Web Semantics: Science, Services and Agents on the World Wide Web 8 (2010) 394–408

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

From the web of data to a world of action

Alan Dix[a,b,*], Giorgos Lepouras[c], Akrivi Katifori[d], Costas Vassilakis[c], Tiziana Catarci[e], Antonella Poggi[e], Yannis Ioannidis[d], Miguel Mora[f], Ilias Daradimos[c,d], Nazihah Md.Akim[a], Shah Rukh Humayoun[e], Fabio Terella[g]

[a] *Computing Department, Lancaster University, Lancaster, UK*
[b] *Talis, Birmingham, UK*
[c] *Department of Computer Science and Technology, University of Peloponnese, Tripolis, Hellas, Greece*
[d] *Department of Informatics & Telecommunications, University of Athens, Athens, Hellas, Greece*
[e] *Dipartimento di Informatica e Sistemistica, Universita' di Roma "La Sapienza", Rome, Italy*
[f] *Escuela Politécnica Superior, Universidad Autonoma de Madrid, Madrid, Spain*
[g] *EXALTECH S.r.l., Rome, Italy*

ARTICLE INFO

Article history:
Received 30 April 2009
Received in revised form 13 February 2010
Accepted 13 April 2010

Keywords:
Task support
Spreading activation
Intelligent user interfaces

ABSTRACT

This paper takes as its premise that the web is a place of action, not just information, and that the purpose of global data is to serve human needs. The paper presents several component technologies, which together work towards a vision where many small micro-applications can be threaded together using automated assistance to enable a unified and rich interaction. These technologies include data detector technology to enable any text to become a start point of semantic interaction; annotations for web-based services so that they can link data to potential actions; spreading activation over personal ontologies, to allow modelling of context; algorithms for automatically inferring 'typing' of web-form input data based on previous user inputs; and early work on inferring task structures from action traces. Some of these have already been integrated within an experimental web-based (extended) bookmarking tool, Snip!t, and a prototype desktop application On Time, and the paper discusses how the components could be more fully, yet more openly, linked in terms of both architecture and interaction. As well as contributing to the goal of an action and activity-focused web, the work also exposes a number of broader issues, theoretical, practical, social and economic, for the Semantic Web.

1. Introduction

From human readable web pages, to formal semantics of linked data and the emergent social semantics of tags and folksonomies, we routinely look to the web as both a source of information and a place to put data. However, the web is also a locus of action: users want to get things done, whether booking a hotel room, or editing an online spreadsheet.

The boundaries between web and desktop interaction are blurring. On the one hand, the traditional PC desktop is now inhabited by widgets such as the Mac Dashboard, web fast-download apps such as Java Web Start or Adobe Air, and expanded browser functionality such as Chrome. On the other hand, computation and applications that once were part of the desktop are now hosted on the web (for example word-processing with Google Docs), and various technologies enable web applications to function even when users have no connectivity to the internet (for example DojoX [33], Google Gears and the offline mode of HTML5 [75,76]). Furthermore, in emerging markets such as India and China, this convergence will be total, as the sole computing experience for many will be through mobile devices and predominantly the web.

So far most of these web-like or web-based applications are separate, and web activity is glued together by the user, often through crude cutting and pasting between web applications. This separation is epitomised by Google Web Elements,[1] which embed content in users' web pages, but, with the exception of Google maps, are largely sealed from one another and their context; while considerably more functional than data feeds, in the end no more integrated than early web syndication. Embedded applications, such as those in Facebook or Google Widgets, are integrated more richly with

* Corresponding author. Tel.: +44 1524 510319.
E-mail addresses: dixa@comp.lancs.ac.uk, alan@hcibook.com (A. Dix), G.Lepouras@uop.gr (G. Lepouras), vivi@di.uoa.gr (A. Katifori), costas@uop.gr (C. Vassilakis), catarci@dis.uniroma1.it (T. Catarci), poggi@dis.uniroma1.it (A. Poggi), yannis@di.uoa.gr (Y. Ioannidis), Miguel.Mora@uam.es (M. Mora), drid@ee.teiath.gr (I. Daradimos), nazakim@yahoo.com (N. Md.Akim), humayoun@dis.uniroma1.it (S.R. Humayoun).

[1] http://www.google.com/webelements/.

1570-8268/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.websem.2010.04.007

their respective underlying platforms, but again largely firewalled from each other, preventing synergistic interactions.

However, there is an emerging need to offer greater support to users in performing web-based activity that cut across individual applications, and potentially to partially automate common tasks.

For over 20 years the dominant interface paradigm has been instrumental: populating the interface with virtual 'things' (documents, shapes, files as icons) that are made as transparent as possible and manipulated 'directly' by the user [67,46]. However, the balance is changing and a level of 'intelligent', mediated interaction is becoming more accepted. This is partly because Moore's law means that it is easier to do more clever things, but more significantly because of the changing environment.

On the big-screen web (web on a desktop or laptop PC), this is largely due to the sheer size of data available on the web, so that Google search or Amazon suggestions become acceptable compared with searching enormous directory structures.

On the small-screen web, including mobile phones, the costs of ordinary interaction are relatively higher and so, as pointed out by one of the authors in a keynote even back in 1999, the advantages of using 'intelligent' techniques are comparatively greater [22]. Similar considerations are driving HP Labs' "Simplifying Web Access for the Next billion" (SWAN) project.[2]

In this paper we discuss several technologies offering the user automated task support, and in particular weaving together fragments of web and desktop interaction, so that the coherence in the user's mind is to some extent also reflected in the system. To be effective, such interactions have to fit with the human user; we are looking for 'appropriate intelligence', a blend of user controlled and computer aided activity set within a context of interaction that is meaningful and beneficial to the user.

Some of this currently uses standard semantic technology, some more bespoke representations. We will describe how the shared semantic representations promised by the Semantic Web can aid this integration, and discuss some of the obstacles and hence challenges for the development of core Semantic Web technology.

The decomposition of software enabled by mash-ups, plug-ins and widgets has tremendous potential for the democratisation of software, offering an alternative to behemoth applications and the stranglehold of massive vendors. To attain this, however, we need better ways for micro-applications to work *together* rather than just plug *into* larger software; a sit-alongside model for future software.

The next section will elaborate the motivation and background for this work, presenting a motivating scenario, a description of human activity that is both the context and also inspiration for the automated support we provide, and a short review of other ways global data is used to help user interaction. Section 3 then goes on to present the four core components of our wider vision for task support, reviewing additional literature and related systems for each as appropriate. In Section 4, issues of integration are discussed, based largely on two prototype systems, a web-based application Snip!t [27,30] and a desktop system On Time [13], which bring together various of the components discussed in Section 3. This integration experience is analysed in terms of architecture, inter-operability and user interaction. Finally Section 5 looks at implications and issues for the Semantic Web and web-based user activity arising from the experiences of design, development and deployment outlined in the previous sections.

This paper is partly about our own existing work, dating back over 10 years and its ongoing trajectory [23,26,27,31]. However, it is also about a vision common with others such as the HP SWAN project, Heath et al.'s call for a task-focused web [43], and Berners-Lee's 'underground map' of the future web landscape [7]; a vision of interactions formed through small units of task-based web activity being linked together by and for users to create a more seamless next generation web experience.

So, while the authors' own components, technologies and systems are described in some detail, the intention is not so much to promote these aspects of our own work, but more to use them as a proof of concept of this wider vision; using our own experiences to populate an initial roadmap for the future.

2. Motivation and background

2.1. Origins of the work

We have come to this work through a number of roots, but brought together in the TIM (Task-centred Information Management) project [56,9], part of the DELOS EU Network of Excellence on Digital Libraries. The backgrounds of the team included formal ontologies, ontology visualization, databases and intelligent internet user interfaces. Research within the field of Personal Information Management (PIM) [52], as its name suggests, is focused mostly on the user's information resources: calendars, files, emails, bookmarks. In contrast, TIM took the view that what users *do*, their activity, or tasks, is more often their primary goal.

In this paper we follow the natural extension of this vision, seeing the whole web of data, not just personal information, as the raw material for assisted user action.

2.2. Scenario—current fragmented interaction

Consider an imaginary user, Jane. She has just received an email from a friend, John, about the birthday of a mutual friend. John has noticed that the folk group 'The Weavers' are playing and suggests they all go out to dinner and a concert together.

Jane first checks the exact date of their friend's birthday in her address book. She then does a web search for 'The Weavers', finds the page describing their concerts, and books three tickets for the concert near the friend's birthday. She copies the postcode of the concert venue and enters it into her favourite restaurant review site, where she finds a nearby restaurant, follows a link to its web page and re-enters the date to book a table. So that she does not forget, she enters the information into an online diary, including the restaurant URL, a portion of the concert page describing how to get to the venue and a copy of the original email message. Finally, she copies the URL of the diary entry and pastes it into the notes field in the address book entry for the friend.

In this scenario Jane makes use of applications both locally on her PC (address book) and remotely on the web (restaurant booking). She performs some information related activities (looking up the birthday and the web search), some 'action' based ones (booking the concert and restaurant) and stores some information in a personal store (the online diary). Note that it was easy to link to entire web pages, but not portions, and easy to add links to web applications in local ones (URL of diary page stored in diary), but not the other way round (a copy of the email, not a link to it, in the online diary).

Some links between these individual interactions are automatically created for her (search results and link from review site to restaurant home page), but some she needs to accomplish 'by hand' (copying friend's name from email to address book, and entering dates). When she gets to a page or application, either by following links, or by deciding where she wants to go, she has to enter the same information (the date) several times into different forms, and she had found the date itself as the result of an earlier information interaction (looked up in her address book).

[2] http://www.hpl.hp.com/india/research/swan.html.

Table 1
Kinds of human action from [30].

	Pre-planned	Environment-driven
Explicit	(a) Follow known plan of action	(b) Means–end analysis
Implicit	(c) Proceduralised or routine actions	(d) Stimulus–response reaction

For Jane these comprise a single activity, but for her computer they are a series of largely disparate interactions.

2.3. The nature of human action

Accounts of human activity range from very formalized task analysis, which assumes or promotes pre-planning [20]; to those who consider more 'situated action' [70], where activity is seen as much more driven by the exigencies of the moment.

Similar to the latter, proponents of distributed cognition regard cognitive activity itself to be 'spread' between our heads, the world and often other people [47,50]. Early studies looked at Micronesian sailors, navigating without modern instruments for hundreds of miles between tiny islands. They found that no single person held the whole navigation in their heads, but it was somehow worked out between them [49,50].

More radically still, some philosophers talk about our mind being embodied, not just in the sense of being physically embodied in our brain, but in the sense of being in our brains, bodies and the things we manipulate in order to do 'mind-like' things [14].

At a more pragmatic level it is clear that day-to-day activities comprise a combination of environmentally driven and pre-planned actions. In the scenario presented above, the initial email is an example of the former, where the presence of the email triggers new activity, whereas the pattern of booking concert, booking restaurant and storing it in her diary, may be one that Jane has performed often before.

Both planned sequences and environmentally triggered activity may be explicitly considered, or more automatic or unconscious for the user. For example, whereas the user may be explicitly aware of the need to book the concert and the restaurant, the low-level action of filling out search terms into Google happens (assuming Jane is an expert user) largely without thinking, like riding a bicycle.

Table 1 summarises these kinds of action, and in previous work we have used this classification as a way of understanding how our different component technologies work together to support and augment normal user activity [30]. In particular, data detector technology (Section 3.2) helps the user to react to new data such as the name of the friend in the mail message, whereas means to predict task sequence (Section 3.4) are largely about supporting and potentially automating planned or routine sequences of actions.

In general a touchstone of our work is to imagine what a human helper would do and then, while not trying to pretend to be human, still seek technology which behaves similarly, including leveraging interactivity.

Note that while there is an extensive task analysis and cognitive modelling literature, even the most complete task modelling notations do not encompass all of the kinds of interactions in Table 1. Indeed, many who take a more 'situated' or holistic view of human activity would regard such (typically rigidly hierarchical) modelling as inappropriate, simplistic or misguided. In our own work we have not attempted to create such a total model of the human's activities, but instead used more integrative understanding of tasks and activity [Dx02, 30] and the framework in Table 1 to broadly structure our approach.

However, when we come to the more constrained user interactions with the system, then any sort of assistive or predictive algorithms must perforce contain a model of the user' tasks. Much of this is itself implicit in individual tools and algorithms, but in past and ongoing work we are developing more explicit and formal representations of task structure [12], in order to reason more effectively about system inferences and actions.

2.4. The web of data for people

The web of data is a very lofty goal, turning the human information of the web into machine interpretable data. However, of course the ultimate aim of this is so that this machine interpretation can achieve things for people or with people. People do not want clever technology; they want to get things done.

Of course, as in any area, there are applications where the benefit to individual users is real, yet very diffuse and indirect; for example, e-science[3] where an ordinary person does not know about the sophisticated management of formal ontologies and GRID services sitting behind the science, just that it furthers knowledge, and perhaps eventually, at some point, contributes to the products they, or their great-great-grandchildren, buy in the supermarket.

However, some of the most iconic web applications harness global reasoning much more directly in order to help personal day-to-day interactions. Google PageRank builds a model of importance of web pages based on vast computation over the link structure of the entire web, effectively computing a single eigenvector of the link structure regarded as a transition matrix [8]. However, all this sophistication and power is used simply to give you better web search. Similarly the algorithms behind recommender systems [63], found in web sites such as Amazon and the tagging systems of del.icio.us, harness mass data to help individuals find the right information; and we are beginning to see Semantic Web based applications in these areas, such as Swoogle,[4] Revyu [45] and SIOC [BB08].

While these applications make use of web-scale data (albeit mostly bespoke) to help individual user interactions, they do so at an application-by-application level with little connection between applications except web-links (and in the case of Amazon few of those). The user is left to thread together the disparate snippets of interaction.

There are existing applications that address this threading in particular domains, for example, the way CiteULike[5] recognises web-pages corresponding to citeable reference sources and TripIt,[6] which understands a range of travel sites in order to build itineraries. The challenge, which we begin to address in this paper, is how to achieve this in a generic way, and so make services such as TripIt either emerge from ordinary interactions, or at least be far easier to produce. These existing domain specific services demonstrate clearly that cross-application integration of activity as well as cross-repository integration of data is potentially both valuable and usable.

3. Components for user task support

In this section we present the core technologies we are using or developing in order to attain automated task assistance. First is the use of a *personal ontology* as a repository and *spreading activation* in order to model memory and context. Second is *data detector technology*, which can be used to turn unstructured data into the locus for interaction, thus triggering activity. Third are algorithms to help users during specific actions by *inferring relationships between form fields*, linking data to action. Finally, we consider methods to allow

[3] http://www.rcuk.ac.uk/escience/.
[4] http://www.swoogle.com/.
[5] http://www.citeulike.org/.
[6] http://www.tripit.com/.

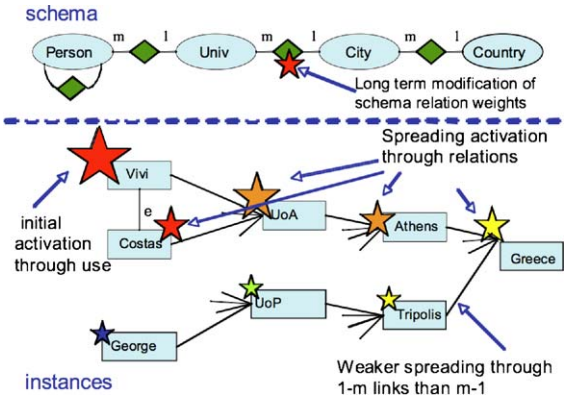

Fig. 1. Spreading activation through a personal ontology.

the system to propose to the user *potential actions and sequences of actions*; linking one action to another. In each case we will present both our own technologies and systems and also discuss related work in the area.

We note again that while we are describing our own components and systems, we are doing so not as an end point, but more as an exemplar of the potential for task-based interactions.

3.1. Memory and context: personal ontologies and spreading activation

Core to our approach has been the development of personal ontologies, describing a user's individual information space, including, for example, classes such as colleagues, work projects, friends, and events. Some of the classes in such an ontology will be *generic* (such as Person or Location), but the ontology will also include *egocentric* classes (such as Friend), which have a common meaning but are interpreted relative to the individual; and also *idiocentric* classes, which by definition do not mean anything to others (for example, Jane might have a subclass of Friends called "Friday Gang" who meet to dance on Tuesdays).

Our own work has initially been focused on explicitly populated personal ontologies [53]; however, there has also been substantial work on creating aspects of such ontologies automatically by mining email, files etc., as part of Semantic Desktop research (e.g. Gnowsis [65], Nepomuk[7]). Interestingly the results of this have often shown that users make more use of their own explicit ontologies, although the larger automatic data is clearly of value [68]. This underlines the need to ensure that underlying technologies are set within suitable user interaction, which in this case may be as simple as marking the provenance of data and selectively presenting it so that users are not swamped.

Spreading activation was originally formulated as a model of human language and memory [15,2], but applied to many areas including information management [17,41], ontology engineering [58], and web page adaptation [48]. In our own work, we have been using spreading activation to model the user's context [54].

The basic idea is simple. Suppose Alan has just had an email from Vivi. In Alan's personal ontology (Fig. 1), the Vivi entity will be activated. The activation on the Vivi entity is then spread to related entities (a colleague Costas and her institution UoA); this then spreads further to the city where UoA is located, its country, and eventually to quite distantly related parts of the ontology. This means that if, for example, Alan starts to fill out a web form to search for flights, then Athens would be top of the list of suggested locations.

[7] http://nepomuk.semanticdesktop.org/.

The algorithms we use modify the level of spread depending on the fan-out of relations so, for example, the rate of spread from city to country is greater then the spread from country to city, as each city has only one country, whereas a country has many cities. This and other parameterisation ensures that the spreading does not 'run away with itself' and produces relevant results. We have found that special care is needed with very heavily linked 'greedy' nodes in the ontology (notably 'Me/Self'), to avoid positive feedback effects.

It should be noted that the term 'context' is used in many different senses for adaptive systems. At one extreme is very long-term and almost static context such as user profiles and preferences, which can be used, for example, to influence ranking in search. Then there is more dynamic, but relatively long-lasting, such automatically muting a phone when you are in a meeting [38]. Finally there are the things that you are doing 'now', such as booking a trip to Madrid or writing a paper on task-centred interaction.

Various frameworks have been proposed, mostly focusing on the first two of the above. For example, in Heath et al.'s [44] categorisation 'personal context' (profile and preference) and 'knowledge context (file system, email etc.) are largely long-term, whilst 'computing context' (e.g. state connectivity) is largely in the middle ground. Dix et al.'s framework [24] for mobile context is also focused principally on medium term context with, as would be expected, location being central. However, the faster timescales are not neglected entirely and Heath et al. [44] mention context-sensitive menus in the sense of reacting differently to images or audio files, and Dey et al.'s influential Context toolkit [19] includes support for context such as "making dinner".

Within our framework we also deal with multiple timescales. The spreading activation is used to model short-term memory, the relatively instant reaction, so that the next web interaction may be influenced by the last email. Entities that receive high short-term activation then get their medium-term activation incremented, which is used to model aspects of context over dozens or hundreds of interactions covering periods of hours. Finally, sufficiently high medium-term activation triggers long-term activation, modelling concepts and things that have general importance, which are also explicitly influenced by the users as they add information to their personal ontology.

These three levels in part have a pragmatic origin, but they also mimic human memory. The short-term (working memory) vs. long-term memory distinction is well established [1], but the medium memory term (or mezzanine memory [26]) is clearly a common phenomenon ("what am I currently doing"), but less well studied, with the exception of models of 'long-term working memory' in text comprehension [36] and situation awareness in command and control environments [35].

All the levels of memory have means of both being activated and also decaying over time, but all have essentially a single stream of context. In the future, we also need to emulate the way humans are able to easily swap between contexts and build up memory in each. For example, if reading email we may swap back and forth rapidly between several activities and yet, for us, each activity maintains its own history of contexts and topics. The challenge is to enable the automated system to do the same. While we have some proposals for this, it is currently work in progress.

The fuzziness of spreading activation is especially useful when linking personal data into the web of 'linked data' [6]. This is because we can, in principle, selectively pull in data from the web that is connected to 'hot' topics in the personal ontology, leading to a cache of activated entities from anywhere in the web of data. The details of this are described elsewhere [31]; however, early work suggests that this is tractable so that in principle the entire web of data can then be regarded as if it were part of the user's personal resources. To do this requires us to accept a level of defeasible rea-

soning, as only sufficiently 'hot' web data will be in the local cache (we have termed this *warm word assumption* reasoning). However, it means that a substantial amount of common sense knowledge virtually comes 'for free'. For example, if Jane's personal ontology records that John lives in Milan, then Italy can also become activated even if the fact that Milan is in Italy is not explicitly recorded, as the information will be drawn in from Geonames.[8]

3.2. Triggering activity: data detectors

Data detector research dates back to the late 1990s, including the Intel Selection Recognition Agent [62], Apple Data-Detectors [61], CyberDesk [72] and onCue [23]. Data detectors use some form of textual analysis to look for data types or key terms in text such as names, dates or locations, which are then used to suggest possible actions.

One of the authors was involved in the development of onCue. This analysed clipboard contents using simple heuristics to determine the type of data that had been copied/cut. Depending on the kind of data found in the clipboard, onCue changed icons in a sidebar representing different tools and actions available for the data, both on the PC and on the web. For example, a personal name would mean various directory web services were suggested, whereas a table of numbers could be inserted into Excel or used to generate an interactive visualization.

Earlier work, in particular the Microcosm hypermedia system developed in the late 1980s [39], can also be seen as examples of a wider class of systems that automatically look for data values in text to be used as a source of either simple hyperlinks, or user activity.

Simple data-detectors to create live links from URLs and email addresses are common in many applications, although more sophisticated versions less so. Apple Data Detectors are still in the Mac OS infrastructure, but are rarely included in applications. This may be because of the complexity of installing new data detectors compared with the ease of Dock widget and iPhone app downloads, but also because the more 'incidental' interaction ([25], Chapter 18) of data detectors, where the system offers help opportunistically, does not fit with the traditional download/install model for software.

More recently a number of systems have arisen in this broad area including Microsoft SmartTags, Citrine, an intelligent clipboard transformer [69], and CREO, which uses a large semantic database to find potential topics in web pages [37]. A number of commercial systems offer ways for blogs and web pages to create dynamic links to eCommerce sites (e.g. for books or music), usually requiring some level of hand-annotation rather than automatic target detection.

In the Semantic Web arena, microformats and RDFa, whilst also requiring explicit markup, offer ways to 'late bind' content to other potential data sources, and services such as OpenCalais[9] offer a level of automatic markup. The WordPress plug-in zLinks,[10] requires the blog author to manually mark potential link text in a post, but then the system dynamically connects this to matching data using its own RDF services. Magpie [34] is a totally automated browser plug-in, which scans web pages for terms that occur in a number of knowledge sources, very similar to the early visions of the way Microcosm could be scaled for the web [11]. Like Microcosm, these Semantic Web systems link to data whilst data-detectors tend to be more focused on linking to actionable resources.

Each of the above, both data detectors and related systems, is based *either* on some form of syntactic matching of the text (e.g. regular expressions, BNF), *or* on literal matching against large corpora. In our own work on data detectors within the Snip!t system [27,30], we have combined corpus lookup with syntactic rules. For example, if a single name in the text (such as "Alan") is found in a list of common given names, this can trigger a syntactic rule to match the whole of the name ("Dix, Prof. Alan"), using a form of inside-outwards parsing.

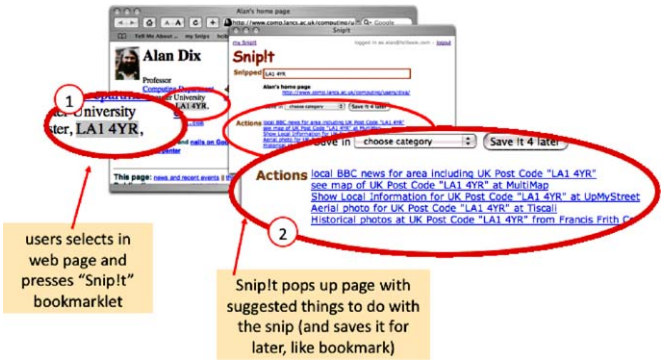

Fig. 2. Snip!t data detection and action suggestion.

In Fig. 2, we see an example of Snip!t in action. The user has selected a portion of a web page (1) and 'snipped' it, forming a sort of bookmark entry that also includes part of the text of the page. The snipped section contains a postcode which the recogniser component in Snip!t has identified and so the snip page (2) includes 'actions' for the postcode, including linking to mapping sites, local weather and news.

Snip!t uses a bipartite architecture for its data detectors, inherited from onCue. Most data-detector technology, including Apple Data-Detectors, specify a pattern to be detected and some action to be performed based on the action in one unit. In contrast, onCue and Snip!t have two separate kinds of component: *recognisers* that map syntactic patterns or table lookup results to semantic data types; and *services* that are triggered by particular data types and link to web or desktop resources that can deal with that kind of data.

This separation has proved very powerful in terms of reuse of components, as a single recogniser can provide data that can be used by many services: note that in Fig. 2 there are five actions triggered by the postcode. Furthermore, the linkage between the two is a form of semantic annotation, so that there is the potential for inter-operability with other Semantic Web technology, for example, provider-side annotations in the form of microformats or RDFa.

Snip!t has an API for writing more complex recognisers and services, but simple components can be specified using XML description files. Fig. 3 shows the description file for the UK postcode recogniser. We shall step through the main features.

The initial tag says that this is a regular expression-based recogniser and the regular expression to be matched can be seen inside the `<pattern>` tag. The pre and post context say that a valid postcode must begin and end at word boundaries.

The `<keyed>` tag is more interesting, declaring that a simpler type 'MIXED' triggers this recogniser. A lower-level recogniser scans for words that contain a mixture of letters and numbers (and are thus likely to be parts of various forms of codes). The postcode recogniser is only activated when a MIXED word has already been spotted in the text, making scanning large texts more efficient. The same mechanisms can be used hierarchically for more complex types; for example, triggering the address recogniser only when a postcode has been matched. The name recogniser also uses this mechanism and is only triggered when a word has been found in lists of common given names or family names, which in this case also helps to reduce false positives.

[8] http://www.geonames.org/.
[9] http://www.opencalais.com/.
[10] http://zitgist.com/.

```
<simpleregexprecogniser>
  <name>ukpostcode_recogniser</name>
  <title>UK Postcode recogniser</title>
  <keyed>
    <keys>MIXED</keys>
  </keyed>
  <pattern>
    <pre_context>\W</pre_context>
    <match>([A-Za-z][A-Za-z0-9]{1,3})[
\t]{1,6}([0-9][A-Za-z]{2})</match>
    <post_context>\W</post_context>
  </pattern>
  <fields>
    <field name="postcode" value="ALL" />
    <field name="outer" value="REGEX 1" />
    <field name="inner" value="REGEX 2" />
  </fields>
  <match>
  <type>ukpostcode</type>
  <description>UK Postcode
$$</description>
  </match>
</simpleregexprecogniser>
```

Fig. 3. Snip!t recogniser description file.

Note also that the recogniser not only matches the text as a postcode, but also has sub fields 'inner' and 'outer'. These terms are not commonly used, but are the official terms to represent the two halves of a UK postcode, e,g, in "LA1 4WA" the inner part is "LA1" and the outer part is "4WA". These subfields are identified by the portions of the regular expression that match them: "`REGEX 1`" denoting the text that was matched by the first bracketed term "`([A-Za-z][A-Za-z0-9]{1,3})`". Similarly a person name has subfields for title, given name, and family name, and a date has year, month and day.

Fig. 4 shows a similar description file used for one of the postcode services. It is a URL-based action (basically creating a URL from a pattern specified in the ⟨urlpattern⟩ tag), and specifies the type that is required, 'ukpostcode', the URL pattern and also a pattern for text to use in presenting the action to the user. Note that the outer and inner subfields are both used in the URL pattern, whereas the description uses the postcode as a whole.

3.3. Linking data to action: inference and support for form filling

Browsers perform a level of automatic form filling using a combination of the URL of the page and the names of input fields. Research systems, including that of the Simplicity project [64], W3C draft "Client Side Automated Form Entry" [74] and Hausenblas' "profile auto-complete" [42], have extended this to include mappings between specific form's field names and user profile data.

Our own work has extended this by automatically inferring rich ontological type tags such as "name_of Friend" over an unconstrained personal ontology, and furthermore linking the semantics

```
<urlservice>
  <name>bbc_postcode</name>
  <type>ukpostcode</type>
  <title>bbc.co.uk UK</title>
  <description></description>
  <icon>http://www.bbc.co.uk/favicon.ico</
icon>
  <descpattern>local BBC news for area
including UK PostCode
&amp;quot;$postcode&amp;quot;</descpattern>
   <urlpattern>http://www.bbc.co.uk/cgi-
bin/whereilive/query/runquery.pl?loc=$oute
r+$inner</urlpattern>
</urlservice>
```

Fig. 4. Snip!t service description file.

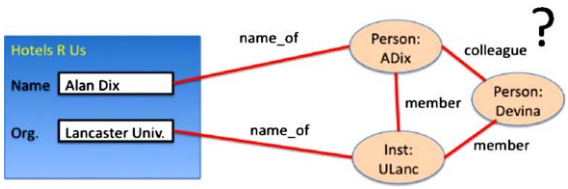

Fig. 5. Form-field inference: (i) match terms in form to ontology and (ii) look for 'least cost' paths.

of multiple fields of the same form or even fields in separate forms within a task sequence [28,29].

The first time a user encounters a web form, the system will only be able to offer suggestions based on any implicit or explicit data type information already in the form, for example, using the names of the fields as in standard web-browser pre-filling. However, the spreading activation means that suggestions can be tailored to the current context. In the case when there is insufficient data to produce an acceptable suggestion the user may explicitly choose values from the ontology or simply enter text by hand.

When the form is complete, the system matches the text in the fields against strings in the ontology and this identifies candidate entities and property slots corresponding to the *individual* fields. The inference algorithm then looks for 'least cost' paths through the ontology between candidate entities for the field.

Fig. 5 shows a simple example. On the left is a form consisting of two fields, "Name" and "Org.". If the "Name" field is already semantically tagged to say it is a Person name, then the system can simply suggest the names of people in the personal ontology. Alternatively, if the web form is not annotated (which is usually the case), the user may simply type the name "Alan Dix". At this point the system matches the text "Alan Dix" against strings in the personal ontology and finds a match with the name property of the entity 'ADix', which is a Person. Simple generalisation then allows us to infer a type for the "Name" field "name_of Person".

However, the form contains both a name and a university name, and a knowledgeable human assistant would notice that the institution was the one where "Alan Dix" worked. The algorithm we use does the same reasoning. Having found an entity 'ADix' (or possibly several) with property "Alan Dix" and another 'ULanc', with a property matching "Lancaster University", the algorithm looks for paths through the ontology linking 'ADix' and 'ULanc'.

In fact, in Fig. 5 two such paths between the entities have been found in the personal ontology. One path is directly through the 'ADix' entity to the institution 'ULanc'. The second path is indirect through a colleague of Alan Dix, 'Devina', who also works at Lancaster University. The system will choose the former as the preferred path as it has 'lower cost' where cost is based on the length and the fan-out of the relations traversed, and potentially weighted based on the current levels of activation.

The concrete paths through the ontology are then transformed into rules by treating internal entities as wild cards, generalising from a single case. So the preferred path for Fig. 5 would become:

`name_of > Person(p) > member > Inst(i) > name`

When the user next starts to fill in a name, for example, "George Lepouras", the system can pre-fill or suggest "University of Peloponnese" by following the rule.

Note that the results of this process are different from those obtained from more common probabilistic techniques. For example, if a form has a field for 'name' and for 'place', then a simple system might pre-fill both fields based on past form filling, but of course, not be able to deal with unseen forms. A more complex system might be able to pre-fill the 'place' field with 'Lancaster' after the name field had been filled out with 'Alan Dix' as the place 'Lancaster' is often associated with the name 'Alan Dix', but only after

being presented with training. In contrast, the form field inference can offer useful suggestions after only one user-completed form and furthermore, once the field relationship has been inferred for one user, could be used for others with no training. Furthermore more complex relationships, such as triads that would be require very large training sets for probabilistic methods are no more difficult for the form-field inference. When linked to the spreading activation, suggestions can also influenced by recent activity. So if the most recent email was about 'Scottish folk dancing' and from someone connected with Alan's personal rather than professional activities, then the spreading activation would be more likely to propose 'Tiree' as the place even if Lancaster is more likely ignoring context.

The fact that the fields are linked presents some interesting interaction issues, as the set of suggestions for one field various depending on what others have already been filled. For example, in Fig. 5 the options for the "Org." field, changes if the user first selects or enters data for the "Name" field and *vice versa*. In fact, this happens already in many (non intelligent) web forms, such as aircraft booking: as one selects the departure location the arrival location choices are filtered to only show those where the airline offers flights.

In the airline example, the rule limiting the arrival location is fairly obvious; indeed the reverse is annoying when an (even less intelligent) form allows one to enter impossible combinations. However, it is not clear how complex this can become without confusing the user. If we have a whole sequence of forms that are going to actioned, then a 'filling in fields' metaphor may break down and instead it may be preferable to cycle through a set of fully completed (but editable) options, as this would expose the interactions between fields more clearly. Long-term deployment studies will be needed to answer some of these questions.

More generally, users appear to be willing to accept assistance, defaults or suggestions, even if they are not immediately comprehensible, so long as the impact of wrong decisions is not too great (see Section 4.2 below). Hence offering suggestions, or pre-filled sequences, seems appropriate as long as the user can freely edit them when the inference is not to their liking.

To some extent, form filling seems a limited form of interaction, although, as is evident, it offers significant technical and interaction challenges itself. However, it is representative of any kind of user action that requires 'parameters', whether this is a file being dragged over an application icon or an image being cut and pasted between drawing and word-processing applications. Of course, form-based interaction is very important on the web and probably more so on mobile devices, and is also common in other task automation systems, not least Apple Automator.

3.4. Linking action to action: inferring task sequences

Data detectors help the user initiate action based on automatic semantic annotation of text such as email messages or human-readable web pages. If instead the user has chosen for herself a form to complete, then the form-filling algorithms help her to complete the chosen action. The remaining piece is to help the user in *selecting* actions, and potentially automating common sequences of actions for the user.

Task sequence and structure inference has a long pedigree (e.g. Cypher's work [18], Beale and Finlay's 1992 edited collection [5], and more recently Lieberman's "Your Wish is My Command" [57]), but has never 'made it' into mainstream interfaces. This is partly because, as a formal problem, grammar induction is 'hard' and computationally expensive. This is made more difficult because users may interleave several tasks, confounding sequence predications based on Markov models or *n*-grams [40]. At a human level it is easy to leave the user more confused than supported.

An interesting recent example is DabbleDB's magic/replace,[11] a web-based application to help users clean up tables of data prior to import into an online database. Users perform sequences of edit operations on sample records and the magic/replace infers the general transformations, such as changing capitalization. The algorithms are simpler than many used in intelligent user interface research and the application is further simplified by being data-focused rather than based purely on sequences of actions, yet it is surprisingly powerful. However, despite the potential demonstrated by DabbleDB, it is rare to see inference at even this basic level of sophistication in production systems.

While DabbleDB demonstrates that domain-specific inference can be of substantial value, we aim to address more generic domains. Our own preliminary work suggests that techniques to 'thread' low-level actions may be combined with interactive and incremental learning, to offer a way to cut through the Gordian knot of task inference.

We have two basic strategies. The first simply uses the existing mechanisms. If the result of an action is semantically marked up (e.g. XML from a web service, microformat-annotated HTML), then we can use it to propose other web forms where one of the input fields matches the type of an output value from the previous step. Alternatively, if the output is human readable only, then we can use the Snip!t recognisers to annotate on the fly. This means that the outputs of one action can be used to propose the next action, so that a sequence can be suggested step by step. It is limited as it only allows single-step proposals, but leverages the existing tools.

However, we are also working on more sophisticated task-sequence inference. Happily the use of the ontology makes this easier too. Assume the user is performing a task, such as the scenario in Section 2.2, involving a number of web applications. If only one web application is involved then the application will take the user from form to form using its own business logic. However, when users are dealing with several sites they have to maintain this linkage themselves. Given several such form-based actions, we try to match the input or output fields of one form (assuming suitable markup of the output) with the inputs of a subsequent form in the user's task sequence.

Sometimes the connection is very direct, for example the same text (such as a date) is being used on several forms. For other pairs of forms, we might see the user entering the date of a meeting into one form, and then the location into a different form soon after receiving an email from the meeting convener. In this case we may generate indirect links through the ontology using the algorithm outlined in the previous section. Finally the user might drill down through the personal ontology from an input/output that already exists in it, and choose the new input this way, giving a clear link between the actions.

In all cases, the end effect is to have the trace of user interactions linked together through the ontology. Whilst actions from different user tasks may be mixed up chronologically, there is a semantic link between actions relating to the same task. The semantic links can then be used to create threads through the trace, which will correspond to different tasks or sub-tasks the user is engaged in.

Fig. 6 depicts this. The darker and lighter squares represent two sequences of actions. Of course, the system would not be aware of this difference, and indeed without the linkage might need many exposures to say "ACB", "ADB" "AXB" to work out that action 'B' frequently follows action 'A'; precisely the problem of dealing with interleaving.

[11] http://cleanupdata.com/.

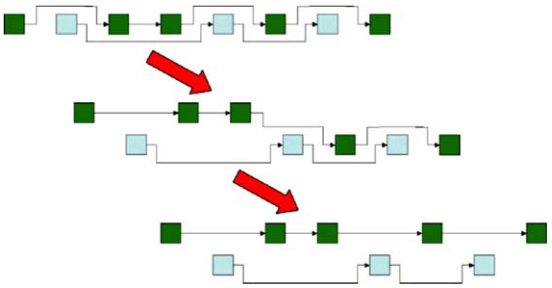

Fig. 6. Teasing apart task threads from interleaved user actions.

At the top of Fig. 6 we see the two traces mixed together chronologically, as they would initially appear to the inference system, with semantic links going from action to action. The lower 'frames' show the two tasks being 'pulled apart' using the semantic linkage. This effectively finesses the problem of interleaving: instead of being a computationally hard problem, it is simply a matter of chasing links.

Once the two sequences have been disentangled it is easy to use task prediction rules such as Markov models or grammar induction, or even very simple rules with single-step learning: if in a previous task sequence form A was followed by form B and C then we can offer these as a suggestion when the user next chooses form A.

Furthermore, the concrete linkage between the actions in the original task sequence can be generalised in the same way as the within-form field relationships described in the previous section. This means we cannot only propose form B followed by form C, but pre-fill some or all of the fields of B and C based on the inputs and outputs of A.

Note particularly that this form of task inference allows single-step learning, unlike Markov or similar techniques that typically require substantial corpora of examples.

4. Putting it together

Having seen the different component technologies, we now discuss how they fit together architecturally and in terms of user interaction.

4.1. Architecture

Fig. 7 shows a simplified view of how these different technologies fit together. The solid arrows represent 'control' flows, which influence the choice of the next action, whereas the dashed arrows represent information flows, although the information of course influences the decisions.

On the left we have incoming emails, active files, web pages or other kinds of documents and data that the user encounters. This may have existing markup, but if not, or in addition to the existing markup, data detectors are used so that we have semantically annotated text. This is then matched against actions in service descriptions as in Fig. 4. Alternatively the user may spontaneously decide to perform some action.

The form inference engine is used both to learn from user interactions and to make suggestions to pre-populate parameterised actions. The task-sequence inference process then records interactions in order to learn patterns and also makes suggestions based on prior learning drawing on a stored history of past user action. Note, this history is shown as a separate store, but will typically be closely linked to, or part of, the personal ontology.

Underlying all of these is the personal ontology itself and spreading activation. These may influence the initial data detectors, for example, 'Prince' may be interpreted differently if the user has been having an email discussion about pop stars than if the interchange concerned Buckingham Palace. They will also influence the choices of actions, and the form and task-sequence inference processes.

Note that this ontology may have different levels of reasoning and this will influence other aspects of the picture. Snip!t uses a simple form of forward chaining so that, for example, when a 'date' is detected it is compared with the current date and one of the subclasses 'today', 'future date' or 'past date' is also asserted. In contrast, On Time (see below) uses a more sophisticated underlying reasoning engine supporting the DL-Lite ontology language [10].

While this is the planned picture and all the individual components exists, not all the interactions are currently in place. Snip!t includes data detectors and action selection. Also the form-inference has been linked to the personal ontology, but not currently integrated with Snip!t. The most extensive integration is through a desktop prototype system, 'On Time' [13]. On Time includes visualisation of the personal ontology, and spreading activation initiated by data found using data detectors on recently

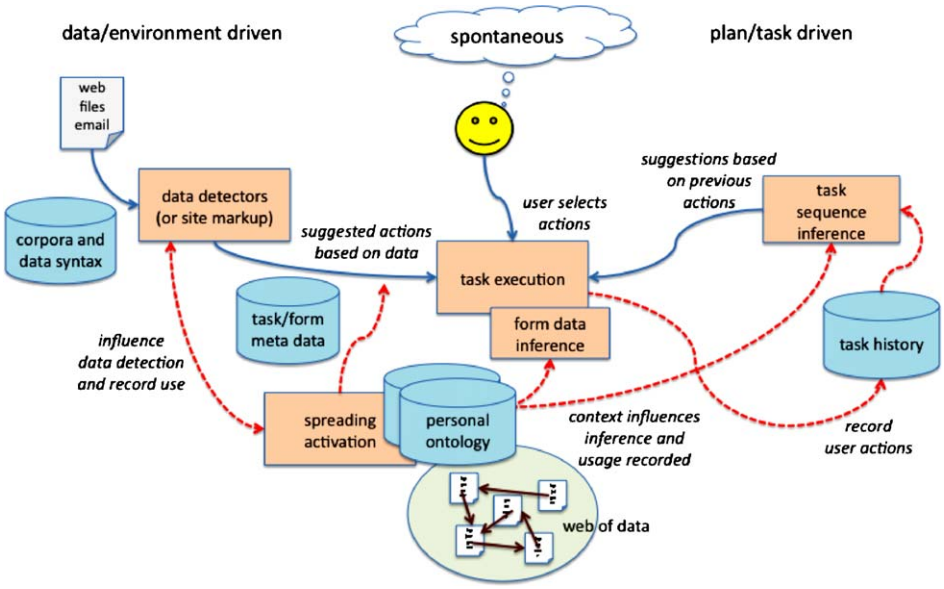

Fig. 7. Interaction between component technologies.

active files. The spreading activation is used to propose actions based on the most active entities as well as to pre-fill forms. However, in all of these, task sequence support is limited to at best rudimentary chaining through data detectors and the spreading activation.

A key issue, as we move from individual components to an integrated system, is to retain and if possible increase the independence of the components so that we can eventually seek inter-operability between alternative technologies, for example, server-side markup vs. client-side data detectors for initial semantic annotation. In a web paradigm, integration is as much about deconstruction as combination; for example, the data detectors in the Snip!t application were originally closely tied to the core code base, but this has now been separated into three parts: (i) the core websnip application, (ii) data detection using recognisers, and (iii) action suggestion using services. We follow a general restructuring pattern of 'interface drift' [21], initially separating code by moving it behind internal APIs followed by radical excision into web services.

Nevertheless, while we are seeking to retain independence of components, we also wish to create a more integrated user experience. This seems to be a core challenge for all web interaction.

Fig. 8. onCue toolbar.

4.2. Interfaces for intelligent interaction

There are several major interaction challenges in the work that will arise principally as we tackle full integration into a single open system. The implementation of efficient presentation and interaction techniques is as crucial as the algorithms for correctly identifying users' actions and understanding the context of their actions. One issue is how to present support 'tips' or suggestions to the user. This clearly should not interrupt the user's work, which would be worse than the absence of any support, but should be readily available.

The Microsoft Office Assistant, 'Clippy', is a prime example of what can go wrong in automation. It is based on sound recognition algorithms that can potentially be useful in helping the user compose letters or do other Office tasks. However, it is modal and pops up in the middle of typing. Even if the advice is useful it has broken the user's flow of thought, but if it is wrong the user is left very annoyed. Hatred of 'Clippy' has spawned numerous web pages and blogs, and has even been the topic of an Honors Thesis at Stanford [71].

Snip!t avoids this kind of problem by having a separate tab for suggested actions, but this has the disadvantage that it can easily be missed. In contrast, onCue, the early data-detector web assistant that one of the authors worked on, was more proactive, continually monitoring the user's clipboard activity and using an 'always on top' side palette (Fig. 8) that adapted to the clipboard contents [23]. However, to avoid a 'Clippy' scenario, onCue was designed according to core principles of 'appropriate intelligence':

1. It should do good things when it works.
2. It should not do bad things when it does not.

The first is the obvious rule for making good demos of clever things. The second takes as given that any sort of intelligent support will sometimes be wrong. This acceptance that intelligent algorithms will often be wrong is crucial for making intelligent user interfaces (and indeed any user interface) acceptable for long-term use without becoming annoying.

As an example of appropriate intelligence, contrast Clippy, which annoyingly interrupts typing, with another feature of Microsoft Office, the Excel sum (sigma) button. When the user presses the sum button the formula function 'sum()' is entered, and some cells are preselected. The preselected cells are based on very simple heuristics, first scanning above the current cell for a contiguous vertical set of numeric cells, or failing that scanning to the left. If the pre-selected cells are not what the user requires than she can simply select other cells. The normal action of selection means that this requires no more effort than selecting the cells with no pre-selection—the only cost of failure for the sum button is that the user has to check visually whether the selection is correct.

onCue was explicitly designed with these rules in mind. The onCue palette was always on top and hence potentially distracting, but it was not modal, so it never 'stole' keyboard focus (as 'Clippy' does), and when the icons changed due to changes in the clipboard content, this was faded in over a period of about one second, which was sufficient to prevent visual distraction yet fast enough to ensure they were always 'up to date' with the current clipboard [23].

In general, 'appropriate intelligence' is about detailed interaction design that embeds intelligence within a forgiving interaction framework. Such design has an emphasis on user control and transparency of computer activity (not just data), which minimises the cost of corrective actions. This then has a knock-on effect on underlying algorithms where the critical criteria are not about maximising accuracy (although this is still important), but more about the comprehensibility of results and 'good enough' measures.

In our own work, in both spreading activation and form inference we are producing multiple weighted results, not single 'best' values, so that it is possible to present alternatives to the user, not "all or nothing" proposals. Furthermore, this means that we can in the future use the results of user actions as explicit or implicit input into further reasoning—for example, using the fact that a user has accepted a suggestion as an indication that it was a correct inference.

On Time follows a similar principle of offering proactive suggestions in a non-modal side bar (see Fig. 9). This includes suggestions for the next task (top), and also items detected in recently active files with suggestions of how these can be stored in the personal ontology.

onCue, Snip!t and On Time are all offering single actions. When the suggestions or support offered by the system are more complex (and probably potentially most valuable), they will need to be presented in a way that is comprehensible, both in terms of what is going to happen, and why it is appropriate. However, traditional expert-system style explanations are clearly not going to be appropriate in the midst of ordinary user interaction. It is likely that techniques should exploit the interactive and action-based

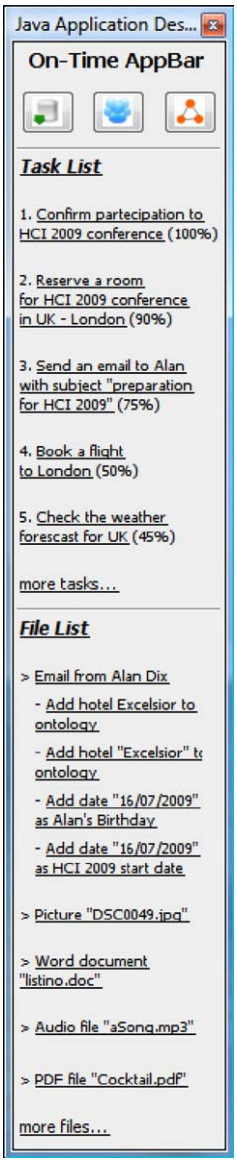

Fig. 9. On Time AppBar.

environment; for example, animating what is happening/will happen, and perhaps allowing the user to play/rewind these automated actions, or see them laid out spatially (as in Apple Automator) and select which to accept.

Support presentation must also work well for both the small screens typically found in mobile phones and PDAs and the big screens used in many PCs. As we adapt to these, we will need to consider alternative approaches including drill-down, cue-card paradigms and multi-modal interfaces. As noted, it is perhaps with small screens that intelligent interaction techniques are likely to be most valuable, yet it is also precisely here that the presentation and integration challenges are most demanding.

4.3. Evaluation and scaling

As described in the introduction, our main aim is to establish feasibility and proof of concept, so we do not expect to have fully polished user interfaces.

Snip!t has been deployed now for over six years and used regularly by a small number of users. It has not been subject to formal evaluation, however we have had user feedback and bug reports. The main feedback has been feature request (e.g. Unicode support and wish for it to operate on PDF documents as well as web pages) and problems with the classification interface for the bookmarks (it predated tagging), but there has been no explicit feedback or problems related to the more intelligent behaviours. However, we are aware that the latter is limited to the recognisers and services pre-programmed into the system as described in Section 3.2. Hence there is clearly a need for more ground-up learning of services such as the form-field inferencing.

On Time has not been subject to extensive user evaluation however preliminary formative user studies have been performed using cooperative evaluation techniques [73] with six users. Simple quantitative measures (errors and timing) were used for two closed tasks, involving the creation and modification of entities in the personal ontology, and in addition some qualitative responses were collected. While the users were satisfied with the overall system concept, the tests did reveal a number of more peripheral usability issues, leading, for example, to the redesign of some icons, and also some issues related to visualisation of the ontology.

However, while the usability of the final systems will be essential, our main concerns have been thinking towards scalability; things may work well with a single user and small test-case personal ontology, but fail under real volume use. This has influenced algorithm design throughout. For example, in Snip!t, the use of triggered recognisers was partly driven by this consideration. Most of the syntactic recogniser are only applied when triggered by a previous table-lookup recogniser; this means that very large numbers of recognisers can be included without needing to execute regular expression scans for all of them (or compiling them into a single monster machine).

The form-field inference component was deployed as a standalone tool for a period of three months, with an initial ontology containing near 500 instances of personal information. During the three months of use over 371 instances were recorded covering 31 different web forms, based on which the inference algorithm produced 76 different rules, processing each instance in less than 60 ms. The performance of the algorithm that chooses and executes rules is able to find the set of values for each field in 3.11 ms for each web form instance. This performance allows us to use these algorithms interactively with larger ontologies in a common personal computer.

In order to test the accuracy of the algorithm, after the deployment (when all the correct field values were known), the data was split into training and test sets and evaluated (see Table 2). In 22% of cases there was no suitable rule (either the form was only seen once, or the algorithm was unable to construct rules due to unavailable paths in the ontology). However of the 78% with a matching rule, more than 3/4 of cases had the correct value as the single suggestion or one of the suggestions. Note too that this interface is simply suggesting auto-fill values, hence, following the principle of appropriate intelligence, in 60% of cases is helpful, but in the 20% of cases when there is a false positive does not unduly hinder the user.

For the spreading activation, our aim is to be able to include links to arbitrary web data and so we have developed caching-based variants of spreading activation that pull in data from external sources only when needed; in broad terms when an entity's activation exceeds some limit then all related triples are fetched. To test this we used a data set of programmes and music released as part of the BBC Backstage initiative [4]. This includes approximately 20 million triples describing approximately half a million entities accessed via a SPARQL endpoint [32].

The results of this showed that the working set of active entities could indeed be kept manageable and capable of sub-second response times by choice of suitable thresholds, and furthermore by modifying the algorithm in this way the results were not signif-

Table 2
Form-fill inference accuracy.

	Rule match		No match
	Value found	Not found	
Single result	25.0%	3.13%	21.9%
List of suggestions	34.4%	15.6%	
Total	59.4%	18.7%	21.9%
As % of rule matches	76%	24%	

icantly impacted in terms of the choice and ranking of the more highly activated entities [31]. Fig. 10 shows an example of the results obtained. Each data point represents a single entity/URI: the *x*-coordinate is effectively unconstrained activation and *y*-coordinate is the activation using caching and threshold. The points to the left along the horizontal axis have zero activation in the modified algorithm due to the threshold. Critically the rank-order and numeric activation of the higher-activation entities to the right is virtually unchanged, thus demonstrating robustness of the algorithm.

In addition, the working set of active entities that resulted ranged from a few hundred to two thousand despite a virtually unlimited data set. This means that many other parts of the system, such as the form-field inference component, need only operate on this restricted set of entities rather than the complete web of data, further enabling efficient scaling.

5. Challenges for a web of action

Each core technology has its own challenges, as does the integration process. Some of these are made simpler by a more pragmatic approach (e.g. making multiple suggestions to the user in cases of uncertainty), but others are made more complex. In this section we will consider some of the issues that are highlighted by work in the area to date, and those that are likely to become important in the near future.

5.1. Meta-information on human web sources

The service descriptions used by Snip!t (Fig. 4) are effectively providing meta-information about web applications intended for human use. This drew on a light-weight XML framework used in onCue. Similar meta-description formats are found in other applications such as those (like Firefox and A9) using OpenSearch plugins; these meta-descriptions include parameters and types of GET and POST requests for web forms (see Fig. 11), and sometimes also information on how to parse resulting human readable web pages. Similarly OExchange[12] offers a way for online bookmarking and related service to publish their services.

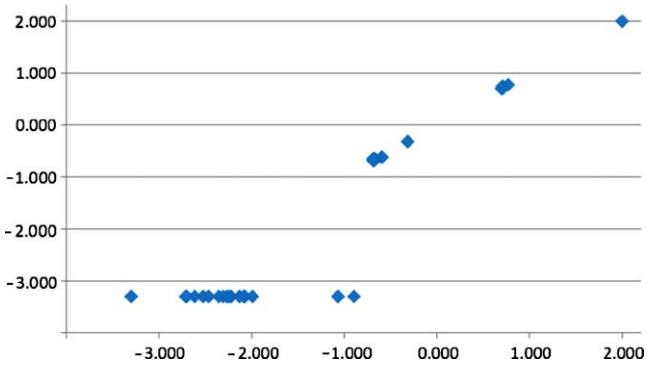

Fig. 10. Spreading activation on BBC data set (from [31]). Axes are $\log_{10}$(activation + 0.01).

Taking a long-term pure Semantic Web vision, one could argue that web developers of human-usable web services should provide parallel semantic services delivering RDF results, with some form of meta-description (e.g. WSDL or some variant of VoiD[13]) alongside, or alternatively annotate human-readable result pages with semantic markup (e.g. with microformats or RDFa). So it could be argued that third-party meta-descriptions, as used by OnCue, Snip!t and data-detectors in general, in some way run counter to this more semantic goal.

In fact, wrappers of various sorts have been around as long as the web itself, and continue today, especially in the context of the deep web [3,55,16]. This is partly an essential bootstrapping exercise: unless semantic content is sufficiently universal, then users will not rely on it, and if users do not expect it providers will not supply it; external meta-data and inference at the time of use can effectively transform the human web to semantic form and break the impasse.

Looking longer term it is likely that even in fully semantic services, ontologies will evolve or local ontologies will be used that require meta-description in the form of mappings. However, it also seems likely that many web resources will direct themselves primarily towards human readership, with semantic markup as a secondary goal. This has certainly proved the case on the Macintosh where Apple Event support in applications (at least for recording) is still at best partial despite 15 years of promotion.

In the medium term, at least, it is reasonable to assume that many different kinds of Semantic Web application will require meta-information, possibly supplied by third parties. So some form of shared repositories and shared standards for representing this meta-information is needed. There were calls for such standardisation and repositories many years ago, but at the stage when wrappers were bespoke code [55]. Now with existing XML standards and a growing Semantic Web infrastructure, it is possible to create such wrappers in declarative formats and with shared semantics given by standard ontologies. Given this the time seems ripe to revive these efforts at establishing meta-information repositories.

5.2. Ontology issues: higher order reasoning, value classes and query types

In fitting our work into formal ontologies we have encountered a number of issues.

One such complication is that humans often think in ways that are considered 'hard' or 'high-level' in ontologies, for example, the class 'Friend' in a personal ontology is effectively 'friends_of Me'. A deep theoretical challenge will be to allow some of these hard-for-machine/easy-for-human steps, but in constrained ways to prevent the full costs or semantic issues that otherwise would arise. A similar problem occurs with the spreading activation. This effectively involves following potentially any relation and, hence, requires second-order ability to quantify over relations, whereas, typically, second-order features are not natively provided by *tractable* ontology languages based on Description Logics.[14] In both cases there seem to be patterns of human-like reasoning that appear to be tractable, but cut across the standard onion-skin layers of logics.

We have also encountered interesting issues with more complex structured data inferred using the Snip!t data detector. As we saw, a postcode has 'inner' and 'outer' parts and a date includes 'year',

[12] http://www.oexchange.org/spec/.
[13] http://semanticweb.org/wiki/VoiD.
[14] http://www.w3.org/Submission/owl11-tractable/.

```
<SearchPlugin>
  <ShortName>PHP Manual</ShortName>
  <Description>PHP Manual Search</Description>
  <InputEncoding>UTF-8</InputEncoding>
  <Image width="16" height="16">
     …
  </Image>
  <Url type="application/x-suggestions+json" method="GET"
      template="http://www.php.net/manual-lookup.php?pattern={searchTerms}"/>
  <Url type="text/html" method="GET" template="http://www.php.net/manual-lookup.php">
    <Param name="pattern" value="{searchTerms}"/>
  </Url>
</SearchPlugin>
```

Fig. 11. Firefox search plugin for the PHP online manual (https://addons.mozilla.org/firefox/).

```
<vCard:N rdf:parseType="Resource">
  <vCard:Family> Crystal </vCard:Family>
  <vCard:Given>  Corky </vCard:Given>
  <vCard:Other>  Jacky </vCard:Other>
  <vCard:Prefix> Dr </vCard:Prefix>
</vCard:N>
```

Fig. 12. Example RDF for vCards (fragment) [51] note 'N' (name) property introducing a blank node.

'month' and 'day'. In the original code these were simply an associative array in the data type representing the recognised elements. On the other hand, when representing these in the personal ontology we found we needed to create entities to represent a 'person name' or a 'date', even though these represented values not things. The person referred to by the name is of course the 'thing', as is the day referred to by the date, but these are different from the name (which could refer to several people) or the date (which could be wrongly attributed to a day).

These values could have been represented as serialised strings, and then re-parsed when needed, but this seems to run counter to explicit semantics and would create additional problems, for example making it hard to search for all dates in a certain year. Perhaps the most common representation is to 'flatten' the values so that a person has individual properties 'family-name' and 'given-name', but again this obscures the fact that these are linked.

Our solution has been to label these as 'value classes' where the implication is that instances of such a value class are defined by some of their properties, and cannot be updated. This seemed to be closest to preserving the semantics and is similar to decisions made in the RDF vCard standard (see Fig. 12), or, even more generally, in common object-oriented programming best practice.

There are also some interesting data typing issues that became apparent when we looked at the inferred rules for form filling. Internally the rules are represented by path expressions through the ontology that specify which relations should be traversed. However, the meaning of the rule corresponding to the form in Fig. 5 is something like:

```
FORM (?persname, ?orgname)
  FOR SOME ?p:Person, ?i:Institution
    (?p name ?persname)
    (?i name ? orgname)
    (?c member ?i)
```

On the surface, this is similar to the pattern of a SPARQL query and its operational semantics is also similar in that the rule effectively picks out matching tuples from the personal ontology in order to select the best candidates for the 'Org' field (`orgname`) if the 'Name' field (`persname`) has been filled (or vice versa). However, note that whereas the ostensive data typing of RDF is through classes, the form rule effectively introduces what in type theory is termed a 'dependent type'. Once the 'Name' field has been prefilled, the type of the 'Org' field is not simply names of the instances of a particular class, but instead the names of the instances that satisfy a particular constraint. Dependent types have been studied extensively in mathematics, in particular in Martin-Löf intuitionistic type theory and also embedded into programming languages [59]. As a higher order construct they are usually regarded as complex computationally and semantically and yet arise naturally in the semantics of user interaction.

5.3. Linking to the desktop

Note that while Snip!t is operating on web resources, On Time is primarily operating on the desktop (though it can invoke web actions). For the user these two worlds would ideally be seamless; indeed, many users quite reasonably have a very hazy understanding of the differences, especially now that email, office documents, etc., may be accessed on the web or on the desktop with very similar interfaces.

A problem we have faced, which has also been encountered in Semantic Desktop projects, is that while resources on the web are referred to in a standard way through a URL, there is no such means to reliably and uniformly refer to desktop objects. Files can be accessed by the "file:" URI protocol, but other objects to which one might wish to refer, such as email messages or address book entries, have no similar standard scheme. Looking more closely, some applications do have their own scheme (e.g. Apple Mail's "message:" protocol), but there is no standard scheme and one cannot rely on being able to create a reference. Furthermore, the URIs generated by the "file:" and "message:" protocols are only meaningful on the machine on which they were formed. So, for example, if a file URI were stored in de.licio.us, it would dereference differently depending on the machine on which it was invoked . . . it lacks the 'U' in URI!

This problem was recognized in the Gnowsis project [65], and arising from this Sauermann has made a proposal for Desktop URIs [66]. However, because this was framed in the context of a single Semantic Desktop, like the "file:" protocol, these Desktop URIs are not usable off the target machine. The Magnet URI scheme [60] would partially address this as it allows references to resources by content-related information such as an SHA digest. However, a Magnet URI effectively references a single version of an object, so, it would no longer refer to a file if the contents change. We have made our own proposal for Globally Accessible Local URIs, using proxies to allow obfuscated (and hence) private, yet dereferenceable URIs for local resources [29]. However, this seems an important issue for web–desktop integration in general, and clearly needs community agreement, as well as retrofitting of plug-ins and adaptors for common desktop applications before new integrative applications can reliably operate across domains.

5.4. Sharing and community, risks and rewards

Designing a component-based architecture allows the integration of ready-made tasks from a variety of sources. For example, a user may download a new data detector or a ready-made action, or even exchange custom made tasks with friends. This raises issues both for the user and for service providers. Any framework that allows the distribution of custom tasks, especially when these may be invoked semi-automatically, needs to preserve the user's privacy, and to ensure data and information security—what if one downloads a task for travel booking only to find out it embezzles credit card numbers?

There is also the potential to share inferred information. The form-field inference is effectively producing meta-descriptions of web forms 'for free'. If this were shared we would have the opportunity for mass user-powered bootstrapping of Semantic Web content, without those involved having to see a line of RDF! Similarly, the results of spreading activation could be shared between friends, organizations or the world, allowing popularity to be assessed at a far finer level than is possible in traditional recommender systems. This kind of 'behind the scenes' sharing is potentially very powerful, as it requires no explicit effort except the user's, but has corresponding issues: to make sure that any shared information neither compromises individual privacy, nor has the potential for misuse such as popularity 'spamming'.

At a business level, we need to ensure that the integration and automation does not undermine the business models of the service providers on which it depends. For example, if web forms are automatically filled in and the resulting web pages parsed and only 'relevant' data extracted, this may give a better experience for the user. Then again, if this means that users are not exposed to advertisements or branding then the service providers will lose income and eventually discontinue their services.

A similar problem occurred in the early days of WAP where the telecoms operators were able to charge for WAP use through data volume or subscription payments, but WAP information providers had little opportunity to monetise unless they were explicitly selling products or services. Not surprisingly, this early WAP provision was largely limited to operators' own portals and shopping.

Of course, much software is produced *gratis*, especially micro-applications such as iPhone Widgets and FaceBook apps. However, the success of the iPhone as a platform is surely also because Apple has provided an appropriate remuneration scheme through iTunes.

6. Summary and ongoing work

This paper has presented technology targeted at producing a web of action, where our day-to-day activities not only have information at hand, but also are actively supported as activity, not merely information seeking. We have presented a number of technologies, which address different aspects of automated task support based on the categorisation of human activity in Table 1.

The personal ontology is used to enable the system to share some of the user's knowledge of the world, assisted by spreading activation to model a degree of contextual awareness. We have noted how the use of Semantic Web linked data can be used to give this a degree of 'common sense' or world knowledge in addition to the more individual knowledge of the personal ontology itself. We are, however, still looking at ways to allow the form of rapid switching between multiple contexts that we do (relatively) easily as human beings.

Semantically tagged data can be used as a locus for environmentally triggered actions. However, as much of the data encountered by users is not of this form, data detector technology can be used to effectively allow semantic annotation at the point of use. For the results pages of web applications, it is a moot point whether eventually all web applications will provide suitable annotation at the provider-side or whether some form of data detector will always be needed for non-semantic applications. There is always a tension because the cost of adding semantics is often at the provider end whereas the benefits typically accrue to the user. Certainly this is likely to be needed indefinitely for more 'light weight' content such as email messages, blog postings, and even files such as this paper.

We also saw how the use of semantics in a (web linked) personal ontology can be used not only to help users to complete web forms, but also to effectively create an automatic semantic markup. Again it is a moot point whether *all* web forms will at some point have such annotation, rendering the need for automatic markup redundant. However, the same underlying algorithms can also be used to track not just what connections are possible by matching types, but how connections are *actually used* between forms and actions.

This leads to task-sequence inference, which, as we have noted, has a long track record, although it is not seen, to date, in commonly available systems. We have seen how semantic markup could radically impact this inference, making possible much more reliable single-step learning and generalisation. However, this component has, so far, not been prototyped in our systems with the exception of single-step suggestions based on matching input types.

As well as these component technologies, we have presented two integration platforms, one, Snip!t, predominantly web-based, the other, On Time more desktop-based. This integration raises its own challenges, both architecturally and in terms of appropriate user interaction methods. In addition, in the previous section, we saw how both individual component technologies and their integration highlight various issues for Semantic-Web-based interactions. Some of these are technical, including the need for dereferenceable desktop URIs and inference/typing issues. Some require standards and sharing, in particular meta-information repositories. Finally, some are more about the human side of the web, not least the need to ensure adequate rewards for those producing both information and services.

There is a tension in our approach to integration, which aims both to be open and yet to produce a consistent user experience. However, this reflects the entire web experience, which is both unified through the browser and linking, and yet presents a smorgasbord of different page and interface styles. In contrast traditional applications strive to produce fluid consistent interface styles, yet are set amongst other applications unified only by platform style guides, which, paradoxically, those applications striving to produce the most optimal user experience are most likely to flout.

This brings us back to some of the broader issues that motivated this paper. The movement towards, at one level, more fragmented interaction, a world of mash-ups and widgets, seems not just a temporary aberration, but a necessary step. Traditional applications appear to be stagnating, with little new functionality and much apparent fragility, so that each new release brings surface changes, but as many new bugs as benefits. Indeed, ten years ago, a member of a major software company told one of the authors that the graphics engine in their core product was so ossified that it was impossible to add new interactive features. In contrast, Google Docs encourages third party graphics add-ons through Gadgets, working in the same spirit as social networking applications such as Facebook; both are prepared to sacrifice total control over user experience to gain variety and individuality in that experience—post-modern interaction?

We certainly do not have a solution let alone a complete understanding, of these issues, but the various technologies and prototypes discussed in this paper do suggest that it is possible to conceive of a future web that not only hosts global human knowledge, but also supports individual human action.

Acknowledgements

Parts of this work were supported by the Information Society Technologies (IST) Program of the European Commission as part of the DELOS Network of Excellence on Digital Libraries (Contract G038-507618). Thanks also to Emanuele Tracanna, Marco Piva, and Raffaele Giuliano for their work on On Time.

References

[1] C.R. Atkinson, M.R. Shiffrin, Human memory: a proposed system and its control processes, in: K.W. Spence, J.T. Spence (Eds.), The Psychology of Learning and Motivation, vol. 8, Academic Press, 1968.
[2] J.R. Anderson, A spreading activation theory of memory, Journal of Verbal Learning and Verbal Behaviour 22 (1983) 261–295.
[3] N. Ashish, C.A. Knoblock, Semi-automatic wrapper generation for internet information sources, in: Proceedings of the Second IFCIS International Conference on Cooperative Information Systems, COOPIS. IEEE Computer Society, Washington, DC, June 24–27, 1997, pp. 160–169.
[4] BBC Backstage, British Broadcasting Corporation, dated 2004–2005, Accessed on 6/7/2009, http://backstage.bbc.co.uk/.
[5] R. Beale, J. Finlay (Eds.), Neural Networks and Pattern Recognition in Human–Computer Interaction, Ellis Horwood, 1992.
[6] C. Bizer, T. Heath, T. Berners-Lee, Linked Data. The Story So Far, International Journal on Semantic Web and Information Systems, Special Issue on Linked Data, 5(3) (2009) 1–22. Preprint at: http://tomheath.com/papers/bizer-heath-bernerslee-ijswis-linked-data.pdf, in press.
[7] T. Berners-Lee, Looking Back, Looking Forward: The Process of Designing Things in a Very Large Space, Inaugural Lecture, Southampton University, 14 March 2007, http://www.w3.org/2007/Talks/0313-bcs-tbl/.
[8] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, in: Proceedings of Seventh ACM International World-Wide Web Conference (WWW '98), Brisbane, Australia, 1998.
[9] T. Catarci, A. Dix, A. Katifori, G. Lepouras, A. Poggi, Task-centered information management, in: C. Thanos, F. Borri, L. Candela (Eds.), First International DELOS Conference, Pisa, Italy, February 13–14, 2007, Revised Selected Papers, LNCS 4877, Springer, pp. 197–206.
[10] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: the DL-Lite family, Journal of Automated Reasoning 39 (3) (2007) 385–429.
[11] L.A. Carr, W. Hall, H.C. Davis, D.C. DeRoure, R. Hollom, The Microcosm link service and its application to the world wide web, in: Proceedings of the First WWW Conference, Geneva, 1994.
[12] T. Catarci, B. Habegger, A. Poggi, Intelligent user task oriented systems, in: Proceedings of the Second SIGIR Workshop on Personal Information Management (PIM), 2006.
[13] T. Catarci, R. Giuliano, M. Piva, A. Poggi, F. Terella, E. Tracanna, The On-TIME user interface, in: The 5th Conference of the Italian Chapter of AIS (ItAIS 2008), Paris, France, 2008.
[14] A. Clark, Being there: putting brain, in: Body and the World Together Again, MIT Press, Cambridge, MA, 1998.
[15] A.M. Collins, E.F. Loftus, A spreading-activation theory of semantic processing, Psychological Review 82 (1975) 407–425.
[16] V. Crescenzi, G. Mecca, Automatic information extraction from large websites, Journal of ACM 51, September (5) (2004) 731–779, http://doi.acm.org/10.1145/1017460.1017462.
[17] F. Crestani, Application of spreading activation techniques in information retrieval, Artificial Intelligence Review 11 (6) (1997) 453–482.
[18] A. Cypher, A, EAGER: programming repetitive tasks by example, in: S.P. Robertson, G.M. Olson, J.S. Olson (Eds.), Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Reaching Through Technology (New Orleans, Louisiana, United States, April 27–May 02, 1991), CHI'91, ACM, New York, NY, pp. 33–39, http://doi.acm.org/10.1145/108844.108850.
[19] A.K. Dey, G.D. Abowd, D. Salber, A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, Human–Computer Interaction 16 (December (2)) (2001) 97–166.
[20] D. Diaper, N. Stanton, The Handbook of Task Analysis for Human–Computer Interaction, CRC Press, 2003.
[21] A.J. Dix, Software engineering implications for formal refinement, in: C. Ghezzi, J. McDermid (Eds.), Proceedings ESEC'89, Springer–Verlag, 1989, pp. 243.259, http://www.hcibook.com/alan/papers/ESEC89/.
[22] A. Dix, Hazy, Crazy, Lazy Days are Over—Time for Designers to Think, Keynote Designing Information for Mobile and Broadband Network Users, London, 15 December 1999, http://www.hcibook.com/alan/papers/mobile-and-b-99/.
[23] A. Dix, R. Beale, A. Wood, Architectures to make simple visualisations using simple systems, in: Proceedings of Advanced Visual Interfaces (AVI 2000), ACM Press, 2000, pp. 51–60.
[24] A. Dix, T. Rodden, N. Davies, J. Trevor, A. Friday, K. Palfreyman, Exploiting space and location as a design framework for interactive mobile systems, ACM Transactions on Computer–Human Interactions 7, September (3) (2000) 285–321, http://doi.acm.org/10.1145/355324.355325.
[25] A. Dix, J. Finlay, G.D. Abowd, R. Beale, Human–Computer Interaction, third ed., Prentice Hall, 2004.
[26] A. Dix, The brain and the web: intelligent interactions from the desktop to the world, in: Proceedings of VII Brazilian Symposium on Human Factors in Computing Systems, IHC'06, vol. 323, 2006, p. 142, doi:10.1145/1298023.1298080.
[27] A. Dix, T. Catarci, B. Habegger, Y. Loannidis, A. Kamaruddin, A. Katifori, G. Lepouras, A. Poggi, D. Ramduny-Ellis, Intelligent context-sensitive interactions on desktop and the web, in: Proceedings of the International Workshop in Conjunction with AVI 2006 on Context in Advanced Interfaces (Venice, Italy, May 23–23, 2006), CAI'06, ACM, New York, NY, pp. 23–27, http://doi.acm.org/10.1145/1145706.1145710.
[28] A. Dix, A. Katifori, A. Poggi, T. Catarci, Y. Ioannidis, G. Lepouras, M. Mora, From information to interaction: in pursuit of task-centred information management, in: Proceedings of DELOS Conference 2007, Pisa, Italy, 2007.
[29] A. Dix, A. Katifori, G. Lepouras, Globally Accessible Local URIs—Discussion Ideas, 2008, http://www.hcibook.com/alan/projects/TIM/docs/gloi/.
[30] A. Dix, Tasks = Data + Action + Context: Automated Task Assistance through Data-Oriented Analysis, in: P. Forbrig, F. Paternò (Eds.), Proceedings of the 2nd Conference on Human-Centered Software Engineering and 7th International Workshop on Task Models and Diagrams (Pisa, Italy, September 25–26, 2008), Lecture Notes in Computer Science, vol. 5247, Springer–Verlag, Berlin, Heidelberg, pp. 1–13, http://dx.doi.org/10.1007/978-3-540-85992-5_1.
[31] Alan Dix, Akrivi Katifori, Giorgos Lepouras, Costas Vassilakis. Spreading activation over ontology-based resources: from personal context to web scale reasoning, International Journal of Semantic Computing, Special issue on web scale reasoning: scalable, tolerant and dynamic, in press.
[32] L. Dodds, Understanding the Big BBC Graph, dated 11 June 2009, Accessed on 10/08/2009, http://blogs.talis.com/n2/archives/569.
[33] The Dojo Foundation, The Book of Dojo, 1.3, http://docs.dojocampus.org/.
[34] M. Dzbor, E. Mottaa, J. Dominguea, Magpie: experiences in supporting semantic web browsing, Web Semantics: Science, Services and Agents on the World Wide Web 5 (3) (2007) 204–222.
[35] M. Endsley, Toward a theory of situation awareness in dynamic systems, Human Factors 37 (1) (1995) 32–64.
[36] K. Ericsson, W. Kintsch, Long-term working memory, Psychological Review 102 (1995) 211–245.
[37] A. Faaborg, H. Lieberman, A goal-oriented web browser, in: Proceedings of the Conference on Human Factors in Computing Systems (CHI 2006), ACM Press, 2006, pp. 751–760.
[38] D. Fallman, B. Yttergren, Meeting in quiet: choosing suitable notification modalities for mobile phones, in: Proceedings of the 2005 Conference on Designing For User Experience (San Francisco, California, November 03–05, 2005). Designing For User Experiences, vol. 135, AIGA: American Institute of Graphic Arts, New York, NY, p. 55.
[39] W. Hall, H. Davis, G. Hutchings, Rethinking Hypermedia: The Microcosm Approach, Kluwer Academic Publishers, Norwell, MA, 1996.
[40] M. Hartmann, D. Schreiber, Prediction Algorithms for User Actions. Workshop on "Lernen, Wissen und Adaptivität" (Learning, Knowledge, and Adaptability), LWA 2007, German Society for Informatics, pp. 349–354.
[41] M. Hasan, A spreading activation framework for ontology-enhanced adaptive information access within organisations, in: Proceedings of the Spring Symposium on Agent Mediated Knowledge Management (AMKM 2003), Stanford University, California, USA, 2003.
[42] M. Hausenblas, Having your profile forms automatic filled in ... Blog post dated 5th April 2009, http://webofdata.wordpress.com/2009/04/05/pac-intro/.
[43] Tom Heath, Martin Dzbor, Enrico Motta, Supporting user tasks and context: challenges for semantic web research, in: Proceedings of the Workshop on End-user Aspects of the Semantic Web (UserSWeb), European Semantic Web Conference (ESWC2005), Heraklion, Crete, 2005, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-137/11_heath_final.pdf.
[44] Tom Heath, Enrico Motta, Martin Dzbor, Context as foundation for a semantic desktop, in: Proceedings of the 1st Workshop on The Semantic Desktop, International Semantic Web Conference (ISWC2005), Galway, Ireland, 2005, http://tomheath.com/papers/heath-motta-dzbor-semdesk2005-context-semantic-desktop.pdf.
[45] T. Heath, E. Motta, Revyu: linking reviews and ratings into the web of data, Journal of Web Semantics 6 (4) (2008).
[46] E.L. Hutchins, J.D. Hollan, D.A. Norman, Direct manipulation interfaces, Hum–Computer Interaction 1 (December (4)) (1985) 311–338.
[47] J. Hollan, E. Hutchins, D. Kirsh, Distributed cognition: toward a new foundation for human–computer interaction research, ACM Transactions on Computer–Human Interaction 7 (2) (2000) 174–196.
[48] Hussein, Tim, Ziegler, Jürgen: adapting web sites by spreading activation in ontologies, in: ReColl'08: International Workshop on Recommendation and Collaboration (in conjunction with IUI 2008) Gran Canaria, 2008.
[49] E. Hutchins, Understanding Micronesian navigation, in: D. Gentner, A. Stevens (Eds.), Mental Models, Lawrence Erlbaum, Hillsdale, NJ, 1983, pp. 191–225.
[50] E. Hutchins, Cognition in the Wild, MIT Press, 1995.
[51] Renato Iannella, Representing vCard Objects in RDF/XML, W3C Note, 22 February 2001, http://www.w3.org/TR/vcard-rdf.
[52] W. Jones, J. Teevan (Eds.), Personal Information Management, University of Washington Press, 2007.
[53] A. Katifori, C. Vassilakis, I. Daradimos, G. Lepouras, Y. Ioannidis, A. Dix, A. Poggi, T. Catarci, Personal ontology creation and visualization for a personal interaction management system, in: Proceedings of PIM Workshop, CHI 2008, Florence, Italy, 2008.

[54] A. Katifori, C. Vassilakis, A. Dix, Ontologies and the brain: using spreading activation through ontologies to support personal interaction, Cognitive Systems Research 11 (1) (2010) 25–41.
[55] N. Kushmerick, (Toward) an extensible wrapper repository standard, Workshop on AI & Information Integration, AAAI-98, Madison, 1998.
[56] G. Lepouras, A. Dix, A. Katifori, T. Catarci, B. Habegger, A. Poggi, Y. Ioannidis, OntoPIM: from personal information management to task information management personal information management, SIGIR 2006 Workshop, Seattle, Washington, August 10–11, 2006.
[57] H. Lieberman, Your Wish is my Command: Programming by Example, Morgan Kaufmann, San Francisco, 2001.
[58] W. Liu, A. Weichselbraun, A. Scharl, E. Chang, Semi-automatic ontology extension using spreading activation, Journal of Universal Knowledge Management 0 (1) (2005) 50–58.
[59] C. McBride, J. McKinna, The view from the left, Journal of Functional Programming 14 (January (1)) (2004) 69–111.
[60] G. Mohr, MAGNET v0.1, Created on 2002-06-12, Revised on 2002-06-17, Accessed on June 2008, http://magnet-uri.sourceforge.net/magnet-draft-overview.txt.
[61] B. Nardi, J. Miller, D. Wright, Collaborative, programmable intelligent agents, Communications of the ACM 41 (3) (1998) 96–104.
[62] M. Pandit, S. Kalbag, The selection recognition agent: instant access to relevant information and operations, in: Proceedings of Intelligent User Interfaces (IUI, 97), ACM Press, 1997, pp. 47–52.
[63] P. Resnick, H.R. Varian (Eds.), Special Issue on Recommender Systems, Communications of the ACM, 40 (3) (1997) 56–89.
[64] E. Rukzio, A. Schmidt, H. Hussmann, Privacy-enhanced intelligent automatic form filling for context-aware services on mobile devices, in: Proceedings of Workshop on Artificial Intelligence in Mobile Systems 2004 (AIMS 2004), in Conjunction with UbiComp 2004, Nottingham, UK, 2004.
[65] L. Sauermann, The Gnowsis semantic desktop for information integration, in: Proceedings of the 3rd Conference Professional Knowledge Management, Kaiserslautern, Germany, 2005.
[66] L. Sauermann (Ed.). Desktop URIs (accessed on May 2008), http://aperture.wiki.sourceforge.net/SemdeskUris.
[67] B. Shneiderman, The future of interactive systems and the emergence of direct manipulation, Behavior and Information Technology 1 (1982) 237–256.
[68] L. Sauermann, D. Heim, Evaluating long-term use of the Gnowsis semantic desktop for PIM, in: Proceedings of the 7th International Conference on the Semantic Web, Karlsruhe, Germany, 2008, pp. 467–482.
[69] J. Stylos, B. Myers, A. Faulring, Citrine: providing intelligent copy-and-paste, in: Proceedings of the 17th Symposium on User Interface Software and Technology (UIST 2004), ACM Press, 2004, pp. 185–188.
[70] L. Suchman, Plans and Situated Action, Cambridge University Press, MA, 1987.
[71] L. Swartz, Why people hate the paperclip: labels, appearance, behavior, and social responses to user interface agents, Honors Thesis, Symbolic Systems Program, Stanford University, 2003.
[72] A. Wood, A. Dey, G. Abowd, Cyberdesk: automated integration of desktop and network services, in: Proceedings of the Conference on Human Factors in Computing Systems (CHI 97), ACM Press, 1997, pp. 552–553.
[73] P.C. Wright, A.F. Monk, A cost-effective evaluation method for use by designers, International Journal of Man–Machine Studies 35 (1989).
[74] W3C draft, Client Side Automated Form Entry, W3C Working Draft WD-form-filling-960416, http://www.w3.org/TR/WD-form-filling.html.
[75] W3C Editor's Draft, Web Storage dated 4 February 2010, http://dev.w3.org/html5/webdatabase/.
[76] W3C Editor's Draft, Web SQL Database dated 4 February 2010, http://dev.w3.org/html5/webdatabase/.