SerenA: A multi-site pervasive agent environment that supports serendipitous discovery in research

{Anonymous for review}

{Anonymous for review}

Abstract. We present SerenA, a multi-site, pervasive, agent environment that suppers serendipitous discovery in research. The project starts from the premise that human users cannot be aware of all the research information that is relevant to their work, because of the compartmentalisation of research into fields around particular journals, and, simply, because there is too much to know. In particular, the Semantic Web provides a resource which can assist, but there is more to be discovered than the things that a user might deliberately search for. SerenA, then, attempts to assist researchers by presenting them with information that they did not know they needed to know about their research.

Keywords: Serendipity; semantic web; pervasive agents

1 Introduction

We describe SerenA, an agent-based, semantic, pervasive, embedded personal assistant system, for academic researchers. SerenA is designed in response to the {Anonymous for review}, and is, in the first instance, intended for arts and humanities researchers.

The key idea of SerenA is to create a Serendipity Arena: a virtual space in which serendipitous discovery of several different kinds is more likely to happen than elsewhere. For example, two SerenA users with common research interests might be travelling to the same place, but not be aware of the fact; SerenA would alert them and, if its users desired, arrange a discussion meeting. However, the aim is not merely to create an academic dating service; rather, the idea extends to objects in the world too, so that, for example, a researcher arriving at King's Cross station in London could be informed that a unique manuscript, of direct interest to their work, and borrowed temporarily from Egypt, is on show in a nearby museum on that day; what is more, they could be told this in advance, when they book their train ticket on line, so they can plan for a visit. Above all, SerenA is intended to find for its users things that they did not know they wanted.

These superficially simple matching tasks involve significant background reasoning, about the users' interests and activities, and about real-world data, such

as location and time, both current and planned. They require a substantial software infrastructure that is capable of travelling with the user, but also supplying substantially more computational power than a standard mobile device. It must also be capable of semantic analysis of a users writings and actions, if not in real time, at least in a time frame that is practically useful, depending on context (consider the two examples above: the temporal requirements of the corresponding notifications are quite different). SerenA's target users cannot be assumed to be expert computer users, and therefore it must work with them in ways that are both easy to use and engaging.

Thus, under detailed scrutiny, the superficially simple idea of SerenA quickly expands from an engaging and useful tool to an archetypal general AI problem, including language understanding, proactive semantic reasoning, intelligent interaction and pervasive presence. The only factor to reduce the challenge is the nature of serendipitous discovery itself: unusually in computer science, and only within reason, the system can be *usefully ambiguous or wrong*. This is so because, ultimately, SerenA forms a hybrid system with its users, and unexpected, off-the-wall information from the computational part of the hybrid can nevertheless usefully inform or stimulate the human part. Therefore, SerenA has reduced responsibility for correct reasoning: its suggestions need only be useful; they do not have to be logically correct.

The funded SerenA project, as a whole, includes designers, HCI and usability specialists, and computer scientists, working together in a broad coalition. It has considered the epistemology of serendipity [Anon], the meaning of the concept of serendipity to researchers, and their reactions to it [Anon], and approaches to concept extraction from text [Anon]. Also work has been done on language processing with a view to discovering users' interests from their tweets [Anon] and their goals from their notes and email messages [Anon]. The current paper, though, takes a top-down perspective, explaining the overall conceptualisation of the system, explaining how the primary challenges are met, and in particular focusing on the agent system that forms the core of the pervasive environment.

With this aim in mind, the rest of the paper describes a selection of the functions of SerenA, from a user's perspective, identifying the engineering challenges, and outlines how they are integrated by the core agent system around which SerenA revolves. There is not space here to describe every aspect of SerenA in detail. Rather, we aim to give a flavour its capabilities, and to demonstrate how the agent system contributes, on multiple levels, to the elegant implementation of the whole.

2 Affordances and Constraints

We now describe SerenA from the user's point of view, explaining how the various affordances of the software affect its design. We identify the points at which the agent approach is particularly useful.

2.1 Ubiquitous intelligent personal support

SerenA is conceived as a ubiquitous computing environment, which should interact with its users via their mobile devices and desktop computers, and via public installations in public spaces. The requirements of these three *interactors* are somewhat different. The first two are *private* to the user, and authenticated, while the third is *public*: this difference impinges on the nature of the information that can be displayed. Mobile devices tend to have significantly smaller displays than desktop machines, and a public display is more likely to be large than small, so this dimension distinguishes the first category of interactor from the second and third. In the SerenA project so far, we have focused on the mobile private SerenA, with only one proof-of-concept public SerenA interactor currently planned. These are described in §2.2.

SerenA's ubiquity, delivered through mobile devices, imposes constraints on its design, which agent-based designs are well suited to meet. First, the behaviour of the system must be consistent and persistent. That is to say, an interactor must behave uniformly, and must not lose information if, for example, it is switched off, or (more likely) if there is a network outage. A very neat way of addressing this issue is to maintain an agent that simulates the mobile device, and then to implement synchronisation of information between the mobile device and its agent, which can be done independently of the workings of the rest of the agent system, and of the user's interaction with the device[1]. We term this kind of agent, that echoes the behaviour of an entity in the physical world, a *shadow* agent[2]. Shadow agents of other kinds appear too, the most important being the user agency, a group of agents that shadow the user, supplying information about the user (e.g., location, research interests, privacy settings) to the rest of the system, but also autonomously acting on their behalf to make serendipitous discoveries. The user agency is described in §3.2.

A key advantage of the shadowing approach is that it allows us to meet SerenA's requirements for high-power computing and network access in static installations with high-powered servers with fast access to the Internet, both of which are *sine qua non* for the deep inference required of SerenA if it is to be helpful to its users. A further concomitant advantage is the stability and security of such managed systems. The agent community, running on these servers, can work uninterrupted, communicating internally with the shadows of the physical world interfaces, and those interfaces can be updated live when connected, or asynchronously when an interactor reappears on the network after being disconnected. What is more, the asynchronous nature of the agent community absolves SerenA's implementers of the need to manage the appearance and disappearance of interactors, and also the delays inherent in web services and sites; agents com-

[1] Of course, the user's interaction with the device as a whole may be restricted by a network outage (as the Internet becomes temporarily unavailable), but this is not a soluble problem.

[2] The term is borrowed from the UK government system, where the official Opposition party forms its own cabinet, *shadowing* the ministerial functions (and dysfunctions) of the elected government.

municate via message passing, and, simply, when they are disconnected, or when there is a delay, no external messages messages appear (though of course the corresponding shadow agent may continue working independently on the basis of prior information). A final bonus of shadowing is in the use of shadow agents to represent external internet resources within the agent community. This means that issues of translating between languages and formats need not be spread throughout the system, but can be handled by a specialist shadow, so that the information is manipulated into a SerenA-friendly form, exactly once, as soon as it enters the system. Our use of FIPA ACL standards[3] means that responses can be straightforwardly associated with their corresponding queries, on receipt, using conversation management.

It is in the nature of agent systems that they are conceptually distributed, though this is not always the case in terms of implementation. SerenA is implemented using the Java Agent DEvelopment Framework[4] [1], which supports two important SerenA requirements. In JADE, agents run in notional *containers*, one or more containers per machine, but multiple machines can be connected together into a JADE *platform*. This allows failover to be implemented at the agent level: duplicate containers of SerenA agents can be run at multiple sites, and one fails, another can take over.

2.2 Interactors

Private SerenA

One of SerenA's design constraints is that its users should not have to learn to use specialist interfaces and equipment; rather, it should work invisibly behind familiar interaction paradigms, making itself noticed only when a direct user response is required, or when the user's attention needs to be drawn to SerenA output.

To this end, we have designed an interface, which is the current focus of our interaction work, as an Android app, conceptualised as a *Semantic*[5] *Sketchbook* [Anon]. The researcher-user is invited to make notes, add tagged images, keywords, and so on, all in free text. The notes can be organised in a predictable but useful way. Interaction with the user is then managed by SerenA processing the user's text, and then adding annotations (e.g., items of text, web links), to the notes, making the distinction between the user's notes and SerenA's additions clear by means of typography. Example views are given in Figure 1. When SerenA adds a new annotation, the Android notification system informs the user, according to their preferences. The user can then follow up the suggestions at their leisure, or delete them if they wish. In some cases, time-sensitive notifications will fade away when they become stale, though a user might indicate that they should be retained, and override this.

[3] `http://www.fipa.org/specs/fipa00061/SC00061G.html`

[4] JADE; `http://jade.tilab.com`. It seems likely that Erlang (`www.erlang.org`) is a future candidate for such implementations.

[5] This epithet is justified further in §3.

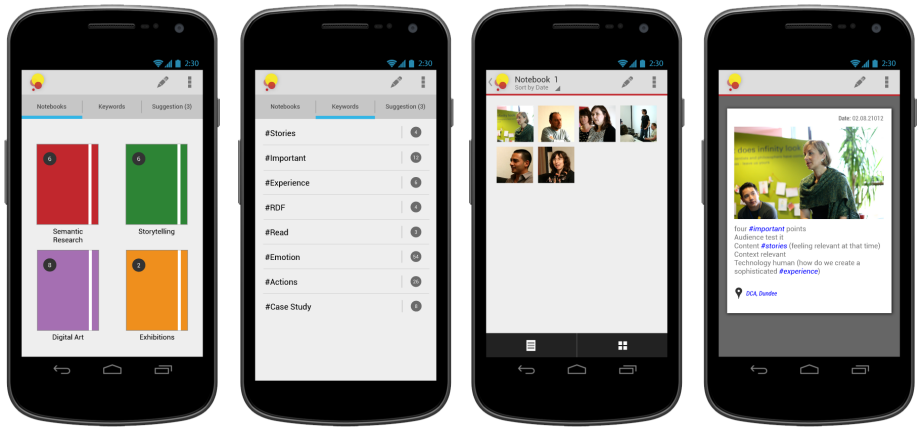

Fig. 1. The Semantic Sketchbook. From left to right: Notebook view (creating a new notebook and personalising the cover); Goal list (keywords and goals); Sort By: Visual and date; Keywords and Multimodal notes with Goals (Tags in the body of the text).

This kind of interaction is advantageous for two reasons: first, SerenA must avoid the paperclip effect[6]; and, second, the reasoning required to avoid the presentation of pointless information is often extensive (see §3), and it cannot be performed on the fly, while the user waits. The concomitant advantage of asynchrony is that network outages do not degrade the experience: the user will come to expect SerenA suggestions at some time after they make their notes, but not immediately.

Public SerenA

The criteria for a public instantiation of SerenA are quite different. First, there are significant security issues concerned with a user's personal or private information: in the current prototype, we address this simply by using information that we know to be public. There are major open opportunities in a public installation of this kind: it is not merely intended to be a terminal that individuals can use to access private SerenA, but something altogether more collective. One key affordance, given the right social context, is the ability to communicate with more than one user at once via the same channel. For example, at an academic conference, one might display connections between consenting delegates, in terms of the connections that SerenA has found between their work and interests: since SerenA is focused on finding *unexpected* connections, this approach might be expected to add value to the social interaction at the event.

Our first prototype public SerenA is conceived in the context of a major UK city library. It is architecturally melded with the building, in that its outputs are projected directly on to walls, using site-specific designs that integrate with the

[6] The irritation produced when Microsoft's Clippy character used to intrude unexpectedly, distracting the user from their task, with often incorrect information.

architecture. Its outputs consist of simple visualisations (the simplest being mere text) of documents that are ordered via the library's on-line order system. The information is filtered, so that no connection with the library user ordering it can be made: their name is not displayed, nor is the time at which they ordered it. In principle, however, interesting work could be done finding connections between documents ordered by different people, and displaying on public SerenA a summary of these: a sort of local, temporary Zeitgeist analysis. An example design is shown in Figure 2.

Communication with the Agent System
SerenA's interactors communicate with the Agent System via shadow agents as outlined in §2.1. Externally, the connection from the shadow to the mobile device is implemented directly in Java, running as a background process in Android, and connecting to the shadow via a persistent WebSocket[7,8]. This process then communicates with front end UI managers, such as the Semantic Sketchbook. Necessarily, some local storage of information (such as what is on the current display, what is in the notebook, both from user and from SerenA, and what has or has not been synchronised with the shadow agent) must be managed, and this is done using a local database, working in the same language as the agent system (see §3); other issues of synchrony are dealt with at the level of WebSockets or below, in TCP/IP.

3 Knowledge Representation and Inference to support Serendipity

3.1 Formalism: The Semantic Web

In order to make interesting and unexpected connections, SerenA explores and combines information from many different sources, using the growing array of Semantic Web resources. Increasing amounts of information from many different domains are being made available as Linked Open Data (LOD). LOD uses syntactic and semantic standards such as The Resource Description Framework[9] (RDF) and OWL[10] (the Web Ontology Language), and is available for query across the web. SerenA also uses Semantic Web ontologies currently being developed and integrated to express information in specific domains, such as FOAF[11] to describe people and relationships, DBpedia[12] for general knowledge, GeoNames[13] for geographic locations and DBLP[14] and Dublin Core[15] for publi-

[7] http://www.w3.org/TR/websockets/
[8] http://tools.ietf.org/html/rfc6455
[9] http://www.w3.org/RDF/
[10] http://www.w3.org/TR/owl2-overview/
[11] http://www.foaf-project.org
[12] http://dbpedia.org
[13] http://www.geonames.org
[14] http://dblp.uni-trier.de
[15] http://dublincore.org/documents/dc-rdf/index.shtml

a)

b)

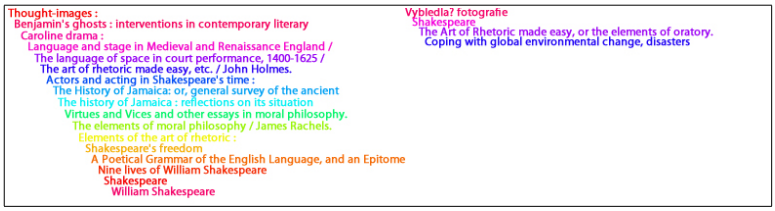

Fig. 2. An example design for one instantiation of public SerenA. Books being ordered from a library are presented as part of the architectural structure of the library in near-real time. a) The live display. b) Direct view of the information displayed.

cations. These ontologies and resources are being realised in individual web-accessible databases which can be searched by tools such as Sindice[16] or merged into larger databases of machine-readable information such as FactForge[17]. As a result of these initiatives it is now possible to combine information from many different sources, at a general, domain-independent level, to link location data obtained from a mobile device to information about nearby places of interest, for example.

SerenA's domain knowledge and agent control commands are represented in RDF, a good choice for expressing highly structured knowledge-based information; together with the OWL-DL subset of OWL, this affords a Description Logic [2, ch. 9], a well-understood basis for knowledge representation and inference. SerenA's agents communicate in RDF, also, and its message envelopes meet the FIPA Agent Communication Language specification. This approach eliminates the need to translate domain knowledge acquired from the Semantic Web to internal agent knowledge representations. It can also support multiple levels of agent reasoning.

Because SerenA agents are themselves described in RDF, the possibility is open to build agents that reflect on and modify the behaviour of others. However, this is currently deferred to future work.

3.2 User modelling: Agent.Me

Over time, SerenA builds a model of its user, expressed in RDF, including information given by the user, information inferred directly under the control of the user, and information inferred about the user by the system. This information is made available to a collection of agents, which act as the user's shadow in the agent community. The modelling process is kick-started by our Discover.Me.Semantically service[18], which searches for information about a new user, and then consults with them to select what is relevant.

Discover.Me.Semantically is a web-based tool that allows its user to author RDF representing their professional and personal interests, skills and expertise. The stand-alone implementation allows the user to download this RDF representation as a file to be hosted on their own web pages[19].

The standalone implementation (which the reader is invited to try) also offers a path to visualize this RDF on a linked-open-data visualization called LodLive[20], to explore the paths along links away from their skills and interests. The tool knows of several Web resources, and records some aspects of equivalence between them, and these *sameAs* links can also be explored. The RDF representation of the user so generated can also be stored in SerenA's user model, and

[16] `http://sindice.com`
[17] `factforge.net`
[18] Source code under GPLv3 license: {Anonymous for review}. Running instance: {Anonymous for review}
[19] Having a `foaf.rdf` file attached to one's academic webpage is becoming commonplace for Semantic Web based researchers.
[20] `http://lodlive.it/`

then be used by a user's shadow, or Agent.Me to assist in finding information of interest. Other readily available information sources also contribute to the user model, such as bibliographical information, obtained from a user's BibTeX or EndNote file. In the longer term, we intend to repurpose ideas from Intelligent Tutoring Systems, in which the computer's model of a user is made visible to the user [3], so that the user can reflect on it, and correct it if necessary.

3.3 Supporting Serendipity with Inference: Goal Detection

It is in the nature of serendipity to be unpredictable: if one could create the effect to order, it would not be serendipity, by definition. Our aim, instead, is to enhance the conditions in which serendipity might take place. Ultimately, this process will be managed by the agent system, with some agents requesting information to give to the user, and others searching for answers according to the competence of the resource they are shadowing. The asynchronous, open nature of the agent system, and also of our interface method (see §2.2) mean that relatively little overhead needs to be expended on simple question-answer interaction. More interesting reasoning, however, can be carried out by generalised reasoning agents, but this is future work.

A key issue in supplying the user with useful information is to understand the research goals that they are expressing in their notes, files and email[21]. We have begun work with the GATE natural language processing system[22], with some success in detecting goals in natural language sentences [Anon], and an ontology for goals has been defined [Anon]. These ideas will inform the Agent.Me.

An example of these ideas in action can be found in our case study of connecting users within one institution. For this, we use information gathered for the UK Research Excellence Framework[23]. The information is about publications of every academic in the institution (coverage is not universal, in fact, but this does not prevent the system from working), and about the academics themselves. This information is available in RDF format, delivered via a web service. Also included, for many publications, is the plain text of the abstract.

Our approach is to extract semantic annotations from titles and abstracts, using the OpenCalais web service[24]. Once this is done, we run semantic web queries to deduce answers to such leading questions as:

- Which people in different schools (who therefore may not know each other) describe the same concepts in their papers?
- Are there more experienced specialists (e.g., professors) who often publish papers on concepts also of interest to (e.g.) early-career research associates?

Perhaps unsurprisingly, these queries tend to produce many answers, and we are working on heuristics to filter them in a useful way, with respect to promoting serendipity.

[21] Of course, SerenA does not access files or email without permission.
[22] `http://gate.ac.uk`
[23] `http://www.ref.ac.uk/`
[24] `http://www.opencalais.com`

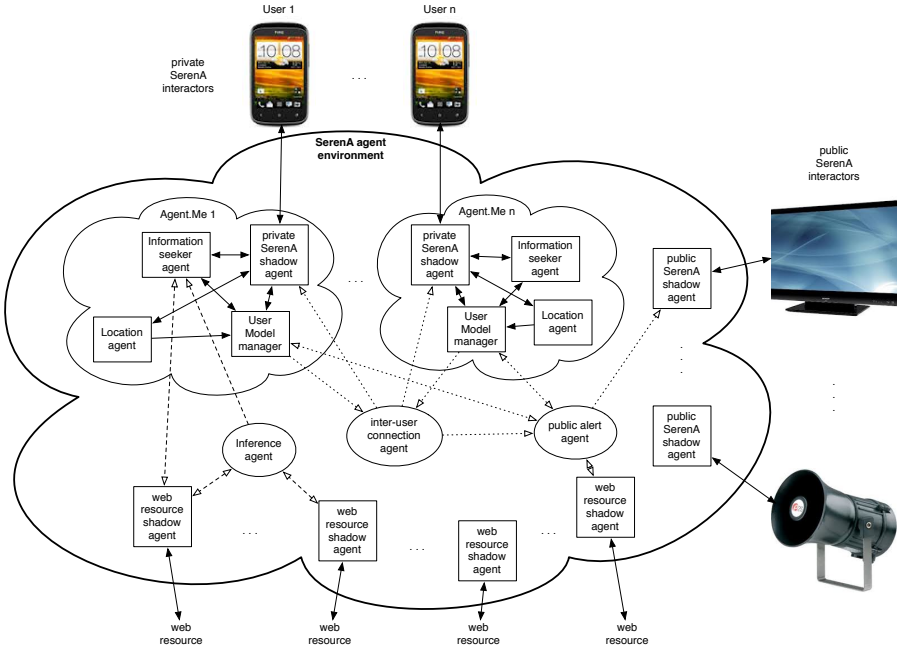

Fig. 3. Overview of SerenA, including two potential agent conversations. Solid arrows indicated on-going administrative information flow. Dashed arrows indicate information flow resulting from the addition of a new note to the Semantic Sketchbook, running on User 1's phone. Dotted arrows indicate information flow resulting from the arrival of two users with common interests in a location with a public SerenA installation. See §4 for details.

4 System Overview and Example

Figure 3 illustrates the overall system architecture with some example agent types, and also shows the potential communication between agents in two specific tasks. These are: information flow resulting from the addition of a new note to the Semantic Sketchbook (dashed arrows); and information flow resulting from the arrival of two users with common interests in a location with a public SerenA installation (dotted arrows).

New Sketchbook Annotations
When User 1's device shadow agent announces to User 1's Agent.Me that a new note has been added to the user's Semantic Sketchbook, the associated information seeker agent broadcasts a request for information on the tags included in the note. Some time later, a web resource agent finds a semantic web resource that referring to some of the same concepts. The resource is returned to the

user's interactor via its shadow, and presented in the Sketchbook as a title with a clickable link.

The same broadcast request also reaches an inference agent, whose speciality is to make connections between SerenA users who publish on related concepts in different research fields. This agent can search for publications in DBLP, by concept; it then broadcasts a request to all Agent.Mes to ask for the research fields and institutions of SerenA's users whose papers it finds. It then returns an answer to User 1's Agent.Me listing the papers of those users in their institution, who publish on the concepts of interest but in different research fields.

Arrivals and meetings
A more complicated example arises when two SerenA users who have enabled their location agents arrive at the same conference, where there is a public SerenA interactor. The users' location agents broadcast their arrival at the event (which their Agent.Mes infer is significant from their diary entries), and the inter-user connection agent notices their physical coincidence. It queries their respective user models, and learns that they are both interested in being introduced to other SerenA users. The inter-user connection agent broadcasts the opportunity to meet, which is relayed to the users' interactors by their Agent.Mes, but is also picked up by the local public alert agent, associated with the public SerenA installation. This agent checks with both users that they allow their images to be used on public SerenA, and, if they do, tells the shadow agent to announce their presence and introduce them to each other.

5 Conclusion and Future Work

In a paper of this length, it is not possible to cover all the aspects of a system design as large and diverse as that of SerenA. Here, we have attempted to convey the basic ideas and *raisons d'être* of the system, and to give enough detail to explain the contribution of the agent framework and communication style, and the resulting interaction with users. We believe this to be a novel contribution to practical applications of multi-agent technology, in that to our knowledge, a distributed agent system of this scale has not previously been deployed.

There is, it is clear, a substantial amount of work to do before we can claim that SerenA has fulfilled its potential—although formal evaluation of various aspects of the work outlined in this paper is in progress, there are many more possibilities for emergent behaviour that have yet to be enabled. Detecting and enhancing such behaviours will be the focus of future work.

Acknowledgements

This work is supported by {Anonymous for review}.

References

1. F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi. Jade — a Java agent development framework. In R. Bordini, M. Dastani, J. Dix, and A. Fallah Seghrouchni, editors, *Multi-Agent Programming*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, pages 125–147. Springer US, 2005.
2. R. J. Brachman and H. J. Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufmann, 1985.
3. S. Bull, V. Dimitrova, and G. McCalla. Open learner models: Research questions. *International Journal of Artificial Intelligence in Education*, 17(2):83–87, 2007.