A personal agent-based approach for API evolution

Cristian Vasquez

IDLab, Department of Electronics and Information Systems, Ghent University – imec,
cristian.vasquezpaulus@ugent.be

Abstract. Personal agents and APIs can enable personalized applications that facilitate the daily processes that a person uses to gather, classify, persist and retrieve information in daily activities. General-purpose APIs will cover part of the functionality, but others need higher degrees of customization. This thesis focus on systems where to count with common models is difficult, and investigate a bottom-up approach where personal agents expose hypermedia APIs and copy, transform and combine API features from others, as a mechanism to improve their means of interaction.

Keywords: agents, hypermedia, emergent semantics, semantic web service discovery, personal dataspace

1 Introduction

We refer to a personal dataspace as the grouping of all kind of digital information that is gathered and stored by an individual over long periods of time, under that person's control (but not necessarily exclusively so)[1].

In the last years, several decentralized social networks have been proposed [2], with applications can access a unified, user-controlled dataspace. In such a decentralized setup, one would want intelligent agents to assist users in their activities while having services tailored to their needs.

Furthermore, the agents might interact with other agents through APIs, requiring (i) syntactical interoperability, i.e., establish the basic symbols to exchange a representation without losing structure or information, and (ii) semantic interoperability, i.e., reducing interpretation gaps to trigger acceptable behaviors through interactions.

Interoperability is traditionally achieved when peers can refer to a common reference model or ontology but become increasingly difficult in decentralized scenarios. The representations agents exchange, depends on how their schemes are aligned, but in some cases, direct mappings to common models are not available. In these setups, an agent would prefer to gather, infer or compose

ad-hoc mappings during interaction with their peers, to interpret, transform or negotiate structured content. Also, in a growing ecosystem of applications that are separated from storage, we can expect a significant number of models changing over time. In this context, provenance is desired to help to interpret the original intended meaning in the long term.

This work focuses on cases where to reach global interoperability is difficult, and investigate an approach where APIs are decomposed in features to be exchanged in a network utilizing clone, mutation, divergence and convergence mechanisms while persisting `provenance trails` of the API evolution. The utility of such approach is three-fold: (i) to promote API evolution through agents that `collect` API functionality from others, (ii) to generate useful artifacts for service discovery (iii) and to incrementally build mappings between the underlying models of each agent, to improve their means of interaction.

2 Relevancy

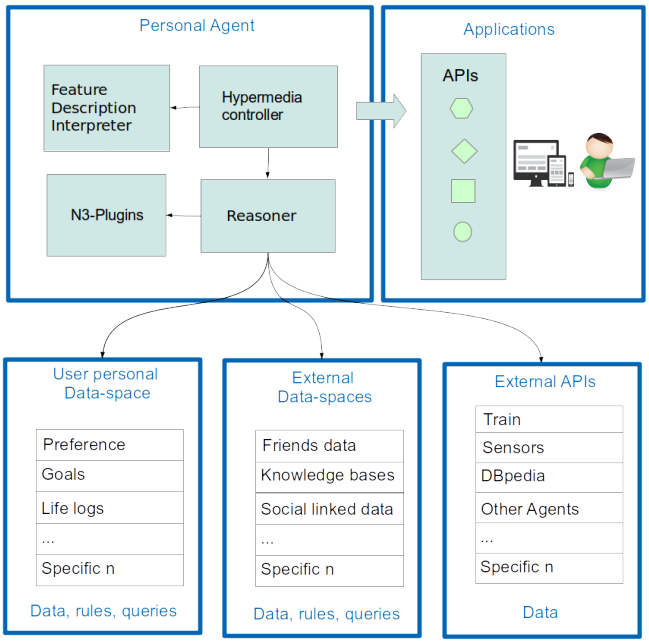

Fig. 1. Personal agent exposing APIs generated by means of reasoning and resources in the web

There is a growing interest in APIs that adapt to the data of a user, as they show significant improvements over fixed designs according to a specific user profile.

One use case is health-care applications, where care plans need to be tailored to medical patient profiles and coordinated across several care providers. A centralized approach has drawbacks since is difficult to acquire and continuously update the user models required to provide sophisticated tailoring [3]. One approach is to have applications that consume APIs exposed by a personal agent with access to a personal dataspace and external sources. This was the case of the GPS4IntegratedCare[1] project, which developed a Smart Workflow Engine to suggest and execute care paths tailored to complex medical patient profiles. One of the outputs was an agent designed according to the following requirements: (i) simplicity of a uniform interface (ii) visibility of features by other service agents (iii) portability of components by moving from program code to data.

The result was a modular design that enables building APIs as a stock of building blocks or `features` to add to the agent dataspace. A feature is defined by patterns that define what RDF triples to use as data, which N3 documents to use as rules and a query. The approach extensively applies hypermedia notions to Semantic Web Services and will be further explored in this work.

3 Related work

To create the proposed system I will use existing research from two fields, which I categorize as Hypermedia-driven APIs and Bottom-up Interoperability Elicitation.

3.1 Hypermedia-driven APIs

Extensive research has been done regarding the discovery, composition, and orchestration of Semantic Web Services [4] which traditionally makes use of central registries where developers advertise their services. More recent and complementary approaches are the ones based in hypermedia. Hydra [5] is a lightweight vocabulary to create hypermedia-driven Web APIs, whereas RESTdesc [6] focus on capturing functionality of APIs to be used by agents. Discovery of linked data interfaces has been proposed in [7] employing hypermedia links and controls to facilitate source selection.

3.2 Bottom-up interoperability elicitation.

Since this study focuses on cases where to reach global interoperability is difficult, of particular interest are the approaches that do not aim to reach it. One is the Frisco report [8] that considers constructivist's notions as essential to building information systems, where shared knowledge can be seen here as the sum of all individual conceptions of the community, whereas personal knowledge is

[1] https://www.imec-int.com/en/what-we-offer/research-portfolio/gps4integratedcare

closer to the individual's inner reflections. Taking this into consideration, we can say models need to co-evolve along with their communities of use, disclosing multiple perspectives. A community can engineer vocabularies and ontologies through distinct, strategies or ontology-engineering methodologies [9] that address the inherent difficulty of managing a dynamic artifact that reflects our gradual understanding of reality. How to characterize ontology change was studied through layered change operators [10].

The contrast of classical data integration techniques such as ontology matching [11] and bottom-up interoperability elicitation is studied in [12] and summarized in [13]. Of particular relevance is the bottom-up approach presented by Karl Aberer et al. [14], that aims to incrementally develop a global agreement in a network of peers gossiping queries while building `semantic neighborhoods`, a notion adopted in this approach.

4 Research question and Hypotheses

The following questions (Q) and related hypothesis (H) investigate how hypermedia APIs can be applied in the context of personal agents interacting in the Web, and how can `provenance trails` be used for service discovery in a decentralized setup. This work refers to `provenance trails` as a record of how API features are copied, transformed and exchanged between agents and corresponds to sequences of operators that transform API features into others and can be collected by following links.

- (Q1) Can agents expose resources and service descriptions from their underlying dataspaces as Linked Data to be discoverable by others?
- (Q2) How can trails be represented, so they are suitable to characterize API feature change?
- (Q3) To what extent can information derived from provenance trails influence the relevance and precision of API feature search and discovery?
- (H1) An agent can discover and clone services from other agents by relying solely on Linked Data principles.
- (H2) It is possible to gather `provenance trails` and discover Semantic Web Services of high relevance without the need of a central service registry.

5 Preliminary results

The model is used in the `GPS4IC agent`, which is part of the GPS4IntegratedCare[2] project to provide RESTful APIs to be used by integrated healthcare applications. The agent is a nodeJS application that interprets JSON-LD[3] files that contain

[2] https://www.imec-int.com/en/what-we-offer/research-portfolio/gps4integratedcare
[3] https://json-ld.org/spec/latest/json-ld

sets of API features that use patient data, Domain-specific knowledge rules, and other external data sources to build at APIs at runtime for the health-care domain.

The GPS4IC agent organizes data in private and public dataspaces, namely (1) patient dataspaces, (2) service definition dataspaces and (3) knowledge dataspaces, which are logical constructs that group resources in a directory structure for simple storage, navigation, and access. Within each directory node, one can define inference operations O with links to input data and queries. Data can be N3 files, responses from a SPARQL endpoint, invocations to other operations and so on. An API is generated at runtime and exposed at a path P using a set of O that resides in a path equivalent to P of the directory.

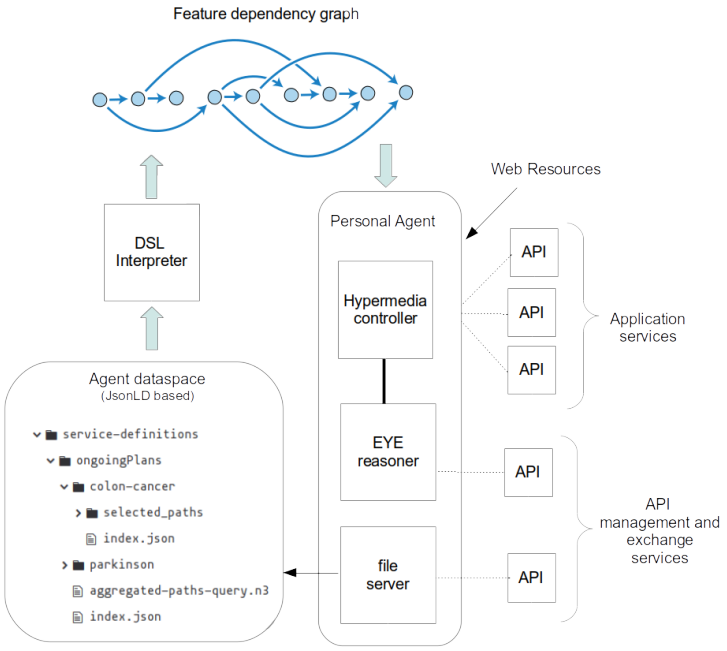

Fig. 2. The current agent uses an interpreter of file-based service descriptions to generate a feature dependency graph that is used by a hypermedia controller to expose APIs matching a request

All the core functionality such as generating dynamic workflows for the patient, detecting medication conflicts and domain-specific knowledge like the definition of diseases, etc. do not reside in the Agent.

Functionality exists as resources that can be linked, cloned and shared using Change Operators such as: (i) add/remove inference feature (ii) import feature (iii) add/remove a resource. Operations often need definitions of resources sets which

in this case, are matches of URI template patterns[4] applied to the corresponding directory structure. Agents do not maintain explicit internal schemas but commit to (believe in) a set of health-care knowledge resources.

6 Approach

This approach is framed in Web environments, specifically in the domain of Web APIs elicitation and intelligent agent support.

6.1 Feature-based

From an API perspective, this work refers to agents that can play `client` and `server` roles indistinguishably, consuming and also exposing APIs to others. Each API consists of an interface implemented by components called `features`, a logical construct that can afford interaction or functionality to be re-used [15]. A feature can be exposed through Hypermedia to be discovered, copied, modified and so on. It can be self-contained, aggregated or linked to outputs of other features, for instance, a `my contacts list` feature can be used by a `today birthdays list` through filtering it, or a dynamic workflow can depend on features that expose execution steps. Having API interfaces separated from features give flexibility to agents, enabling them to choose the features that best match their models [7] or to perform modifications at implementation level to minimize the change of their API Interfaces.

This approach requires features to be transferred between agents, although this can be done in several ways, I already had results with an agent design where service descriptors are interpreted to expose APIs by using only N3 rules, RDF data, and queries. In this way, the API functionality can be directly exchanged through the Web using simple dereferenciation or linking. API responses also expose links to functional descriptions [6] of the underlying features, with metadata about dependencies such as knowledge bases, external API or other features.

6.2 Provenance trails

Each time a feature changes within an agent or is cloned from another agent, new data is registered into what I call `provenance trails`, that refer to the history of feature change and exchange between agents.

To characterize and keep track of feature change, I adopt an approach used in ontology engineering based on layered change operators [10]. In this way, a feature is a result of applying an operator to the previous one, enabling to represent

[4] https://tools.ietf.org/html/rfc6570

provenance as chains of operations. Layers are used to support distinct degrees of granularity, categorizing change operators from atomic, i.e. `adding a triple` to high-level, i.e. `add resource collection: photo gallery`. The use of layered change operators allows characterizing feature change as a complete sequence of steps and makes easier to understand API evolution through inspection of sequences of high-level operators or their equivalent atomic ones. Other uses are to extract new high-level operators from atomic ones through pattern mining [16] or enabling feature discovery through computing similarity between their respective operator sequences [17]. I expect these trails to be valuable data to support bottom-up interoperability through data mining. For instance, if a transitive closure of `same functionality` links forms a cycle within a network of features, we can suspect that they can be marked for reconciliation through a single interface.

6.3 Mappings between personal dataspaces

In the network, agents might have joined semantic communities and peers providing similar contents (`semantic neighborhoods`) [18]. If two agents want to interact using features with models located in two disjoint neighborhoods, they might exchange their features and models and provide mappings between them. This results in directed graphs of mappings that allow querying between neighborhoods, translating queries in terms of its originator, possibly providing access to the data of a target dataspace. In this sense, trails can valuable to form and find `semantic neighborhoods` in the network by means of following links.

6.4 Metrics of feature evolution

Data about how API features change can be useful to learn about the dynamics of agents and their underlying models. For example, operator sequences allow generating quantitative indicators of feature articulation efforts, the speed of change or reusability. The data can then be analyzed by tools and techniques for mining software repositories, such as dependency network analysis [19]. On the other hand, feature exchange graphs can be used to analyze the dynamics of the agent network, identifying the most transferred features or dominant API interfaces or observe how semantic neighborhoods converge or diverge when the semantics of underlying models change. Metrics will be used in a trade-off analysis to distinguish under what conditions this approach is effective.

7 Evaluation plan

The hypothesis will be tested through two research aspects:

1. To evaluate the effects of `provenance trails` via simulations of agents that exchange features through hypermedia in a decentralized environment. To be measured through API evolution metrics and benchmarked against current semantic web service discovery approaches.
2. The impact that keeping such trails has for users while customizing their agents assisted by service discovery mechanisms without a central service registry.

H1 will be validated if instruments of (1) demonstrate automatic feature discovery and exchange using hypermedia APIs and collecting trails. H2 will be validated if (2) demonstrate fewer API articulation efforts by part of the users.

Several tasks still need to be done to perform the studies and evaluate results, answering the research questions.

- To develop vocabularies or ontologies to represent (i) feature exposition, to expose and transfer functional descriptions and their dependencies, (iii) feature change operators, with representative instances organized in layers, (ii) and feature exchange. (Q1),(Q2).
- An extensive trade-off analysis of the approach (Q3).

8 Conclusion

In this work, we present an approach where a personal agent collects functionality from other agents, improving their means of interaction while keeping model autonomy. It also introduces `provenance trails` to support automated API evolution when possible, and if not, diminish the costs that API composition has for humans. I still have considerable work regarding the implementation of ontologies and experiments, which I believe will contribute to state of the art. Being this approach specific, I understand it as complementary to existing approaches such as automatic schema mapping algorithms [11], mapping gossiping [14] between peers and the current Semantic Web Service techniques [4].

There is space for innovation regarding personal highly customizable personal agents, assembling functionality around personal data or even, in more abstract terms, to any relevant life event of a user. I believe that feature reuse is a good compromise between a one-size fits all solution and complex API programming. In the same way, personal agents can help them reach a certain degree of alignment and coherence between personal dataspaces, something essential if users want to interact together and build something valuable with the data they own.

9 Acknowledgments

I would like to thank Miel Vander Sande for his useful advice.

References

[1] G.-j. Houben, "Linking Personal Data : towards a web of digital memories," *Building*, no. 342, p. 4, 2010.

[2] T. Paul, A. Famulari, and T. Strufe, "A survey on decentralized online social networks," *Computer Networks*, vol. 75, pp. 437–452, 2014.

[3] A. Cawsey, F. Grasso, and C. Paris, "The adaptive web," pp. 465–484, 2007.

[4] D. Fensel, F. M. Facca, E. Simperl, and I. Toma, *Semantic web services*. Springer Science & Business Media, 2011.

[5] M. Lanthaler, "Creating 3rd generation web apis with hydra," in *Proceedings of the 22Nd international conference on world wide web*, 2013, pp. 35–38.

[6] R. Verborgh, D. Arndt, S. Van Hoecke, J. De Roo, G. Mels, T. Steiner, and J. Gabarro, "The Pragmatic Proof: Hypermedia API Composition and Execution," 2015.

[7] M. Vander Sande, R. Verborgh, A. Dimou, P. Colpaert, and E. Mannens, "Hypermedia-based Discovery for Source Selection using Low-Cost Linked Data Interfaces," *International Journal on Semantic Web and Information Systems*, vol. 12, no. 3, pp. 79–110, 2016.

[8] E. D. Falkenberg, W. Hesse, P. Lindgreen, B. E. Nilsson, J. L. H. Oei, C. Rolland, R. K. Stamper, F. J. M. Van Assche, A. A. Verrijn-Stuart, and K. Voss, *A framework of information system concepts - The FRISCO Report.*, vol. 4. International Federation for Information Processing; International Federation for Information Processing WG 8.1, 1998, pp. 282–7.

[9] A. Zouaq, "A Survey of Domain Ontology Engineering : Methods and Tools."

[10] C. Javed, M., Abgaz, Y.M., Pahl, "A pattern-based framework of change operators for ontology evolution," *On the Move to Meaningful Internet Systems: OTM Workshops*, vol. 5872, pp. 544–553, Jun. 2009.

[11] J. Euzenat, J. Barrasa, P. Bouquet, and R. Dieng, "D2. 2.3: State of the art on ontology alignment," *The Contributor*, 2004.

[12] P. Cudré-Mauroux, "Emergent semantics," in *Encyclopedia of database systems*, Springer, 2009, pp. 982–985.

[13] K. Aberer, T. Catarci, P. Cudré-Mauroux, T. Dillon, S. Grimm, M.-S. Hacid, A. Illarramendi, M. Jarrar, V. Kashyap, M. Mecella, E. Mena, E. J. Neuhold, A. M. Ouksel, T. Risse, M. Scannapieco, F. Saltor, L. de Santis, S. Spaccapietra, S.

Staab, R. Studer, and O. De Troyer, "Emergent Semantics Systems," in *Semantics of a networked world. semantics for grid databases*, 2004, pp. 14–43.

[14] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth, "The chatty web: emergent semantics through gossiping," *Proceedings of the 12th international conference on World Wide Web*, pp. 197–206, 2003.

[15] R. Verborgh and M. Dumontier, "A Web API ecosystem through feature-based reuse," pp. 1–12, 2016.

[16] M. Javed, Y. M. Abgaz, and C. Pahl, "Layered Change Log Model: Bridging between Ontology Change Representation and Pattern Mining," *International Journal of Metadata Semantics and Ontologies*, vol. 9, no. 3, pp. 184–192, 2014.

[17] M. Javed and Y. Abgaz, "Graph-based discovery of ontology change patterns," *Change*, pp. 1–16, 2011.

[18] D. Bianchini, S. Montanelli, C. Aiello, R. Baldoni, C. Bolchini, S. Bonomi, S. Castano, T. Catarci, V. De Antonellis, A. Ferrara, and others, "Emergent semantics and cooperation in multi-knowledge communities: The esteem approach," *World Wide Web*, vol. 13, nos. 1-2, pp. 3–31, 2010.

[19] R. Kikas, G. Gousios, M. Dumas, and D. Pfahl, "Structure and evolution of package dependency networks," in *Proceedings of the 14th international conference on mining software repositories*, 2017, pp. 102–112.