

The SOA Magazine

Feature Article



The Service Engine: Structured Communication using Modern Service Technologies

by Paul S Prueitt

Published: July 30, 2009 (SOA Magazine Issue XXX: July 2009)

[Download PDF](#)

[Digg This](#) • [De.licio.us](#) • [Slashdot](#) • [Technorati](#) • [StumbleUpon](#) • [Google Bookmark](#)

Abstract: A simple stand-alone fixed finite state machine may be used as an engine that determines all triggers for a particular set of web services. Such an engine requires a uniform design regime over all service contracts [REF-1]. This type of machine may exist as a URI/URL web service provider. The community of such providers defines a network infrastructure for social networks engaged in common tasks. Within this infrastructure, services may be offered and fulfilled using a structured method for machine specification [REF-2] (SMMS). Data persistence is distributed in a peer-to-peer architecture, again very simple in nature. XML files with very simple XML schema [REF-3] are all that is necessary and may be passed around the web using HTTP requests. The files may be manufactured in AJAX, SOAP or REST architectures. XML/XML-schema may also be manufactured using a hash table. Persistence may then be in the cloud with a minimal repository.

Introduction: The Service Engine

So what will a service engine do? It will model context using direct input from everyday use. In essence there is a separation of particular instances, as experienced by the individual, from the composition of universals forming the common basis for human communication. As such it supports social network infrastructure through the provision of a knowledge management type community enumerated framework.

The framework might be a Zachman [REF-4] or Sowa [REF-5] framework, or one of a general class of frameworks. In simple service terms, the common parts of services are associated with templates and services are composed using only those templates. The templates may be seen as a set of frames, or windows, with slots and fillers defined directly by the enumerated framework. The point is that humans use the enumerated frame as a guide in defining frames. The use of a fixed and finite set of enumerated slots and fillers creates closure and as a result we have a property aligning the capability of finite state machines and human intention.

Social Network Infrastructure

The driving force in cloud computing illustrates near term evolution of social network use. Examples include social movements in various parts of the world. Click through to contextually and location specific landing pages, e.g., preexisting web page templates, now shape an emerging advertising sector. Modern service technologies support "structured communication" between individuals working or playing in everyday situations. In a way that is similar to natural language use, re-usable knowledge representation has become un-encumbered by desktop software or operating systems.

It is noted that Twitter [REF-6] sends natural language expressions restricted to 140 characters. Its adoption worldwide was viral. In the abstract we see tweet processing as the beginning of service orientation in the cloud. Letters and words are used and yet limited to a fixed set of letter characters [REF-7]. A knowledge

tweet will be both the same and different. A frame with slots and fillers has a small envelope and is composed by individual humans. Enumerated frameworks are selected through community-based collaboration. The community itself culturally orients this selection. However structured communication is also an encoding of individual human judgment and may be aggregated into community mediated collective knowledge. This aggregation may occur completely within an "ultra" secure regime.

The identified community and individuals within the community play roles that are familiar because of human experience with natural language. The encoding might be as simple as composing a short text message or be far more elegantly supported as part of a community knowledge processor. If social networks, themselves, impose reusable structure to human communication, as we suggest, then one may have a knowledge processor in the cloud [REF-8]. That would be useful in lots of ways.

An Illustration of the Social Network Infrastructure

We first point out that knowledge elicitation within social networks has been a knowledge management concern, and is addressed extensively in that literature [REF-9]. So what is so exciting and different in July 2009? The world is changing rapidly and moving in the direction of democratization and transparency. Social networks form and perform work. Capitalization of social innovation is supported because profits are to be made. Expected application sectors include energy production decentralization, increased government efficiency, and educational processes. Applications to medical research and crisis response are clearly envisioned. Social response to change has a new avenue for creative expression.

Each social network may form structure and its own economy system, as well as a system that protects its internal methods. Costs will be measured in terms of human capital, not IT expenditures. For this reason and others, we suggest that a social network infrastructure is necessary to firmly establish a market presence for those companies that would be competitive in this new economic system [REF-10].

Knowledge Representation as "Situationally" Secure Ontology

We are just beginning to see market forces fund the development of separable [REF-11] knowledge based social network infrastructure. The technical and cultural problems have; however, been studied for some time. The most difficult issues evolve around the merge of two different knowledge representations or the updating of a single representation. The world changes and perceptions change. How does one change one's mind about something? What would have to happen if a finite state machine were "modeling" this change [REF-12]?

Current, over engineered, knowledge engineering practice automation works well as long as all interactions are fully deterministic. In current practice, an expert knowledge engineer will manage the merge, or modification. However, the current knowledge engineering technology in the form of web ontology language is confusing and non-agile. For this reason, the mismatch between deterministic models and active human knowledge has been an impassible challenge.

The core technical issue might be described as concerning the nature of induction, which the human does easily; and deduction, which is what finite state transactions do: "If in state x move to state y". The issue is about entailment, what causes x to move to y.

It has not been demonstrated by anyone that computers do think, do experience or that computing machines have perception. We think, experience and have perception naturally. Moreover, it is often forgotten that computing theory is a subset of mathematics. Perfectly designed computing systems will still be subject to Gödel theorems [REF-13]. A system of interacting finite state machines is a finite state machine. Finite state machines do not become something different simply because we might make them bigger and bigger, or more complicated. An analogy is possible. An integer does not become infinite by adding 1 to it no matter how many times this is done. The notion of infinite changes everything. The same is true regarding a truthful understanding about natural complexity and the nature of computing machines. Mathematics does not now have complexity [REF-14].

To include natural complexity in our design of communication mediums is to change what mathematics is. A computing machine might therefore be usefully defined as "simple", if and only if no non-predetermined actions are possible. The other side of "simple" is then "complex". The confusion over complicated and complex is set aside.

The Opening of the Closed Machine

So how does the finite state machine become open to human interaction? The answer is that a "clopen" mechanism is required.

Definition (Prueitt, 2005):

Clopen means, "closed" formally but may be opened axiomatically by an outside cause [REF-15].

Computers run if there is no complexity, and fail or halt at transitions where no pre-determined state transition is defined. The natural world is quite different; it is complex. If cloud computing is to work then the finite state machines must be closed otherwise they will not work. But these same systems must have clopen mechanisms [REF-16] otherwise they become not relevant.

In process stratification, a normalization of a fixed and finite set of process atoms creates a simple computer, or more precisely a simple machine. This machine then interacts with a natural world. Of course it is an easy matter to create overly complicated machines. We see these complicated machines everywhere. The trick is to create a simple machine that is not complicated and yet is responsive to complexity. This may be done with classical workflow, e.g., a deterministic workflow with no humans in the loop, and with web services that opens deterministic workflow up to modification. The resulting system will form a closed specification to a finite state machine.

Complexity and Computer-Human Interaction

"Enumerated design" using enumerated frameworks makes cost manageable, while providing a means to secure all transmissions of data. Issues related to non-interoperability are completely bypassed. Let us review how this works. Complexity intrinsic to non-deterministic human interaction creates a requirement to open the finite state machine, so defined, to human design processes. Re-closure is fast and managed within a clopen status, a status that separated the design process from the social network. Human interaction may then use the closed finite state machine to securely carry bits of pre-structured information. Stratification creates separately defined slots and filler data minimally expressed within web-propagated encoded/encrypted frames.

Two organizational levels, context and content, are independent. Modification of the set of contextual atoms occurs under specified community design process. Content is supplied within compositional forms and thus content is more easily processes, and understood. Enumerated frameworks minimize the cost of atom modification. Encryption works within the framework structure of information to create very difficult to break coding. If the community needs a modification then it will occur, and if individuals wish to hold on to old patterns, then this may also occur. Reconciliation of intentional differences entails potential separation of that individual from the community. So new communities may form and dissolve based on actual dynamics between humans.

Stratification creates two types of clopen properties. This is helpful. The compositional practices in a social system are a lot easier to modify than are the atoms. Individual compositions are expected to be sometimes unique and the result of creative work. So the use of structured communications is seen as similar in nature to natural language expression where the same alphabet and phonetic set is composed into the expression of humans. We expect novel uses of the structured form and in fact any such novelty may be accompanied with copyrights and property protections. Intellectual property may be strongly protected using super distribution [REF-17].

Process Genealogy

Use of compositional mechanism is called genealogy. Genealogy of the type supporting structured communication requires service architecture that is open, stratified and generative. For example, communication structure is coded in letters as composed in words. The selection is compositional and thus involves genealogy. The mathematics that allows us to set up this infrastructure is category theory, and elements from the foundations of logic.

Process genealogy allows a rich expression without changing the underlying technical specification to social

network infrastructure. Stratification bypasses the core difficulties in current software by providing an easy means to extend or modify an existing closed finite state machine. Remember, if a finite state machine is directed to move out of what has been specified, it halts. There is no outside, of it. So it simply halts. The real world has to step in and do something. The creation process is opened when the genealogy is changed. Once the specification for a finite machine is opened, the finite state machine specification must be closed before the machine will run again. We take advantage of this "flaw". Stated slightly differently: the machine will not work unless the new expression has a specific closure property as discussed by Klausner in the Cubicon specification [REF-18].

Application to Web-based Services

There are significant works done on axiomatic closure [REF-19] and web service closures. REST is likely the most profound [REF-20]. Any of the web service paradigms, for example SOAP and WSDL [REF-21], might be used to produce a prototype of stratification. What has been missing is a framework-based specification of templated transactions. A web service is provided to an object where that object has q slots each slot having a fixed and known number of states (of fillers) [REF-22]. The service may itself move the object's state to another state, again within a known set of states as well as manage communications between other service providers.

This reach-ability condition is easy to achieve and is easy to understand, if and only if the specifics of data categories and details of a fully specified workflow have been completed [REF-23]. Two problems commonly exist. First the set of web services is never completed because the software project fails to complete. The second type of problem is the development of services does not proceed within an enumerated framework.

One way to keep system design managed is to require a mock up of the web services [REF-24]. Mock up provides a test bed allowing a full exploration of accessible states. The finite state machine, so defined, may be instrumented with a specific set of multi-category-control mechanisms and may be an enumerated set of URIs. Of example $\{ 1, 2, 3, 4 \}$ where possible machines might be $2 \rightarrow 1 \rightarrow 3 \rightarrow \text{halts}$ or $2 \rightarrow 1 \rightarrow 2 \rightarrow 1$. Halt is a special "complex type state" where under the translucence property, external modification of state may occur. In a framework based specifications, these mechanisms act over ALL possible state transitions.

The Knowledge Framework as a Cross Product

We may associate base elements with the specific enumeration of an axis of description. For example the six Greek interrogatives;

{ who, where, what, how, why, when }

form an axis of enumeration. A two axis cross product is in the form:

$Q \text{ cross } R = \{ (q, r) \mid q \text{ is an element of } Q \text{ and } r \text{ is an element of } R \}$

Each composition of elements from the set $\{ (q,r) \}$ may be used in establishing a context. Context is then a string, which is the URI/URL for a state, specific to the points in this cross product. The transaction finite state machine is then merely pointing at one location and then another, etc. These "locations" are then context processors which know how to treat slot fillers; much the same as a spell checker knows how to parse text composed of letters and words.

How might XML work with work flow? There is an expectation going in or going out of the XML marshaling process. Workflow, under current practice is autonomous, e.g., without human interaction. Autonomous workflow by definition occurs within in predefined finite state machine. This means that the transition space is closed. Any reaching to the outside creates a halting condition, since the machine will not be able to get to that state. All known states are determined. For example 02 is not an allowed binary state, since "2" is not defined. An attempt to move to 02 will halt the finite state machine. In finite state machines with enumerated frameworks, each position within a state representation may only be filled with a unique identifier. XML serialization means that all properties and attributes are written as well formed XML. Some things are allowed to change and some things are not, without triggering an auxiliary mechanism [REF-25].

A type of category theory arises that maps all possible slot positions to a small representation of state. Representational state transfer, as in REST thesis by Fielding, takes on a whole new dimension. Cloud

computing with ontological categories is then immediately available. The states are code-able as header enveloped pointers to a hash table. A set of such hash table elements may also be mapped to a corresponding set of transmission packets, by concatenation of a normalizing pre-header bit structure.

```
{ (header | payload) } --> { (bitStructure+header | payload) } or
{ (header | payload) } --> { (header | bitStructure+payload) }
```

The normalizing bitStructure encoding may be mapped uniquely to a standard hash table, or to a Java Jar, or a .NET scoped transaction set. This is one way to lift the enumerated frameworks to cloud computing. There are lots of easy and lots of very difficult options here, some of which make modifications to the HTTP standard or to how the current standard is handled.

Structured Method for Machine Specification (SMMS)

Stratified ontology replaces deductive inference with deterministic workflow. Workflow specification is open to human influence. A structuring process takes what is available through measurement, such as through semantic extraction or link analysis methods, and reifies ontology in the absence of an inference engine [Ref-26]. Inference may then become inductive, as ontological information such as taxonomy (or "folksonomy") is made available for reuse. A causative information resource is built with semantic atoms from which informative compounds are orchestrated in response to individual and collective inquiry. A generative nature arises from process stratification, as is illustrated by a gene pool and the expression of genes as phenotype. The two levels are "atoms" and "compounds". The set of atoms acts as a memory store modified by positive or negative re-enforcement.

We note that the language of process genotype, morphology, and phenotype is consistent with formative and generative natures in biology. The individuals may be seen as compositions of atoms, in the way that chemical compounds are compositions of physical atoms. The expression may be seen as the aggregation of genotype into phenotype, as controlled by mechanisms found in morphology. As another illustration, the expression of individual human speech may be seen as the composition of phonemes. In all these illustrations, small sets of atoms are expressed within a selected part of the combinatorial span defined by morphology. A synthesis of three elements is seen as a set of behavioral atoms, a set of mechanisms and a set of individuals.

Conclusion

SMMS forms a set of underlying atoms, a set of mechanisms that are expected as part of an ecosystem composed of a fixed and finite set of categorical abstractions; e.g., as classes in an ontology, and the expressed particular. The particular is deterministically generated using fixed finite state machines. If in a particular instance this generation is considered to be inadequate then a clopen mechanism is available at both the individual and community organizational levels. The situational encoding of knowledge generated by individuals within a social network infrastructure is then possible. Individuals compose by enumerating into a cross product, where enumeration in each cell of the cross product is subject to a categorization of the individual cell of the cross product, as a code.

References

[REF-1] A service contract is a technical specification between computers using service oriented design, see Thomas Erl et al., (2009) "Web Service Contract Design and Versioning for SOA", Prentice Hall Service-Oriented Computing Series from Thomas Erl

[REF-2] Structured method for machine specification (SMMS) uses any one of a class of enumerated frameworks.

[REF-3] Quote from: <http://www.learn-xml-schema-tutorial.com> "XML Schema is an XML-based language used to create XML-based languages and data models. An XML schema defines element and attribute names for a class of XML documents. The schema also specifies the structure that those documents must adhere to and the type of content that each element can hold." Accessed July 4, 2009.

[REF-4] <http://www.zifa.com/>

[REF-5] Sowa, J., Zachman, J.: "Extending and formalizing the framework for information systems

architecture." IBM Systems Journal 31 (1992) 590-616

[REF-6] www.twitter.com

[REF-7] These tweets may easily be aggregated using well-understood semantic extraction techniques, as discussed in Prueitt, Paul S (2006) "Global Information Framework and Knowledge Management", published at: <http://www.bcngroup.org/area1/2005beads/GIF/RoadMap.htm>

[REF-8] The Google search string "knowledge processor in the cloud " is void, however <http://www.processor.com/editorial/article.asp?article=articles%2Fp3114%2F69p14%2F69p14%2F69p14.asp> is a very good description of knowledge processing in a cloud

[REF-9] Prueitt. P. (2001). Foundational Paper on the Transformation of Knowledge Ecology to a Knowledge Economy Knowledge Management Consortium Institute Journal, Vol. 1 Issue 2

[REF-10] Prueitt, Paul (2006). "Service Oriented Architecture in the presence of Information Structure". Topic Maps 2006, Earley & Associates

[REF-11] Each social network may control access to its internal methods without any third party interference. This means that millions of separable social networks may arise, each in control of its methods and of the sharing of knowledge structure. The ultimate regulation of these social processes is revealed in the marketplace of ideas and products, and is thus subject to civil and criminal law enforcement.

[REF-12] The exchange of structured communication is "modeled". This does not mean, perfectly modeled. Mathematics "models" the world of mechanical mechanism and arithmetic models common economic transactions. As human natural language models the world; it is possible that the models are themselves misleading or flat incorrect. Human in the loop activities are deemed as necessary to transparency within the community. Through transparency, within the community, reinforcement corrects models on a continuing basis.

[REF-13] Gödel, Kurt: 1940. The Consistency of the Axiom of Choice and of the Generalized Continuum Hypothesis with the Axioms of Set Theory. Princeton University Press

[REF-14] Prueitt, Paul S. (1996b). Is Computation Something New?, published in the Proceedings of NIST Conference on Intelligent Systems: A Semiotic Perspective. Session: Memory, Complexity and Control in Biological and Artificial Systems. IEEE October 20-23.

[REF-15] Prueitt Paul S (June 2009) Internal NUACC document: "Technical Foundations II: stratified theory and articulated machines

[REF-16] I must call this a Rosen lemma, as it follows from the definition of complexity given by Robert Rosen

[REF-17] Ryoichi Mori, Masaji Kawahara, "Superdistribution: The Concept and the Architecture". Transactions of The Institute of Electronics, Information, and Communication Engineers, vol. E73 #7, July 1990, pp.1133-1146

[REF-18] Klausner, Sandy (private communications and as seen form web site: www.coretalk.net)

[REF-19] The notion of axiomatic closure is discussed in various ones of Paul Prueitt publications.

[REF-20] Fielding, (2006) Representational State Transfer (REST) Chapter Five http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[REF-21] SOAP and WSDL: Simple Object Access Protocol and Web Service Description Language are standards widely adopted.

[REF-22] This notion of slots and fillers is from Frank Schank's work and is seen in the KIF standard.

[REF-23] This is the core of our response to a current client request for assistance. The capability we are presenting is however also more general.

[REF-24] For example see : www.soapUI.com

[REF-25] The distinction commonly made in XML marshaling (serialization or de-serialization) is that features are dynamic and properties are not dynamic (within a transaction envelop). The issue is that dynamic or static within a transaction is open to modification from one set of transaction to another set, using the XML open world hypothesis.

[REF-26] Inference has been found to be a difficult problem where complete solutions may have non-removable barriers. Thus we remove inference from taxonomy and ontology specification.