Parent Item: A process algebra based framework for promise theory

π-calculus

Useful for the implementation of promise theory policy management process.

Extensions:

- Petri nets
- Process algebras with probabilistic extensions

These formalisms are more like control theory— they are about understanding events in fully specified systems, with sychronization and resource contraints. They do not capture the behaviour of a system given an imperfect knowledge or specification, such as that one usually faces in a management scenario. There is thus additional uncertainty to be considered.

Representation

Promise type	Notation	Interpretation
Basic	$n_1 \xrightarrow{\pi} n_2$	Provide service/flow
Cooperative	$n_1 \xrightarrow{C(\pi)} n_2$	Imitate/Follow
Use	$n_1 \xrightarrow{U(\pi)} n_2$	Use/Accept from
Conditional	$n_1 \xrightarrow{\pi_1/\pi_2} n_2$	Promise 'queue': π_1 if π_2

Basic service promises form any number of types, e.g. 'provide web service in less than 5 milliseconds' or 'tell you anything about X'. Some other examples include:

- $X \overset{q \leq q_0}{\rightarrow} Y$: agent X promises to never exceed the limit $q \leq q_0$.
- $X \overset{q=q_0}{\rightarrow} X$: agent X promises to satisfy $q = q_0$.
- $X \overset{\ell \subseteq L}{\rightarrow} Y$: X promises to keep ℓ to a sub-language of the language L.
- $X \overset{S}{\rightarrow} Y$: agent X offers service S to Y.
- $X \overset{R}{\rightarrow} Y$: agent X promises to relay R to Y.
- $X \overset{\neg R}{\rightarrow} Y$: agent X promises to never relay R to Y.
- $X \overset{S,t}{\rightarrow} Y$: agent X promises to respond with service S to Y within t seconds.

A promise is said to be broken if an agent makes two different promises of the same type at the same time (this is different from a promise that has expired or been changed).

Tags: control-theory, π-calculus

Parent Item: In search of an understandable consensus algorithm

Raft consensus

TODO: Print page 4

- **Leader election**: a new leader must be chosen when an existing leader fails (Section 5.2).
- **Log replication**: the leader must accept log entries from clients and replicate them across the cluster, forcing the other logs to agree with its own (Section 5.3).
- **Safety**: the key safety property for Raft is the State Machine Safety Property in Figure 3: if any server has applied a particular log entry to its state machine, then no other server may apply a different command for the same log index. Section 5.4 describes how Raft ensures this property; the solution involves slight extensions to the election and replication mechanisms described in Sections 5.2 and 5.3.

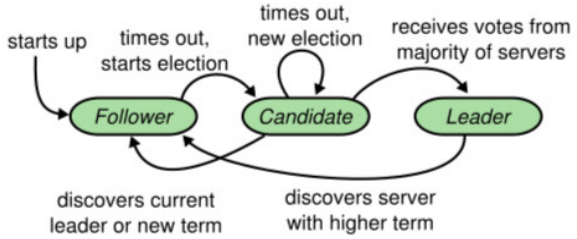
Raft guarantees that each of these properties is true at all times. The section numbers indicate where each property is discussed :

- **Election Safety**: at most one leader can be elected in a given term. §5.2
- **Leader Append-Only**: a leader never overwrites or deletes entries in its log; it only appends new entries. §5.3
- **Log Matching**: if two logs contain an entry with the same index and term, then the logs are identical in all entries up through the given index. §5.3
- **Leader Completeness**: if a log entry is committed in a given term, then that entry will be present in the logs of the leaders for all higher-numbered terms. §5.4
- **State Machine Safety**: if a server has applied a log entry at a given index to its state machine, no other server will ever apply a different log entry for the same index.

Server states

A Raft cluster contains several servers (five is a typical number, which allows the system to tolerate two failures). At any given time each server is in one of three states:

- **Leader**: In normal operation there is exactly one leader and all of the other servers are followers. The leader handles all client requests (if a client contacts a follower, the follower redirects it to the leader)
- **Follower:** Followers are passive: they issue no RPCs on their own but simply respond to RPCs from leaders and candidates.
- **Candidate:** is used to elect a new leader as described in Section 5.2. Figure 4 shows the states and their transitions;



Followers only respond to requests from other servers. If a follower receives no communication, it becomes a candidate and initiates an election. A candidate that receives votes from a majority of the full cluster becomes the new leader. Leaders typically operate until they fail.

Raft RPCs

RequestVote RPCs

initiated by candidates during elections

AppendEntries RPCs

initiated by leaders to replicate log entries and to provide a form of heartbeat

InstallSnapshot RPC

for leaders to send snapshots to followers that are too far behind. Upon receiving a snapshot with this RPC, a follower must decide what to do with its existing log entries. It must remove any log entries that conflict with the snapshot (this is similar to the AppendEntries RPC). If the follower has an entry that matches the snapshot's last included index and term, then there is no conflict: it removes only the prefix of its log that the snapshot replaces. Otherwise, the follower removes its entire log; it is all superseded by the snapshot

---

Raft vs. Paxos consensus

Paxos is more difficult to understand. Raft primary goal is understandability, including decomposition (Raft separates leader election, log replication, and safety) and state space reduction (Raft reduces the degree of nondeterminism and the ways servers can be inconsistent with each other).

Paxos drawbacks

- Exceptionally difficult to understand. Paxos' opaqueness derives from its choice of the single-decree subset as its foundation.
- Does not provide a good foundation for building practical implementations. For example, there is little benefit to choosing a collection of log entries independently and then melding them into a sequential log; this just adds complexity. It is simpler and more efficient to design a system around a log, where new entries are appended sequentially in a constrained order.
- Uses a symmetric peer-to-peer approach at its core. This makes sense in a simplified world where only one decision will be made, but few practical systems use this approach. If a series of decisions must be made, it is simpler and faster to first elect a leader, then have the leader coordinate the decisions.

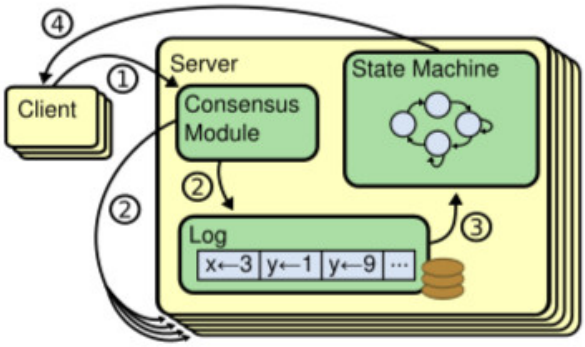
There are significant gaps between the description of the Paxos algorithm and the needs of a real-world system.... the final system will be based on an unproven protocol.

Raft extra-features:

- Strong leader: Raft uses a stronger form of leadership than other consensus algorithms. For example, log entries only flow from the leader to other servers. This simplifies the management of the replicated log and makes Raft easier to understand.
- Leader election: Raft uses randomized timers to elect leaders. This adds only a small amount of mechanism to the heartbeats already required for any consensus algorithm, while resolving conflicts simply and rapidly.
- Membership changes: Raft's mechanism for changing the set of servers in the cluster uses a novel joint consensus approach where the majorities of two different configurations overlap during transitions. This allows the cluster to continue operating normally during configuration changes.

Replicated state machines (fault tolerence)

Replicated state machine architecture. The consensus algorithm manages a replicated log containing state machine commands from clients. The state machines process identical sequences of commands from the logs, so they produce the same outputs.



large-scale systems that have a single cluster leader use a separate replicated state machine to manage leader election and store configuration information that must survive leader crashes (Ex: Chubby, ZooKeeper)

Keeping the replicated log consistent is the job of the consensus algorithm

Consensus algorithms for practical systems typically have the following properties:

- They ensure safety (never returning an incorrect result) under all non-Byzantine conditions, including network delays, partitions, and packet loss, duplication, and reordering.
- They are fully functional (available) as long as any majority of the servers are operational and can comunicate with each other and with clients. Thus, a typical cluster of five servers can tolerate the failure of any two servers. Servers are assumed to fail by stopping; they may later recover from state on stable storage and rejoin the cluster.
- They do not depend on timing to ensure the consistency of the logs: faulty clocks and extreme message delays can, at worst, cause availability problems.
- In the common case, a command can complete as soon as any majority of the cluster has responded to a single round of remote procedure calls; a minority of slow servers need not impact overall system performance.

Parent Item: Laws of human-computer behaviour and collective organization

---

Autonomous agents: laws of behaviour

Promise Theory

A promise is different from a commitment, since a commitment is a moment at which an agent breaks with one course of behaviour for another discontinuously with sights on a goal, often through some specific action or investment in the future outcome. In some cases the act of committing can result in a persistent promise as its outcome, but promising does not imply an action that makes a discontinuous change.

Promises are made by a promiser agent to a promisee agent as a directed relationship labelled with a promise *body* whichdescribes the substance of the promise. A promise with *body* **+b** is understood to be a declaration to "give" behaviour from one agent to another (possibly in the manner of a service), while a promise with body **−b** is a specification of what behaviour will be received, accepted or "used" by one agent from another

see page 3 (left col)

- promise valuation v

- type / constraint

Since any dynamical, systematic behaviour is a balance between degrees of freedom (avenues for change) and constraints[23], this should be sufficient to describe a wide variety of phenomena. For many purposes, and to avoid extraneous concepts, the environment in which agents live and act can itself be represented as an autonomous agent with extensive internal resources. We denote this agent E.

Parent Item: Managing virtualization of networks and services 18th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2007, San José, CA, USA, October 29-31, 2007: proceedings

Creation of a botnet middleware to manage the context information

---

Parent Item: Promise theory-a model of autonomous objects for pervasive computing and swarms.

Border Gateway Protocol: Autonomous Systems

An area of system management that control theory has difficulty addressing is that of the behaviour of a network which uses the Border Gateway Protocol. BGP is perhaps the only autonomous peer system in widespread use today, and it is frequently the lament of network analysts and designers.

BGP breaks the mould of deterministic algorithms and hence breaks Computer Science's traditional arsenel of techniques for analysis. It is voluntary cooperation which leads analysts to fear BGP. Promise theory is perfect for this task, however and it is worth revisiting this old problem in some detail to discover the limits of BGP. This remains to be done

Mechanics

BGP's main tasks for an 'Autonomous System' (AS) are to advertise networks within its borders.and distribute proposed routing paths to neighbours. It involves implicit trust, and therefore intrinsic uncertainty. These tasks are performed as services, with each AS having the power of veto. In BGP, filter lists and distribution lists are access control filters which set policy about who shall receive route advertisements or traffic. Incoming access control is clearly implemented by either having or not having either a 'use promise' or a conditional relay promise. Outgoing data transfer is implemented or not by having a data transfer promise.

Control Theory

TODO: Add to biblio

- J.L. Hellerstein, Y. Diao, S. Parekh, and D.M. Tilbury, Feedback Control of Computing Systems, IEEE Press/Wiley Interscience, 2004

In spite of the popularity of peer to peer services and personal consumer electronics, models of computer management are still struggling to break away from this idea of centralized control.

The legacy approach to modelling management behaviour is typified by control theory. This is a form of analysis that stems from electrical engineering, where one imagines a network of components that drive one another in a predictable way. Downstream components of a control network are driven by their upstream counterparts, and the entire network is driven by a number of sources.

Control vs. Promise Theory

Rather than adopting the conventional belief that “only that which is programmed happens”, it takes the opposite viewpoint: “only that which is promised can be predicted”. It therefore approaches management from the viewpoint of uncertainty with realism rather than faith in compliance. In promise theory, one assumes a service viewpoint of interactions between computing devices. Each node offers to play a part in a collaborative network by promising to constrain its behaviour in various ways. A typical use of promise theory is to examine the steady state behaviour of a number of autonomous agents (components). Promise theory does not necessarily fully determine the behaviour of its components, it only represents constraints within a set of defined behaviours. It is not limited to linear promises.

Simple comparison

Control theory uses a scalar value with a compatible engineering dimension throughout (like an electric current or voltage). Promise theory, on the other hand, involves promises of several different types, based possibly on possibly structured information models. A single promise can be a declaration of intent to keep a general tuple of data within a given range of values. Promises are thus typed and their attributes can belong to discrete languages

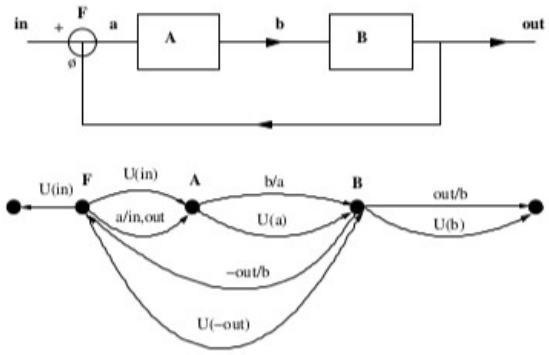


Fig. 1. Corresponding diagrams of control theory (above) and promise theory (below).

Definitions

Definition 1 **Agent policy:** The policy of an individual computing device (agent or component) is a representative specification of the desired average configuration of a host by appropriate constraints.

Promise netwok values

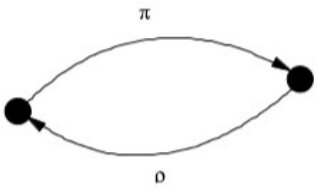
Definition 2 **Received Value**: Any function of the probability of keeping a promise, measured over the duration for which the promise is constant, in any currency/units. e.g. average quality of service over the time of a service level agreement. Only the recipient of a promise perceives this value.

Definition 3 **Bargaining Value:** The value of mutually traded promises, in which the behaviour of

a sender depends on its trading relationship with the receiver. This is the value of the game between the agents.

Game Theory

Game theory is about rational feedback relationships, in which agents have a selfish optimization interest. One of promise theory's useful properties is its close relationship with discrete bargaining games.



A game is a pair of back-to-back promises of comparable value that reinforce one another, i.e. increase the probability of promises being kept.

Consider the first of our models (see fig 1); the two interdependent promises are:

$$n_1 \xrightarrow{q=q_1(t)} n_2$$
$$n_2 \xrightarrow{q=q_2(t)} n_1 \quad (6)$$

where,

$$q_1(t+1) = \alpha_2 q_2(t) + \beta_2\, q_2(t) q_2(t-1)$$
$$q_2(t+1) = \alpha_1 q_1(t) + \beta_1\, q_1(t) q_1(t-1) \quad (7)$$

giving us two coupled equations for the time-development of the service level promises, and hence the average development of policy for the two peers.

The mutual agreements have two terms; these can be given a game-theoretical interpretation:

$$q_1(t+1) = \underbrace{\alpha_2 q_2(t)}_{\text{tit-for-tat}} + \underbrace{\beta_2\, q_2(t) q_2(t-1)}_{\text{tit-for-2tat}} \quad (8)$$

The tit-for-tat term says that we should try to reciprocate the service level that our peer gave us on the previous round. The quadratic term is like $q_1(t)$AND $q_1(t-1)$ says that we should add on a goodwill term depending on whether the peer has been faithful for the previous two rounds, i.e. we are rewarding its generosity.

This form of the game interaction emphasizes the connection with the theory of cooperation

---

Policy-based management

Policy is identified as a specification of the

average configuration of the system over persistent times.

An important aspect of this definition is that it allows for error tolerances, necessitated by randomly occurring events that corrupt policy in the system management loop. There is a probabilistic or stochastic element to system behaviour and hence policy can only be an average property in general.

Promise Theory

Promise theory is a new theory about what can happen
in a network of entirely autonomous components

Rather than adopting the conventional belief that “only that which is programmed happens”, it takes the oppositeviewpoint: “only that which is promised can be predicted”. It therefore approaches management from the viewpoint of uncertainty with realism rather than faith in compliance. In promise theory, one assumes a service viewpoint of interactions between computing devices. Each node offers to play a part in a collaborative network by promising to constrain its behaviour in various ways. A typical use of promise theory is to examine the steady state behaviour of a number of autonomous agents (components). Promise
theory does not necessarily fully determine the behaviour of its components, it only represents constraints within a set of defined behaviours. It is not limited to linear promises.

Promise theory takes a service viewpoint of policy and uses a graphical language to compose system properties and analyse them.

A key feature of promises is that they make a clean separation between what is being constrained and to whom a constraint applies. This is another area in which confusions can arise in policy theory. The form of a promise is: [Cf. π-calculus]

Properties

Property 1 - Forward Dependency promise

A dependency promise requires the recipient of the service to receive assurance that the dependent service will be used.

$$n_1 \xrightarrow{\pi_d} n_2 \otimes \left(n_2 \xrightarrow{U(\pi_d)} n_3 \oplus n_2 \xrightarrow{\pi/\pi_d} n_3 \right) \Rightarrow n_2 \xrightarrow{\pi} n_3, \quad (3)$$

where $\oplus$ represents the parallel composition of promises. We use the following shorthand for this:

$$n_1 \xrightarrow{\pi_d} n_2 \otimes n_2 \overset{\pi/\pi_d}{\not\to} n_3 \Rightarrow n_2 \xrightarrow{\pi} n_3 \,. \quad (4)$$

Example 1: For example, see fig. 2. Consider a webserver that depends on a database.

$$a \xrightarrow{\text{db}} b \otimes b \xrightarrow{\text{use-db}} c \oplus b \xrightarrow{\text{www/db}} c \Rightarrow b \xrightarrow{\text{www}} c \,. \quad (5)$$

The server b makes a conditional promise to the client c which is verified by the agreement about database support from a, and thereby is turned into a real promise.

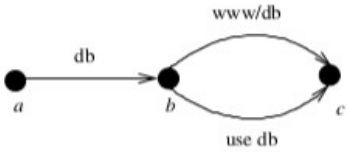


Fig. 2. Turning a conditional dependency into a real promise. The necessary structure is shown in graphical form. Here c receives the assurance that its supplier b will use a dependent service from somewhere, when offering a conditional promise.

This additional wiring might seem like an unnecessary burden, but in fact it is a liberating clarity. Many inconsistencies occur in policy systems from the assumption of a compliance that is not given. By making compliance explicit through declaration, we avoid hidden assumptions

---

Swarm intelligence

Swarn intelligence has been used primarily to motivate algorithms that are based on the observed behaviour of collaborating insect workforces, such as ants and termites. Other forms of swarm behaviour include fish shoals and flocks of birds that navigate together.

Ants and insects often leave pheremone trails and other signals that guide the behaviour of their collaborators, i.e. they provide a service, and their collaborators promise to pay attention to this signal, which is passed through an environmental intermediary channel.

An agent that conceals its presence cannot form a swarm,but an agent t hat is visible can. Other agents can then promise to follow an agent if they see one. Thus a swarming agent promises to be visible. This results in an exchangeof promises that canbe mapped onto a bargaining game. When an agent can see many neighbouring agents and promises to follow each of them, there could be a conflict. We do not have time to discuss the resolution of this issue here. What is clear however is that there is an implicit agreement taking place even in a mutually navigating herd or swarm. An

autonomous agent can choose to follow any other, or not, at its option: that is its autonomous right.

Promise theory therefore elucidates obvious properties of swarming behaviour in a natural way. For this reason, we believe it will be important in modelling not only computer networks but biological networks.

Parent Item: Towards topic maps for a promise theory based configuration management

Background

Capturing expert information in a
knowledge base that allows non-experts to locate it requires a carefully designed knowledge model. We attempt to discover whether the relationship between Topic Maps and Promise Theory can make configuration knowledge management easier, due to the promise model itself. Topic Maps is an ISO standard for representation and interchange of knowledge. Promise Theory on the other hand is a modeling approach which can be used to model a number of other things including a Policy based configuration management.

Configuration management is a process of establishing and maintaining the right values of configuration parameters in order to yield a human-computer
system that conforms to its expected behavior or its Policy state. Promise Theory is a modeling approach that can be used to model a number of other domains including the area of configuration management.

Tags: Ontology, Promise Theory, Topic Maps

Cfengine 3

A reference implementation of Promise Theory

2 approaches :

- **Centralized** : will forcefully push the centrally prepared rules and regulations with or without the will of the end user hosts
- **Policy Based** : works such as in Simple Network Management Protocol (SNMP), but gives autonomy to the agents in a way that they have the full right to pull and implement a centrally pre pared set of configuration policies

Theory, every autonomous agent will make a Promise about its expected behavior based on its choice. That is what makes Promise Theory optimal for a Policy based management framework.

The theory of promises describes policy governed services, in a framework of completely autonomous agents, which assist one another by voluntary cooperation alone

In Cfengine3, as a Promise Theory based configuration management tool, every configuration item will make a promise about its own characteristics and its relationship with other configuration

items.

A promise is a specification of future state or behavior from one autonomous agent to another. It is thus a unit of policy.

We want to be able to promise that the system is correct, verify this and only make changes if our promises are not kept. If you want to think ITIL, think of this as a service that Cfengine provides.

=> continuous check up of the state of each configuration item against its respective Promises to assure a continuesly policy conformant computer system in addition to the initial implementation of those Promises.

"passwd" example

In the body part of the Promise for example, the value of 644 could be assigned for permission attribute of passwd file. That means, the promiser passwd file will promise to have that specific value for its permission attribute.

File permission set is a good example of configuration parameters. The wrong set of values of a permission of a file could matter the configuration status of the whole computer system.

Conclusion

The resulting knowledge management scheme which is based on the combination of the two specified approaches is expected to:

- simplify information search.
- save time that could be spend on information search
- enable timely reaction towards critical things such as system failures.
- result in the representation of domain knowledge which is highly learnable even by the non experts of the domain under consideration.
- prevent possible knowledge lose that could happen when experienced employees quit their jobs.

Future work

Focus on designing the ontology

Configuration/ Knowledge Management

TODO: Add to bibilio

- D. Xiao and H. Xu. A common ontology-based intelligent configuration management model for IP network devices. Innovative Computing, Information and Control, 2006. ICICIC ’06. First International Conference on P. 385-388 ..(2006).
- Goodwin and D. Russomanno. An ontology-based sensor network prototype environment. (2006). http://www.cs.virginia.edu/ ipsn06/WIP/goodwin1 568983444.pdf.

Configuration management is the process of constraining the behavior of a network of machines so that each machine’s behavior conforms to predefined policies and guidelines and accomplishes predetermined business objectives.

Involving :

- People
- Zero or more configuration management tools (ex: CfEngine)
- Set of interconnected computers
- Set of configuration processes intended to give a policy conformant system as an output

Examples of configuration parameters include permission of a file, address of a network interface card or the type of file system that could be mounted on a disk volume.

The possibility of automating configuration of Internet Protocol(IP) networks by using a semantic technology is also found to be promising. A related endeavor of creating an ontology based sensor network environment was also reported to be viable.

Most of the organizations today have no systematic way of managing their configuration information. That means, configuration information such as IP address of a machine and its network service status are stored in a scattered and unorganized manner. Some monitoring tools such as Nagios have done some work in the presentation of monitoring information from different sources in an organized manner.

The idea of using CMDB is also a great work done so far on solving the above problem. However, although CMDB is a great work done so far and also a very important aspect of integrated knowledge management, the need of having a higher level knowledge management approach in addition to CMDB has been stated in different literatures.

This new way of enhancing knowledge management efficiency in the area of configuration management, requires the possibility of automatic data extraction from previous storage locations and possibility of automatic updating of permanent storages such as the CMDB in order to increase the quality of the underlying information. The possibility of those two features namely, automatic data extraction and also an automatic updating of the stored information were proved by ITIL.

This thesis work is about showing the possibility of having an integrated configuration knowledge management by the combined effort of Topic Maps and Promise Theory. The structure of configuration information that is stored in a CMDB and other documents will be represented according to the standards of Topic Maps. That domain knowledge structure will be implemented as a Promise of knowledge structure by using Cfengine 3.

Definitions

Definition 1 **Fact:** something which actually exists and its existence could be verified in some way.

Definition 2 **Information:** communicated fact or facts with a context attached to it in order to be meaningful to the receiver.

Definition 3 **Knowledge:** set of information interpreted in some way to form an understanding.

Definition 4 **Knowledge representation:** a surrogate of a domain of discourse through a formal description of its building blocks in such a way that it could be understood by any one interested.

Definition 5 **Domain ontology:** The concepts with in a given domain and the relationship among those concepts.

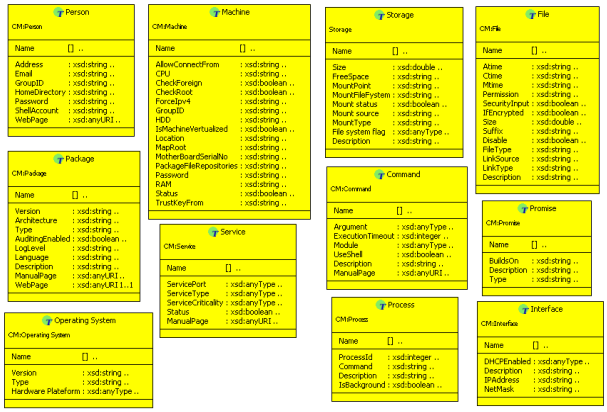
Definition 6 **Meta data:** Description of an information resource.

Definition 7 **Semantic web:** an extension of the World Wide Web with an enhanced information management scheme based on metadata.

Definition 8 **Configuration Management:** is the process of assigning the desired values to systems configuration parameters and a continuous look up over the behavior of the system in order to make sure that it has an acceptable level of performance.

Definition 9 **Configuration parameter:** is a property of a computer system that will play a role in determining the configuration status of the whole system.

Ontology/Topics specific concepts :



Definition 10 **AllowConnectFrom**: list of IPs or hostnames that may have more than one connection to a server port.

Definition 11 **CheckForeign**: tells whether there is a need for checking permissions on the root directory during depth search.

Definition 12 **CheckRoot**: list of host names or IP addresses to grant full read-privilege on a server

Definition 13 **ForceIpv4**: tells whether there is a forced use of IPv4 in connection

Definition 14 **IsMachineVirtualized**: tells whether a machine is virtual or not.

Definition 15 **PackageFileRepository**: a list of machine-local directories to search for packages

Definition 16 **TrustKeyFrom**: list of IPs or hostnames from whom a machine will accept public keys on trust. Definition of occurrance types that are related with the topic type stor- age are listed here below. A storage is a logical parition of a physical storage medium.

Definition 17 **FreeSpace**: absolute or percentage minimum disk space that should be available on a storage before warning.

Definition 18 **MountType**:Protocol type of remote file system

Definition 19 **FileSystemFlag**: List of menu options for bsd file system flags to set. Definition of occurrance types that are related with the topic File are listed below.

Definition 20 **Atime**: Range of access times (atime) for acceptable files

Definition 21 **SecureInput**: shows whether input files are writable by unauthorized users.

Definition 22 **MoutType**: Menu option for type of links to use when copying such as symlink and hardlink. Package related occurrence types are defined here below.

Definition 23 **AuditingEnabled**: shows whether log autding feature of a package is enalbed or not.

Definition 24 **LogLevel**: the reporting level sent to syslog.

Definition 25 **Architecture**: the architecture for package selection such as ”x866 4”. The rest of the definition for the occurrence types of the topic types: promise, person, and command are presented here below.

Definition 26 **BuildsOn**: a list of promise bandles that a promise builds on or depends on somehow (for knowledge management).

Definition 27 **HomeDirectory**: is a directory which contains the personal files of a computer user.

Definition 28 **ShellAccount**: is a personal account that gives a user access to a Unix shell.

Definition 29 **UseShell**: shows if a command is embeded in in a shell environment.

Definition 30 **Module**: shows whether to expect the cfengine module protocol.

Discussion

Results

- The resulting system was found to be good at usability especially in places such as big data centers. However, in cases of smaller size information domain, the evaluators commented on the easiness of using other methods of searching such as commands to get specific information.
- Timeliness of information search was also found to be good with a given condition that there is the proper kind of underlying infrastructure required for processing the search queries at an acceptable speed.
- The system is believed to scale well if there is a way of scoping the information search to get the desired result. That means, in a large sized information domain, a specific information search could result in a large size of search results . This will add more work on the user in getting the specific information. Therefore, with a condition of having additional features such as aggregation and scoping of topics, the system is believed to scale well for larger size of information domain.

Down sides

- The use of either scopes or name types for avoiding the possible ambiguity that could happen in relation to topic names was found to be veryconfusing.
- Since topic map is on its infancy, getting topic map experts as well as related literatures was found to be very difficult.
- The full application of all Topic Maps features was found to be very complex. The effort of choosing the minimum features that can yield in an acceptable topic map based representation was also found to be very confusing.
- Lack of formal query language by the tool used for implementation, namely Cfengine.
- Lack of thorough test due to time constraint
- Lack of standard Methodology for the development and evaluation of Topic maps

incompleteness of the designed ontology has also become one of the downside of this thesis work.

Databases

The possibility of using databases to the storage and retrieval of information was considered and found to be very important in having a permanent storage for a large size of configuration information. Although databases plays a great role in restructuring of information, their limitation in enabling a semantic representation of knowledge has been outlined by many authors including. That is why the need and possibility of extending databases by using topic maps is underlined by the author of this literature. The extension of databases with Topic Maps is expected to increase the usability of the underlying data through values that will be added such as categorization and contextualization of domain information.

Web Pages

The use of web pages for information restructuring was also considered and found to be less useful in increasing find ability of information. The draw back could better be explained in such a way that a query for a single topic could result in a page with further work of finding the specific subject to be done by the user. That is why some literatures call such systems as information locaters rather than information finders. The user will only get help in knowing where the information is rather than the exact answer for the specific
request.

ITIL (Information Technology Infrastructure Library)

The way configuration information organized and stored was a total mess until the recent move to wards the notion of using Configuration Management Database (CMDB). In data centers, with thousands of machines, tracking information about things like files permissions, is a challenging and complex task. That is why an efficient management mechanism for configuration in formation is believed to be crucial by many others

CMDB is a federated hub for configuration information and it can be described as the heart of ITIL framework. Some ITIL practitioners described it as ”the ultimate source of truth”. All ITIL processes rely on CMDB to furnish them with trustworthy configuration item (CI) information, relationships, and interdependencies.

CMDB has a lot to do with solving the problem caused by the lack of integrated knowledge management, it is also claimed to have downsides when the size of information gets bigger. That draw back of CMDB is related to its syntax based approach of information management. That is why ITIL has started a related work on extension of CMDB with a semantic based knowledge management scheme.

Knowledge Representation

1. Keyword matching search = Since this type of search is based on syntax rather than semantics, in most cases, it will result in high recall and low precision of search results. In other words a search engine can return high number of less related or unrelated documents in a syntax based search.

2. Metadata attribute classification = Taxonomy is a form of controlled vocabulary where the terms are related to the concepts based on hierarchical relationships. Taxonomy is subject based classification that arranges the terms in the controlled vocabulary into a hierarchy without doing anything furtherThat is, thesauri extend taxonomies, by adding more built-in relationships and properties.

3. Semantic based knowledge representation = quality of search and integration capability of heterogeneous information. Is preferred to increase find ability of information in a Promise Theory based configuration management domain.
   - Summarized but important information search results will be possible
   - Automated reasoning could be achieved
   - Knowledge will be organized in conceptual space according to its meaning
   - Query answering over several documents will be supported
   - Semantic interoperability will be possible through ontology mapping
   - The accuracy of web searches will be improved including the problem of missing important information.
   - Differences in terminologies will be resolved using standard abstract data models.

=> Encode meanings independently of the underlying syntax. This will enable machines as well as people to understand, share and reason semantically. Using semantic technologies, adding, changing and implementing new relationships can be highly simplified compared to the traditional approaches. In traditional knowledge representation systems , meaning and relationships must be predefined. This will make them less flexible for possible changes.

Tags: Ontology, RDF, Semantic Web

Metadata

Meta data is an old concept that is becoming very popular in relation to its application in enhancing the quality of information search. Among other attributes, a meta data of an information recourse can comprise of its creation date, author and so on. Dublin core, one of the main metadata standards used to describe web resources, has fifteen properties in its element sets to describe an information resource. The fifteen properties including the two examples given above are subject to choice as per the need of the user.

The use of metadata to represent information resources increases the quality of information search in such a way that a document will be searched based on its meaning as documented in the metadata than its syntax.

Methodology

The heuristic Methodology was designed based on the good features of the Methodologies used in Information Architecture and Database design. The five phases of this Methodology are:

- **Start up** :
  Clear understanding of the project by the Topic Map developer and the project stack holders
  - the project stack holders will be identified
  - the scope of the project will be determined
  - the vision of the project will be clearly set
  - potential data sources will be identified
  - the tools and approaches that will be used in the project will be identified
- **Conceptual design** :
  Enable the Topic Map developer gets clear understanding of the domain under consideration. The result of this phase is a conceptual model that is acceptable by the project stake holders with out any detail of underlying knowledge representation approach
- **Physical design** :
  Results in a model that represents the domain
  based on the underlying knowledge representation technology: Topic Maps. Mostly, the result of a Physical design phase is a domain ontology as it is in this present work
- **Implementation** :
  Codifying the designed ontology and populating the resulting Topic Maps with real data.
- **Evaluation** :
  - based on the expected functional requirements
  - based on its acceptability by the project stake holders in the Evaluation phase of the Methodology

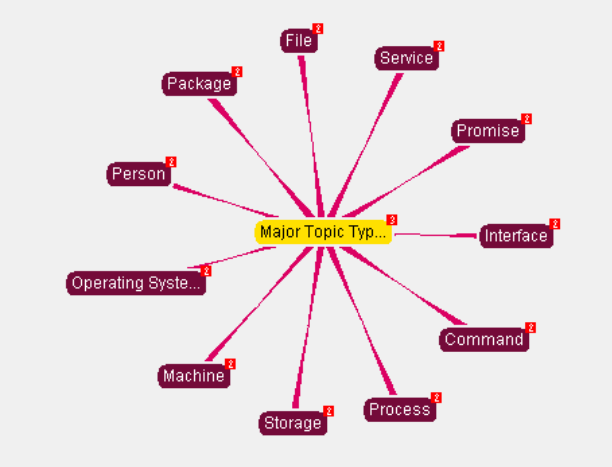
---

Ontology for Promise Theory based configuration management

Different kind of ontologies :

- **Task Ontology**
- **Method Ontology**
- **Application Ontology**
- **Domain Ontology**: defined for conceptualizing the particular domain such as a Promise Theory based configuration management domain.

Topic types



Machine is any device that can accept and process information to provide a desired result based on a program or sequence of instructions on how the data is to be processed.

Computers are the main types of machines for this thesis work. That is why the two terminologies are interchangeably used in different parts of this literature. Personal computers, laptops, server computers all belong to this topic type. A computer runs one or more operating systems.

Operating System is a program that bridges the gap between the package and the underlying hardware components of a computer. **Operating system** as a topic type consists of different operating systems as topics including Windows XP, OS X, and Linux.

Packages on the other hand are application programs designed to serve specific purpose. It could be to give a network service or web service on a machine. The package named Apache is a good example topic of type package which is designed to enable a web service.

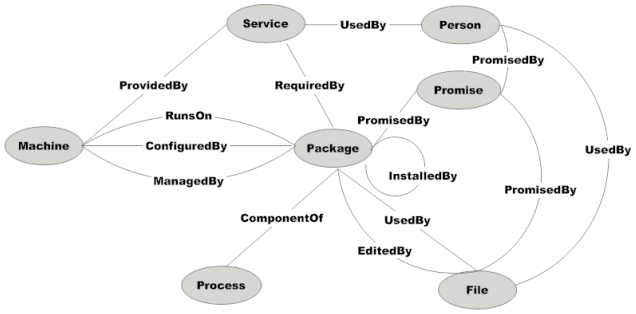
Service is a specific functionality of a computer system such as a web service or network service provided by a machine. A process which is defined in this literature as a package or part of a package on action enables the provisioning of services by performing real activities behind the scene.

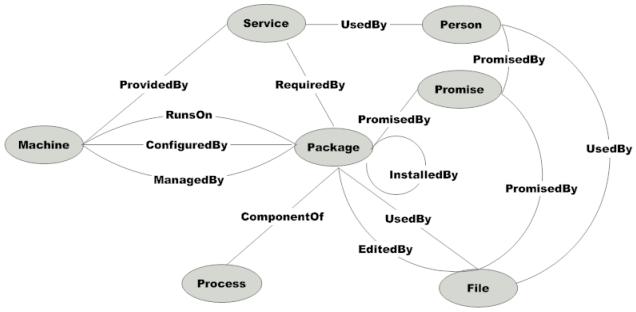
A **Command** is a utility that can be used by users to start a specific process if there is no schedule set for the same purpose. A very good example that can illustrate the relation ship among those topic types is that of a web service that needs a process such as ”httpd” running on the background which is started by a command:”httpd start”.

Storage is a logical partition of a physical storage media that mainly serves as a location for the different kinds of files. [33] has defined a file as a complete, named collection of information such as a program, set of data used by a program or a user created document.

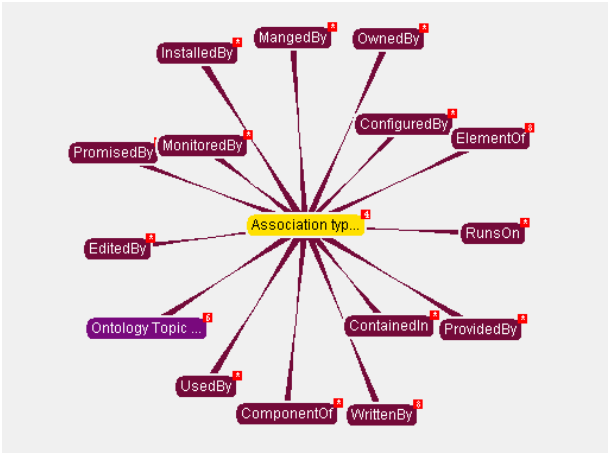
Interface is defined by [33]as a device used to provide network access to a computer. The definitions given by [33] fits the meaning of those topic types in this present literature.

Ontology Overview





Ontology Relations



- **Caused By**: an association type where one topic plays the role of affecting the other by changing its state from state X to state Y.
- **Configured By**: when an entity X can bring the necessary changes forentity Y in order to let it serve its intended purpose. The relationshipbetween a machine and a configuration management package such as Cfengine 3 is a typical example of such kinds of relationships.
- **Edited By**: writings such as a file could be amended by an editor such as user of the file for some purpose.
- **Managed By**: If X is responsible for the proper functioning of Y in order to let it accomplish its intended purpose, X is said to be the manger of Y. The act of managing includes a continuous supervision and action taking when there is a deviation from the desired way of behaving.
- **Monitoring By**: Monitoring is mainly the responsibility of keeping an eye on some thing. If an entity X monitors Y, it keeps an eye on Y so that it may report to the manager of Y when there is a deviation from the desired behavior.
- **Component** Of: The relationship between an entity X which is a component for another entity Y which plays the role of being a complex for thecomponent X.
- **Owned By**: Ownership is a kind of relationship that could exist between an entity which has become a possession and its owner. The relationship between a file and its owner is an example of this type of associations.
- **Promised By**: the relationship between a promise and its promiser.
- **Required By**: If X depends on Y in order to serve its intended purpose, the relationship between them is called RequiredBy.
- **Runs On**: If entity X performs its actitvities or execution on entity Y, the relationship is called as RunsOn.
- **Written By**: the relationship beween a writing X and its writer Y.
- **Used By**: If an entity X uses entity Y at any point in time for any of its purposes, the the type of association is called UsedBy.
- **Started By**: When an entity X makes another entity Y to start doing something or behave in some manner, the association is said to be of type StartedBy.
- **Provided By**: When an entity X has the potential and willingness of providing some thing to some other entity, the relationship that exists between them is called ProvidedBy.
- **Installed By**: An entity X can put program Y into a machine in order to let it serve its intended purpose. This relation ship between a program and an entity who put it into a machine is called as InstalledBy.
- **Contained In**: If an entity X is contained with in Y either physically or conceptually, the relationship is called as ContainedIn.

Promise Theory based Configuration Management domain

Inputs

The domain of a Promise Theory based configuration management takes high level configuration policy as its input. The high level policy will be broken into low level configuration specifications or what we call it
promises in a Promise Theory modeling approach.

The technologies required to realize this desire will be identified in the processes will follow the identification of inputs. In addition to this example, the following list of things could be forwarded as inputs for a Promise Theory based configuration management domain :

- The required number and type of machines
- Operating System architecture of each machine
- Packages that need to be installed on each machine
- The services that should be given by each host
- Storage related issues such as the type of file system
- Security issues including user management aspects

Processes

The high level configuration policies that serves as inputs will go through two major processes :

- Low level configuration specification. This process depends on the architectural information such as the device detail and the desired type of topology of a system.
- Enforce the promises made by each of the configuration items. This will make each configuration item to behave according to its promises. In managing changes, Cfengine 3 will take corrective action in addition to playing the role of informing concerned parties about promises that are not kept.

Outputs

-> A set of computers with the proper configuration values for their configuration parameters (promised values of their attributes). In addition to their own attribute values and way of behaving, a set of Promises about the right kind of relationships among those configuration items is expected as the result of those processes.

The output is a policy conformant computer system with each of its configuration items having the proper attribute-value pairs. Files, packages, disks, process, service and interface are the major configuration items found from the reference manual of Cfengine 3.

Promise Theory

TODO: Add to biblio

- Cfengine-AutoReference. Cfengine reference manual. (version 3.0.1b1). Technical report, http://www.cfengine.org/docs/cf3-reference.html (Acceded April 10, 2009).
- J. Bergstra and M. Burgess. A static theory of Promises (2008)
- An approach to understanding policy based on autonomy and voluntary cooperation. (2005), Lecture Notes in Computer Science, Volume 3775/2005, P. 97-108.

Promise Theory is a modeling approach for system co-operation introduced at Oslo University College. It can be used to model a number of other things including the domain of configuration management. Promises are at the center of Promise Theory.

A promise is an announcement of fact or behavior by a promiser to a promisee, observed by a number of witnesses (referred to as the scope of the promise), whose outcome has yet to be assessed.

2 types of promises :

- A promise to agree to behave like another: is essential for defining groups, roles and social structures with consensus behavior.
- A Promise to utilize the promise of another: is crucial for client-server interaction, dependencies and access control.

Characteristics :

- There must be agents in order for promises to exist
- There must be a promiser(or source agent)
- There must be a promisee (or receiving agent) which might be the same as the source.
- There must be a body which describes the nature of the promise.

Symbol	Interpretation
$a \xrightarrow{+b} a'$	Promise with body b
$a' \xrightarrow{-b} a$	Promise to accept b
$v_a(a \xrightarrow{b} a')$	The value of promise to a
$v_{a'}(a \xrightarrow{b} a')$	The value of promise to a'

Promise Theory + Topic Maps

TODO: Add to biblio:

- Cfengine on Topical Islands.(2009)., A preceding at Topic Maps Norway 2009 Conference.

Both Promise model and Topic Maps have the same basic world view: Principle for reduction of knowledge into atoms and autonomy of concepts that au-
tomatically avoids overlap and conflict. A Promise Model revolves around the autonomy of agents.

In a similar fashion, Topic Maps reduce knowledge structure to the level of subjects or concepts.

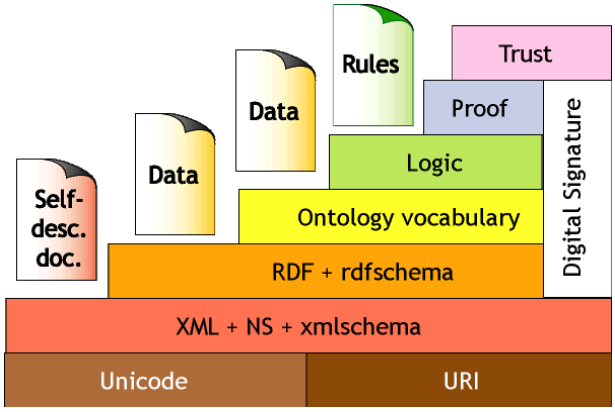
Cfengine 3 classes are also found to be equivalents of Topic Maps scopes. In Topic Maps one can define a scope by listing set of topics that could give a specific context. In the same way, Cfengine 3 has set of built in classes. In addition, Cfengine 3 users are free to define their own classes. That is the other similarity found between the two approaches.

Enabling a graphical navigation of a semantically represented information domain is the goal of unifying Topic Maps with Promise Theory

---

RDF

Technologies organized into layers to achieve the implementation of semantic web :



- URI (Uniform Resource Identifier) and Unicode aspects of knowledge representation.

- XML (Extensible Markup Language) is a language enabling web users to use their vocabulary for better semantic representation of web contents. XML + HTML = better visual and semantic representation of web contents

- RDF (Resource Description Framework) : represent knowledge in a machine understandable format for simple reasoning and other similar purposes. XML represents meta data or information about information. On the other hand RDF represents the information itself.

Topic Maps

ISO standard for knowledge representation and interchange since 2000, and comprised of 9 standards:

2 for interchanging semantic information :

- RDF: Machine approach
- TM: Point of view of human being

Indexes, glossaries and thesauri are all ways of mapping the knowledge structures that exist implicitly in books and other sources of information.

A topic map usually contains several overlapping hierarchies which are rich with semantic crosslinks like ”Part X is critical to procedure V.”

What is a topic? A topic, in its most generic sense, can be any thing, whatsoever, a person, an entity, a concept, really anything ,regardless of whether it exists or has any other specific characteristics, about which anything whatsoever may be asserted by any means whatsoever.

Same as ontology approach:

Type => Instance (ex: Person => Simon)

Occurrence type

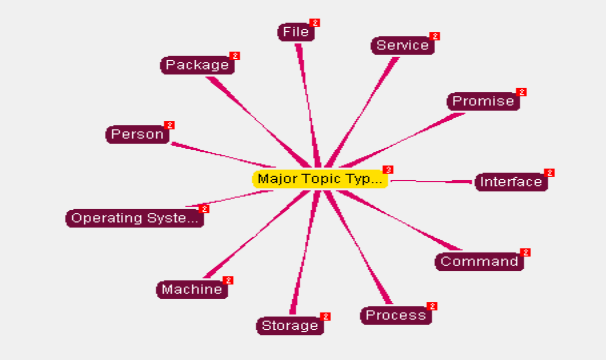
An **occurrence role** tells about the nature of the occurrence by stating whether it is a web page or an article written about the topic.

The **occurrence role type** on the other hand ex-
plains the relevance of that information resource to the subject in question bystating whether the document defines the topic or only mentions it in relation to some thing else.

- **Internal**: properties of a topic with the exception of name property as it has a special significance in the design of Topic Maps ontology.
- **External**: include any external information resource that could say something about the topic in question such as a web page. External occurrences have Uniform Resource Identifier (URI) as their data types.

Association type

A relationship between two or more topics is asserted by the existence of an association



Subject identity

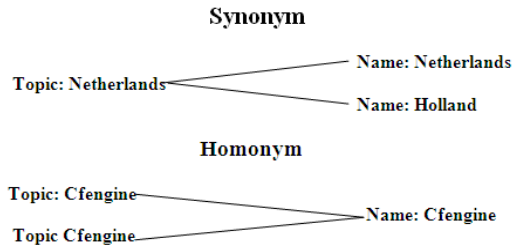
Topic Map is isolated: Identity unambiguous

Semantic Web: "Italia", "Italie", "Italien" = Same Topic

Scopes

A topic has name, occurrences, associations and roles as its characteristics that need to be defined in order to assert the existence of the topic. Scope determines the validity of topic characteristics. In principle, a characteristic of a topic has one or more scopes

Problem: Synonym vs. Homonym

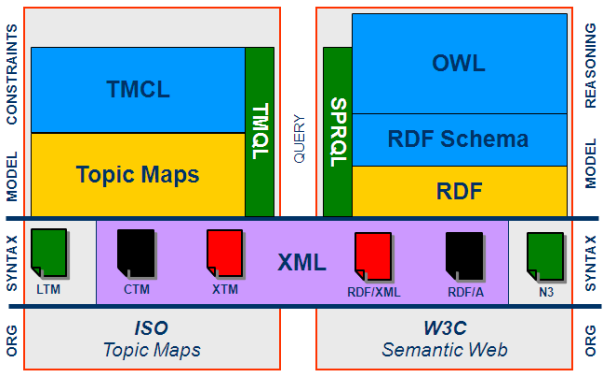


Topic Map Scope is defined by a set of topics.

Name types

Name obviously is one of the properties of a topic but, it has its own special role in topic maps ontology design that makes it unique from other properties of a topic. A name is the default characteristic to identify an entity from the other.

Topic Maps vs RDF



In addition to its optimality for restructuring of information from the perspective of humans, the reasons why Topic Maps are chosen as a knowledge
representation approach in this thesis work are listed here below :

- It is an ISO standard for knowledge representation and interchange with emphasis on the find-ability of information.
- It has a special relationship with Promise Theory
- It enables seamless navigation through information domain.
- It enables precision on information search and retrieval.
- There are enough and capable tools available that makes its implementation much easier
- It has good support for full text searches and complex queries
- It enables additional view of domain information through visualization