, and/or update some Web resources. Figure 1 shows the current situation of IT environments. The Web provides a basic network of knowledge resources. Mobile phones are connected to some Web servers. A mobile phone can access any Web resources via such a server. It can be also accessed from some Web resources via such a server. A sensor/actuator network with ZigBee protocol can dynamically span a network of a coordinator and sensor/actuator nodes. The coordinator can read the value of any sensor, and/or write the value of any actuator, in any node. By defining a Web application or a Web service for the computer with this coordinator node, we can make the value of each sensor of every node accessible through this Web application or Web service. In each house, home electronic appliances are connected to a home network, which is connected to the Internet through a network node. Therefore, these appliances are accessible to and from the Web for their monitoring and control. Figure1 also shows *ad hoc* peer-to-peer networks without any connection to the Web. Nodes in these networks cannot access nor be accessed by any Web resources.

All different types of functions that are provided by mobile devices, sensor/actuator nodes, and electronic appliances are accessible through the Web as Web resources, namely as Web applications or Web services. Therefore, we may consider them as knowledge resources. We can apply our knowledge federation technology to extract some of their sub functions, to make them interoperate

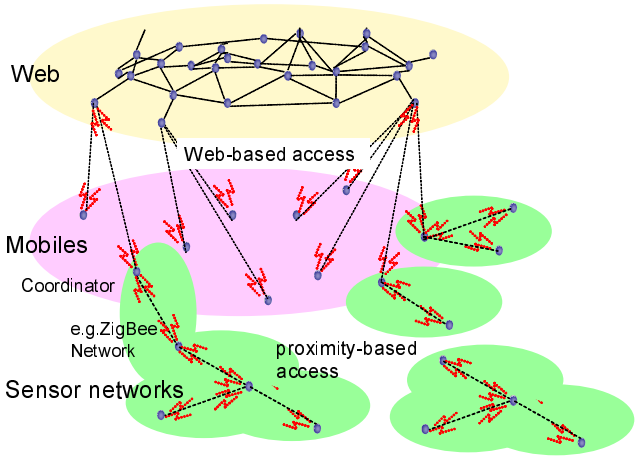


Fig. 1. Current IT environments consisting of the Web, mobile networks, and sensor networks.

with each other, and to compose a new knowledge resource as a Web application or a Web service.

4.2 Meme media as enabling technologies

We have been conducting the research and development of 'meme media' and 'meme market' architectures since 1987. The word 'meme' was coined by Richard Dawkins in 1976 [28]. He pointed out the similarity between biological evolution and cultural evolution. Ideas are replicated and propagated from people to people, which corresponds to genetic replication. Two different ideas are recombined to produce a new idea, which corresponds to genetic recombination. An idea may be propagated from a person to another with some error, and the resultant idea may become a new useful idea, which corresponds to genetic mutation. Finally, ideas are evaluated by a community, and only those that are frequently referred to remain as useful ideas. Dawkins coined the word 'memes' by combining two words; gene and mimesis. A meme denotes a virtual entity that carries a unit of knowledge, as a gene works as a unit of carrying genetic information.

A meme media architecture denotes a component-based knowledge media architecture in which users can visually combine meme media objects to compose a new meme media object, and decompose a composite one to reuse its component meme media objects for different compositions. It enables users to replicate and reedit knowledge resources on meme media, and to distribute them among people. If a world-wide repository of meme media objects is provided, knowledge resources that are published into this repository as meme media objects are replicated and recombined by a huge number of people through direct manipulations, and naturally selected by user communities. Such a repository works as a meme

pool. A meme pool promotes the evolution of knowledge resources published as meme media objects.

The current Web works as a publishing repository of knowledge resources, but does not work as a meme pool since users cannot easily extract fragments of Web resources and recombine them to compose a new Web resource only through direct manipulation of Web resources without writing any codes. Some Web mashup tools could be extended to share the same goal with meme media architectures, but they are still premature for the above mentioned purpose.

We developed the first versions of 2D and 3D meme media architectures 'IntelligentPad' [29] and 'IntelligentBox' [30] respectively in 1989 and in 1995, and have been working on their meme-pool and meme-market architectures, as well as on their applications and revisions. IntelligentPad represents each component as a pad, a sheet of paper on the screen. A pad can be pasted on another pad to define both a physical containment relationship and a functional linkage between them (Figure 2). When a pad P2 is pasted on another pad P1, the pad P2 becomes a child of P1, and P1 becomes the parent of P2. No pad may have more than one parent pad. Pads can be pasted together to define various multimedia documents and application tools. Unless otherwise specified, composite pads are always decomposable and reeditable.

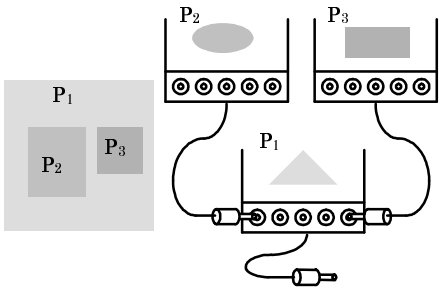


Fig. 2. A composite pad and slot connections.

In object-oriented component architectures, all types of knowledge fragments are defined as objects. IntelligentPad exploits both an object-oriented component architecture and a wrapper architecture. Instead of directly dealing with component objects, IntelligentPad wraps each object with a standard pad wrapper and treats it as a pad (Figure 2). Each pad has both a standard user interface and a standard connection interface. The user interface of a pad has a card like view on the screen and a standard set of operations like 'move', 'resize', 'copy', 'paste', and 'peel'. Users can easily replicate any pad, paste a pad onto another, and peel a pad off a composite pad. Pads are decomposable persistent objects. You can easily decompose any composite pad by simply peeling off the primitive or composite pad from its parent pad. As its connection interface, each pad provides a list of slots that work in a similar way as connection jacks of an AV-

system component, and a single connection to a slot of its parent pad (Figure 2). Each pad uses a standard set of messages 'set' and 'gimme' to access a single slot of its parent pad, and another standard message 'update' to propagate its state change to its child pads. In their default definitions, a 'set' message sends its parameter value to its recipient slot, while a 'gimme' message requests a value from its recipient slot. Figure 3 shows a set of pulleys and springs that are all represented as pads, and a composition with them. Each of these pulleys and springs is animated by a transparent pad. The old version of IntelligentPad could not directly deal with a graphical object animated on a canvas as a pad. Each pad was required to have a rectangular canvas. This restriction has been recently removed in the 2008 version of IntelligentPad that has employed SVG and runs on advanced *de facto* standard browsers empowered by Spaceflight plug-in.

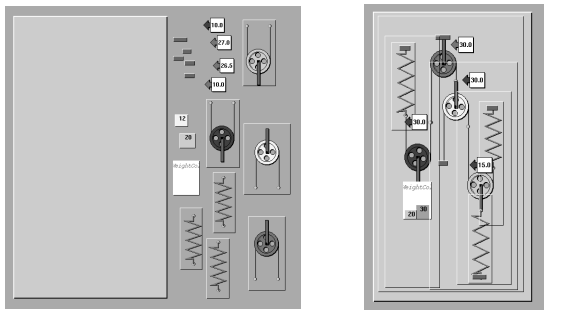


Fig. 3. An example pad composition.

4.3 Wrapping Web resources with 2D meme media for knowledge federation

From 2003 to 2004, we developed new technologies for users to easily wrap Web applications and Web services into meme media objects [14, 15, 31]. C3W [15] is a wrapper framework that enables users to open a browser showing a Web application page, to clip out some input forms and some output portions as pads, and to make such pads, that are clipped out from more than one Web page, interoperate with each other. C3W enables users to make these definitions only through direct manipulation without writing any program codes. For a given Web service with a WSDL description of its interface, our Web service wrapper enables users to define its proxy object as a pad with a specified subset of IO signals. Our Web service wrapper first pops up the IO signal list of the Web service for users to arbitrarily select some of them as the slots of the pad to define, and to specify default values for some other input signals. Users can

make these specifications only through direct manipulation. Then the wrapper automatically creates a pad that wraps the original Web service. This pad works as a proxy object of this Web service.

Figure 4 shows, at its bottom right corner, a Web application by US Naval Observatory showing day and night over the earth. The left object is a composite pad showing the difference of arbitrarily chosen two seasons; the same time on summer solstice and winter solstice for example. This was constructed by just clipping out the date input form and the simulated result as pads from the Naval Observatory Web application, drag-and-dropping them on a special pad, and applying the multiplexing to the input to obtain multiple outputs.

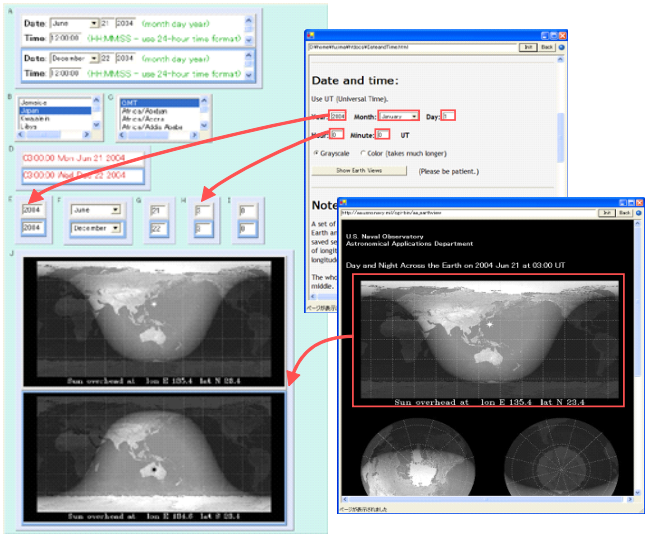


Fig. 4. Construction of a composite pad, using clips extracted from Web pages as pads.

All these operations were performed through direct manipulation of pads. For browsing the original Web page, we use a Web browser pad, which dynamically frames different extractable document portions for different mouse locations so that its user may move the mouse cursor around to see every extractable document portion. When it frames a desired object, you can just drag the mouse to clip out this object as a pad. All the pads thus clipped out from Web pages in a single navigation process keep their original IO functional relationship even after their arrangement on the same special pad C3WSheet. The C3WSheet pad gives a cell name to each pad pasted on it. The pads clipped out from date input forms are named as E, F, G, H, I cells. They respectively correspond to year, month, day, hour, and minute input forms. The simulation output pad is named as J cell. Whenever you input a new date to the extracted date input forms, the corresponding extracted output pad showing a simulated result will change its

display. This simulation is performed by the server corresponding to the source Web application from which all these pads are extracted.

The multiplexer pad, when inserted between the base C3WSheet pad and the extracted date input form pad, automatically inserts another multiplexer pad between the base C3WSheet pad and every extracted pad whose value may depend on the input to this date input form. If you make a copy of the extracted date input form pad on the multiplexer pad, each copy of this multiplexer pad automatically makes a copy of its child pad that was also extracted from the same Web application and is dependent on the update of the date value. Mutually related multiplexer pads maintain the relationship among the copies of their child pads. The original copies, the second copies, and the third copies respectively form independent tuples of extracted pads, and each tuple maintains input and output relationship among its constituent pads. In Figure 4, two simulation results are obtained for two different days.

The above mentioned composite pad allows us to specify any time in GST. You can easily modify this content to accept any local time by combining this with another public Web application for time conversion. The C3WSheet pad in Figure 4 has four more multiplexed pads named as A, B. C. and D cells. They are extracted from the time conversion service. The pads named as A, B, C cells are the extracted input forms, respectively specifying the input local time, the source time zone, and the destination time zone. The pad named as D shows the local time in the destination time zone. The destination time zone is fixed to GST in the cell C. You can just specify equations in the cells E, F, G, H, and I to make their values always equal to the year, month, day, hour, and minute of the cell value of D. Such a direct manipulation process customizes existing Web resources, and makes some of them interoperate with each other to define a new knowledge resource as a composite pad.

Figure 5 shows our basic idea of wrapping different types of knowledge resources into meme media objects for achieving interoperability among them. The arrows (1) and (2) respectively show the wrapping of Web applications and Web services into meme media objects. The arrow (6) denotes the combination of more than one meme media object by direct manipulation to compose a new composite meme media object. The arrow (5) denotes the wrapping of legacy applications into meme media objects. Legacy applications without GUI can be always easily wrapped into meme media objects if they expose their APIs. Legacy applications with GUI are not always easily wrapped into meme media objects even if they expose their APIs. The arrow (3) denotes that any composite pad can be accessed as a Web service through SOAP protocol. We have already developed a tool that automatically creates a corresponding Web service for any given composite meme media object. The arrow (4) denotes the conversion of an arbitrary composite meme media object into a Web application. This conversion is called 'flattening' since it converts a composite media object into a flat Web page viewable with any Web browser. Since composite meme media objects are not HTML documents, they are not directly viewable with Web browsers. We have already developed a tool for the automatic flattening of composite meme

media objects. This mechanism however makes it easy to develop phishing Web sites just by pasting a password stealing pad over a wrapped trustworthy Web application to define a composite pad. Because of the inherent fragility of the current Web technology against phishing trials, we have made a decision that we should not provide this flattening conversion mechanism to the public.

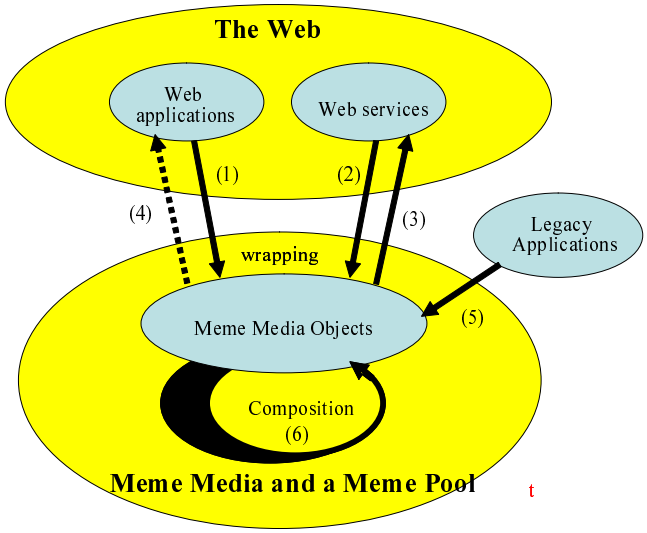


Fig. 5. A basic architecture to wrap different types of knowledge resources into meme media objects for their mutual interoperability.

While the flattening mechanism enables us to use the Web itself as a meme pool, we developed another world-wide repository Piazza that works as a meme pool for pads. Piazza was developed on top of Wiki, and uses Wiki servers. Piazza provides its own browser to access different pages. Each page that can be identified by a URL may contain arbitrary number of pads. Users can open an arbitrary Piazza page, and drag and drop pads between this page and a local pad environment. By clicking the page registration button, users can reflect the page update to the servers. By clicking the page refresh button, users can reflect the current state of the server to the displayed Piazza page. We have already shown that Piazza can be easily developed by applying C3W framework to Wiki system [15]. Figure 6 shows the Piazza browser displaying a Piazza page with several registered pads.

In order to make a meme pool Piazza system from Wiki, you can first access a Wiki page, and clip out the URI input, the HTML input form, the refresh button, and the output page as pads, and paste them on the same C3WSheet pad. You need to paste a PadSaverLoaderPad as a cell of the same C3WSheet pad, and relate its input and output respectively to the extracted input form pad and

Fig. 6. A worldwide repository of pads developed by applying C3W framework to Wiki.

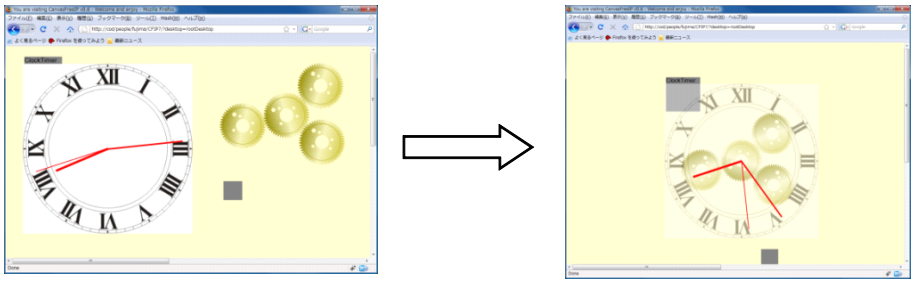
the extracted output page. A PadSaverLoaderPad makes conversion between a pad on itself and its save format representation in XML. Suppose that the PadSaverLoaderPad, the extracted HTML input form pad, and the extracted output page pad are assigned to cells A, B, and C. The relationship among them is defined as follows. We define the equation for the cell A as ←C, and the equation for the cell B as ←A. People can access any page specifying its URI, drag-and-drop arbitrary composite pads to and from the PadSaverLoaderPad of the composed pad to upload and download them to and from the corresponding Wiki server. Each page is shown by the PadSaverLoaderPad. For a jump from a page to another page in a Piazza system, we can use an anchor pad that can be pasted on a Piazza page. This anchor pad holds a URI that can be set through its #refURI slot, and, when it is clicked, sets this URI to the #URI slot of the Piazza, i.e., to the #URI slot of its base C3WSheet pad.

Piazza allows people not only to publish and share knowledge resources represented as pads, but also to compose new knowledge resources by combining components of those pads already published in it. People can publish such newly composed knowledge resources into the same Piazza system. The collaborative reediting and redistribution of knowledge resources in a shared publishing repository by a community or a society of people will accelerate the memetic evolution of knowledge resources in this repository, and make it work as a meme pool.

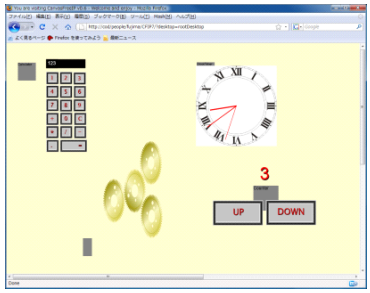
4.4 Knowledge federation and the memetic Web

The most recent 2008 version of IntelligentPad fully exploits Microsoft Silverlight technology, and runs on Internet Explorer 6 and 7, Mozilla Firefox, or Safari browser empowered by Silverlight plug-in. It is a Web-top system that allows us to directly manipulate pads on a Web page.

It is one important feature of this version that its pad may not have its rectangular canvas. Any graphical object including a line segment can be treated as a pad (Figure 7). Three needles of the analog clock in this figure are all pads. Such a needle does not need a transparent canvas on which it is animated. Such a pad is called a canvas-free pad. Our preceding versions could not deal with canvas-free pads. This new version exploits SVG (Scalable Vector Graphics) to describe graphical objects. When a pad is pasted on a canvas-free pad to work as its child pad, the former is bound by the local coordinate system spanned by the latter pad, instead of being clipped by the canvas area of the parent pad. This version also provides various primitive pads for the linear transformation of local coordinate systems. Figure 7 (b) shows an application of an affine transformation to the same gear complex in Figure 7 (a).



(a) a composition using an analog clock and a gear complex, with the dial plate being made translucent.



(b) an affine transformation applied to a gear complex

Fig. 7. The canvas-free IntelligentPad, and some of its example pads.

It is another important feature of this new version that we need no IntelligentPad kernel running on clients. Any advanced *de facto* standard browser empowered by Silverlight plug-in can provide a Web-top environment of pads, and render any composite pad as an extended Web page. Users accessing through such browsers can directly manipulate pads on a Web page, and do not recog-

nize any difference between Web resources and composite pads. Composite pads can be also registered in HTTP servers for other people to access them as Web documents.

The 2008 version of IntelligentPad, together with an advanced *de facto* standard Web browser empowered by Silverlight plug-in, extends the Web to 'the memetic Web' as shown in Figure 8. The memetic Web allows people to publish not only Web resources, but also composite pads. People can browse both of them just by using a browser. Client PCs need not be installed with an IntelligentPad kernel system. Users can extract resource fragments as pads from Web resources through direct operation by using C3W technology. They can also access a composite pad, and make copies of any of its component pads. They can combine these pads extracted from the memetic Web, as well as locally available pads, to compose a new composite pad for their reuse, and publish it into the memetic Web for the further reuse by other users. The memetic Web works as a meme pool of both Web resources and composite pads.

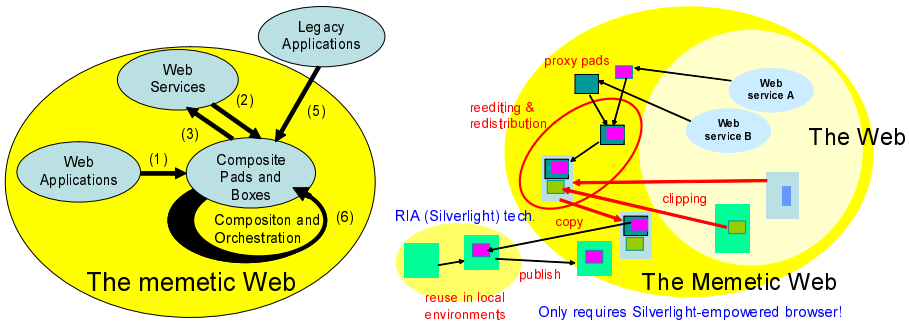


Fig. 8. The memetic Web and different types of knowledge resources.

The handling of more than one pad extracted from different Web applications on the same web top requires so-called cross-frame scripting, which is not allowed by any browser. To solve this problem, the memetic Web uses a proxy server called a C3W server through which each pad of this type accesses the original Web application. The access of different Web applications through the same proxy server solves the cross-frame scripting problem.

The memetic Web allows people to define both service federation and view federation of knowledge resources instantaneously in an *ad hoc* way just through direct manipulation of Web resources and pads. While mashup technologies focus on composing a new Web resource from available Web resources, the memetic Web focuses on the repetitive reediting, reuse, and republication of knowledge resources by end users in an instantaneous and *ad hoc* way.

5 Concluding Remarks

This paper focused on knowledge federation from different aspects; its taxonomy, its necessity, and its required technologies. First, it clarified three different aspects of knowledge federation, namely, semantic federation, service federation, and view federation. Then it focused on *ad hoc* knowledge federation of Web resources including Web applications. *Ad hoc* knowledge federation deals with two of the three aspects, i.e., service federation and view federation. Then it reviewed the current author's R&D on this type of knowledge federation, and proposed an extension of the Web toward the memetic Web. It also introduced our new version of the IntelligentPad system as an enabling technology for this extension. This version runs on *de facto* standard browsers empowered by Silverlight plug-in, and requires no IntelligentPad kernel running on clients.

The memetic Web will become a next-generation Web medium for re-editable knowledge resources, and for user-centric *ad hoc* knowledge federation of knowledge resources. It is accessible to anyone with a mainstream Web browser. The memetic Web will extend the power of community editing to the deep Web storing not only static resources but also executable resources.

The current author is especially interested in its application to accelerate the evolution of science and technology. Current e-science and cyber infrastructure projects are all based on GRID computing. The GRID computing provides HPC (High-Performance Computing) power, services, and workflow definition/execution/management systems for service composition. It emphasizes definition, execution, and management of routine or frequently requested jobs, but not user-centric, immediate, unforeseen *ad hoc* composition with existing resources. It provides tools for editing internal representations of service compositions, but no tools for directly reediting compound documents to compose new knowledge resources. GRID-based service composition provides unified component architecture for both GRID services and Web services, but cannot deal with Web applications. When a new service composition becomes available to public users, the task is already well known to almost every researcher as a routine or frequently used method. How can users define and execute an untried new combination of knowledge resources, for example, to discover interesting correlations between the El Niño-Southern Oscillation and the weather in Europe, volcanic eruptions, and diseases? We want to enable such trials to be set up within 10-20 minutes so that fleeting new ideas can be immediately tried and evaluated. GRID-based e-science and cyber infrastructure projects do not answer this question. *Ad hoc* knowledge federation will become a potential enabling technology to satisfy such requirements in science and technology.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (May 2001) 34–43
2. Biezunski, M., Newcomb, S., Pepper, S.: ISO/IEC 13250:2002, Topic Maps. PDF format (2002)

3. Dean, M., Connolly, D., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L.: OWL Web Ontology Language 1.0 Reference. Cambridge, MA: World Wide Web Consortium (W3C) (www.w3.org/TR/owl-features) (2003)
4. Dou, D., Mcdermott, D.V.: Ontology translation by ontology merging and automated reasoning. Yale University, New Haven, CT (2004)
5. Horrocks, I.: DAML+OIL: A Reason-able Web Ontology Language. In: Proceedings of the 8th International Conference on Extending Database Technology: Advances in Database Technology. (March 2002) 2–13
6. Baird, D.: Thing Knowledge: A philosophy of scientific instruments. University of California Press (2004)
7. Stefik, M.: The next knowledge medium. The AI Magazine **7**(1) (1986) 34–46
8. Heimbigner, D., McLeod, D.: A federated architecture for information management. ACM Trans. Inf. Syst. **3**(3) (1985) 253–278
9. Edwards, W.K., Joy, B., Murphy, B.: Core JINI, Prentice Hall Professional Technical Reference. (2000)
10. Gelernter, D.: Generative communication in linda. ACM Trans. Program. Lang. Syst. **7**(1) (1985) 80–112
11. Picco, G., Murphy, A., Roman, G.: Lime: Linda meets mobility. In: ICSE '99: Proceedings of the 21st international conference on Software engineering, Los Alamitos, CA, USA, IEEE Computer Society Press (1999) 368–377
12. Sun Microsystems: Javaspaces service specification, version 1.2 (2002)
13. Sun Microsystems: Jini technology core platform specification, version 1.2 (2001)
14. Ito, K., Tanaka, Y.: A visual environment for dynamic web application composition. In: HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia, New York, NY, USA, ACM (2003) 184–193
15. Tanaka, Y., Ito, K., Fujima, J.: Meme media for clipping and combining web resources. World Wide Web **9**(2) (2004) 175–184
16. Tanaka, Y.: Knowledge federation over the web based on meme media technologies. In: Lecture Notes in Computer Science, 3847. (2006) 159–182
17. Agrawal, R., Bayardo, R., Jr., D.G., Papadimitriou, S.: Vinci: a service-oriented architecture for rapid development of web applications. In: WWW '01: Proceedings of the 10th international conference on World Wide Web. (2001) 355–365
18. DBpedia. http://dbpedia.org/About
19. Maamar, Z., Benslimane, D., Ghedira, C., Mahmoud, Q.H., Yahyaoui, H.: Tuple spaces for self-coordination of web services. In: SAC '05: Proceedings of the 2005 ACM symposium on Applied computing. (2005) 1656–1660
20. Sotomayor, B., Childers, L.: Globus Toolkit 4: Programming Java Services. Morgan Kaufman, San Francisco, USA (2006)
21. Emmerich, W., Butchart, B., Chen, L., Wassermann, B., Price, S.: Grid service orchestration using the business process execution language (BPEL). Grid Computing **3**(3–4) (2005) 283–304
22. Mcllraith, S., Son, T.C., Zeng, H.: Semantic web services. Intelligent Systems **16**(2) (2001) 46–53
23. Tanaka, Y.: Meme Media and Meme Market Architectures: Knowledge Media for Editing, Distributing, and Managing Intellectual Resources. IEEE Press & Wiley-Interscience, NJ (2003)
24. Young, M.: Google Maps Mashups with Google Mapplets (Firstpress), 1 edition. APress (2008)
25. R..Ennals, Gay, D.: User-friendly functional programming for web mashups. SIGPLAN Not. **42**(9) (2007) 223–234

26. Ankolekar, A., Krötzsch, M., Tran, T., Vrandecic, D.: The two cultures: mashing up web 2.0 and the semantic web. In: WWW '07: Proceedings of the 16th international conference on World Wide Web. (2007) 825–834 SESSION: Semantic web and web 2.0.
27. Tanaka, Y., Fujima, J., Ohigashi, M.: Meme media for the knowledge federation over the web and pervasive computing environments. Advances in Computer Science – ASIAN 2004, Lecture Notes in Computer Science 3321 (2004) 33–47
28. Dawkins, R.: The Selfish Gene. Oxford Univ. Press (1976)
29. Tanaka, Y., Imataki, T.: IntelligentPad: A hypermedia system allowing functional compositions of active media objects through direct manipulations. In: Proceedings of the IFIP 11th World Computer Congress. (1989) 541–546
30. Okada, Y., Tanaka, Y.: Intelligentbox: a constructive visual software development system for interactive 3d graphic applications. In: Proc. of the Computer Animation 1995 Conference, Los Alamitos, CA, USA, IEEE Computer Society (1995) 114–125
31. Tanaka, Y.: Knowledge media and meme media architectures from the viewpoint of the phenotype-genotype mapping. In: SIGDOC '06: Proceedings of the 24th annual ACM international conference on Design of communication, New York, NY, USA, ACM (2006) 3–10