Two Short Term Applications of the Semantic Web — Working Paper (Draft)

Christoph Wernhard
Persist AG
wernhard@persistag.com

March 6, 2001

1 Introduction

We outline two application scenarios which might be in the realm of short term applications of the Semantic Web: a software packaging system and the organization of a business trip. Both of them can be solved with today's technology to some degree, so they do not show the novel potential of the Semantic Web in full. However considering Semantic Web solutions for them is useful to get a picture of the characteristics of the Semantic Web and become aware of some concrete technical issues.

2 A Software Packaging System

We take as a starting point the Debian GNU/Linux packaging system [1, 2], which successfully uses meta information on currently four thousands of independently maintained packages of bundled software to ensure their coordinated functioning.

Roles and Sources of Meta Information In contrast to e.g. a keyword index, the meta information of a software packaging system must have a very high accuracy. Otherwise computer systems using it would be out of function very quickly.

In the Debian system the meta information is provided in a distributed manner for each package by its maintainer, who should be the most competent to determine it.

This meta information is tested, in a sense, by the large user community that runs Debian systems in pre-release versions.

- *This has some similarities with the Semantic Web scenario in general:*

  - *Information is structured and has high accuracy.*[1]
  - *Information is maintained independently by many sites, which are the most competent for doing that.*
  - *Such fragments of information are combined for larger tasks.*
  - *The required accuracy of information is validated continuously by a user community.*

Composition A software package is installed on a system or not, it is never consumed by other software that uses it.

- *A physical object, on the other hand, is consumed when it becomes part of a larger object, i.e. it is no longer available to become part of another object.*
- *Therefore, for combination of software packages, the resource orientation of inferences plays a less important role than for physical systems.*

Cost of Components Since Debian GNU/Linux consists of free software, the only cost of installing a software package are disk space, download time and maybe the human effort needed for the final configuration.

- *The packaging system could be extended for commercial software with price information. In contrast to the other attributes, a price is not an inherent property of a package, but also depends on the particular vendor.*
- *Software vendors could independently provide pricing information about the packages, that can be combined with the other meta information.*

Identification of Components A package is identified just by a symbolic name.

- *This is in contrast to more complex characterizations involving e.g. subtype relationships or shared instantiation of parameters.*
- *However in some cases the package name itself contains meta information: A packages named* xxx-doc *contains documentation for package* xxx, xxx-dev *contains development libraries and headers for package* xxx, task-xxx *combines a set of other packages relevant to an application task (more on* task- *packages below).*

[1]Of course the Semantic Web can incorporate poorly structured information sources, but once information is going to be processed e.g. by an inference engine, highly structured extracts have to be used.

Description of the Semantics of Components The package meta information attribute `Description` is a text description of the package. Along with the symbolic value `Section` (e.g. *base*, *admin*, *graphics*, *editors*, *games*) it provides information about the package's semantics. Text search in the package descriptions can be used to find a package suited for a given application purpose.

- *One can imagine a knowledge based extension about the application relevant capabilities of a software package: e.g. file types it can handle, edit, display, convert, quality measures, certifications, background knowledge required by the user etc.*

- *Third parties could independently provide such additional meta information.*

Component Dependencies Package dependencies are described with attributes that have an ordered and/or-structure of package names as argument: `Depends`, `Recommends` and `Suggests` express decreasing levels of dependency. `Depends` means that the other package has to be installed for the first to work properly.

`Conflicts` means that both packages may not be installed at the same time. `Replaces` means that the other package has to be removed.

The Debian installation procedure consists of two phases: unpacking and setting up a package. The latter step is called *configuration*. Since the dependency constraints only regard configuration, a set of packages can be downloaded and unpacked in an arbitrary ordering, e.g. one that ensures that CD-ROM disks have to be changed as less as possible.[2] After that the packages are going to be configured in their dependency ordering.

Representation by Resource Oriented Inference Rules [3]

-
  ```
  configured(p1),
  configured(p2),
  configured(p3) :=action(configure(p1))
        unpacked(p1),
        configured(p2),
        configured(p3).
  ```

- *This rule expresses that package* `p1` *depends on* `p2` *and* `p3`*: the action* `configure(p1)` *leads the computer system from a state in which* `p1` *is unpacked and* `p2` *and* `p3` *are configured to a state in which* `p1` *is also*

[2]A special attribute `Pre-Depends` can enforce dependency constraints at unpacking time.

[3]See Section 4 for the syntax and intended meaning of the rules.

configured. `configured(p2)` *and* `configured(p3)` *are repeated on the left side of the rule, since they are prerequisite for application of the action, but not consumed by it. The* `unpacked(p1)` *fact is consumed by the action, the* `configured(p1)` *fact is produced by it.*

- *Some cases of* `Conflicts` *might be expressed by resource oriented facts: It is not possible to have two mail transport agents installed because both need to listen to the network to receive mail. This might be modeled by describing both as consuming the same network resource.*
- *"Soft constraints" such as* `Recommends` *(no strong dependency but also chosen for installation by default), and* `Suggests` *(no strong dependency and not chosen for installation by default) might be important in practice. How can they integrated into the inference mechanism?*

Special Components A so called *virtual* package expresses an abstract capability. It has a symbolic name, that can appear as value of the `Provides` attribute, declaring that some package provides that abstract capability. A further package can then depend on the virtual package, which means that just one of the packages providing the virtual package has to be installed. An example for a virtual package is *httpd* (i.e. Web server), which is provided for example by packages *apache*, *aolserver*, *boa*, *cern-httpd* and *roxen*.

Task-packages are virtual packages that combine by their dependents a set of packages for some larger application task chosen explicitly by the user, e.g. *task-sgml*, *task-japanese*, *task-laptop*, *task-games*.

Processing The Debian installation program maintains information about the packages installed on the system as well as about the set of available packages by collecting package meta information from a set of user-specified sources such as CD-ROMs and FTP sites. If invoked with a set of package names the installation program determines which of the specified packages is not already installed, which dependents have to be installed and which conflicting packages have to be removed. It prints a description of what it would do, that has to be confirmed by the user before it is going to be executed.

- *This has some similarity with the operation of a Semantic Web planning agent. In one phase information is gathered from various sources, in a second it is combined into an execution plan that takes into account a given "start" state. After user confirmation the plan is going to be executed.*[4]

[4]This is a conceptual base architecture only. In practice gathering, planning and execution might happen in a more distributed and interleaved manner.

Alternative Compositions In the Debian packaging system, alternatives are simply resolved by preference according to the ordering of disjuncts in the attribute values.

- *The resource oriented inference approach allows the generation and comparison of different possible plans. At first sight this feature might not be needed for a software packaging system, but there are circumstances under which it would be useful:*
- *If software that costs something is included, different installation plans might have essentially different properties.*
- *Disk space is not cheap on all systems.*
- *Planning can contribute to "mental economy": Combined with knowledge based package descriptions, one could determine software needed for certain tasks and get hints which new software systems must be learned to use.*

3 Organization of a Business Trip

A business trip requires coordinated use of services of several different kinds, offered by several providers, where each kind of service again is provided by several competitors. This includes transport (walking, use of own car, taxi, rental car, local public transport, railway, airplane), hotels and restaurants.

Composing and Comparing Services In today's Web detailed information about most of these services as well as electronic booking facilities are available, such as timetables, route-planners, hotel and restaurant guides, price lists etc. Usually each of them has its very own graphical Web interface, that must be crawled through manually to get information needed to make decisions (e.g. compare prices, schedules, availability) and to implement decisions (i.e. order services).

- *With the Semantic Web the coordinated composition of these services based on common interfaces is possible. These interfaces can be supported by the service providers directly or by mediators. A planning agent computes different compositions and compares them. After the user has committed to one of the suggested plans, the agent performs the necessary orderings and prints out the travel documents.*
- *The Semantic Web further allows to directly connect service offerings with information from the customer's site. Business trip planning can e.g.*
  - *be directly connected to a personal date book;*

  - *be used to organize a round trip with several meetings;*
  - *be used to organize a meeting involving several people, based on their date books.*

- *A distributed transaction protocol, is required: if the failure of an order leads to the choice of a different plan, previously made orders must be undone. The scope of transactions possibly should also include the planning phase.*

Existing Aggregations of Services There are some aggregations of the involved services in today's Web: e.g. car rental brokers or car rental agencies giving discount related to the airlines of arrival.

- *These existing aggregations do not involve comparison of solution plans involving different kinds of services (e.g. railway vs airplane).*
- *They seem to rest on particular agreements between the involved parties. Pre-arranged aggregations are cheaper for the customer, since e.g. car rental brokers might buy large contingents and pass this advantage to the user. On the other hand they lead to cartel agreements and monopolies.*
- *The Semantic Web seems to create a market, in which anyone can participate by publishing an offer or inquiry.*
- *However the Semantic Web also seems to require strong control mechanisms, such as "trusted sites".*
- *The Semantic Web is able to perform the function of a car rental broker in a distributed manner. The "contingent" is implicit in the number of orderings.*
- *By the possibility of including offers of small enterprises, special customer needs can be satisfied, e.g. a small car rental agency in a remote place can be included.*

Enterprise Identity In the past years enterprises tried hard to find their identity within the world of connected computers by pushing huge advertising budgets into the HTML-Web.

- *The identity of the involved enterprises is hidden to the user during the planning (i.e. decision making) process of the Semantic Web.*

Flexibility Business trip processes seem not to change very much in the future. Information could be improved, e.g. a route-planner that takes expected traffic on the particular date into account, car navigation systems. Technical alternatives to business trips might become convenient, e.g. videoconferencing.

- *So maybe the flexibility offered by the planning approach is not really useful in this example. The Semantic Web provides a large amount of flexibility in the way plans, i.e. business processes, complex products or services, are composed: new ways of composition which are not pre-conceived as a whole can be inferred from the small partial plans provided by the individual offerers.*
- *A drawback of composing plans from small units is that there might easily be lots plans generated, which are useless in a ways that have not been "pre-conceived", which means that it could be difficult to filter them out.*
- *Besides inferencing plans, a library of plans can be used. It includes descriptions of processes and complex products involving several successive steps. It is organized in a distributed way, similar to schemas describing object sturcture (ontologies). When using those given plans, the Semantic Web agent acts more as a scheduler than as a planner.*

Representation by Resource Oriented Inference Rules [5]

-
  ```
  %% trip_done combines that the business meeting is done
  %% and the person is at home. It can be used as
  %% "toplevel" goal.
  %%
  trip_done(person, place, t_from, t_to, home_place),
  business_meeting_done(person, place, t_from, t_to),
  at(person, home_place)
          :=action(trip_finished)
      business_meeting_done(person, place, t_from, t_to),
      at(person, home_place).

  %% If the person is at the right time and place, the
  %% business meeting can be done. The person remains
  %% at that place.
  %%
  business_meeting_done(person, place, t_from, t_to),
  at(person, place, t_to)
  ```

[5] See Section 4 for the syntax and intended meaning of the rules.

```
        :=action(do_business_meeting(...))
    at(person, place, t_from).

%% Transport by air.
%%
at(person, dest_airport, t_arrival)
        :=action(do_flight(person, flight_no, date))
    at(person, start_airport, t_departure).
```

- *The transport services consume and produce (besides cost and time)* `at(person, location)` *facts. The start state consists of an* `at/2` *fact (along with a point of time), the goal consists of an* `at/2` *fact expressing that the traveler is back home again and a* `have_done_business` *fact. The* `have_done_business` *fact requires a third* `at/2` *fact with a certain time, without consuming it.*
- *How time and cost are represented precisely?*
- *Hotel and restaurant provide* `shelter` *and* `food` *facts, which need to be consumed if the trip goes overnight or over a certain amount of time. How this is represented?*

4 Appendix: Inference Rules

We use the following syntax for a resource oriented inference rule:

`<produced_facts> :=action(<action>) <consumed_facts>.`

This is similar to a Prolog clause, with `<produced_facts>` the head and `<consumed_facts>` the body. In contrast to Prolog, the head may contain multiple literals, and within the arrow (Prolog's `:-`), there is an action term `action`. Also in contrast to (the logical meaning of) Prolog, `<produced_facts>` and `<consumed_facts>` are not idempotent. The informal meaning of such a rule is, that from a state in which all members of `consumed_facts` hold, by applying `action`, a successor state can be reached in which `consumed_facts` do no longer hold (the facts are removed from the state description), and `consumed_facts` hold (are added to the state description). If a fact must be present for a rule to be applied, but is not consumed by rule application, it must appear in both `<consumed_facts>` and `<produced_facts>`.

References

[1] Ian Jackson et al.: *Debian Packaging Manual*

[2] Jason Gunthorpe: *APT User's Guide*