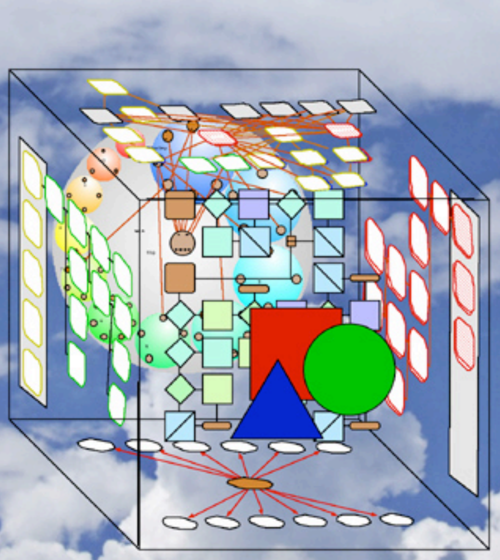
core[talk]

Cubicon

Trusted Infrastructure for the Ambient Cloud

Architectural solutions to NSTIC challenges



(cube of icons)

klausner@coretalk.net

Cubicon

Architectural solutions to NSTIC challenges

The NSTIC vision: Individuals and organizations utilize secure, efficient, easy-to-use, and interoperable identity solutions to access online services in a manner that promotes confidence, privacy, choice, and innovation.

Introduction

In this whitepaper we introduce a new **software architecture called Cubicon**. It utilizes twelve innovations that can help transform NSTIC's vision for a scalable Identity Ecosystem into an actionable strategy. By providing technology-based governance mechanisms that support evolving legislative policies, Cubicon can further assure the NSTIC objective is accomplished. Additionally, we believe that the scope of the strategy needs to be expanded to include a Personal Data Ecosystem, giving users the ability to control their own data.

The following list outlines NSTIC-focused features of our technology:

- Provides a single representation of a subject across multiple identity providers
- Provides a visible digital watermark to all internet-generated intellectual property for source verification
- Uses credentials and trustmarks to control access to a subject's personal data
- Uses certification to verify a subject's personal data
- Provides automatic reputation rating for each member in a trust framework

Cubicon is a **meta-standard technology** that can be used to automate the process of detecting and enforcing privacy policy infractions. Our technology displaces many implementation languages with an 'executable design' language that uses a finite set of language constructs, called language elements, that act like 'software DNA' to construct systematically reusable software components. In effect, Cubicon provides the foundation for a **Trusted Infrastructure for the Ambient Cloud**.

Additionally, Cubicon aligns with NIST's broader mission "to promote industrial competitiveness" by advancing the standardization of software. Standardization will provide large-scale software interoperability, required of complex "hyper-networks" such as the Smart Grid. Our approach is simple: view the entire computing stack *holistically* - from development through execution. Our mission is to drive the software infrastructure into the 21st Century, very much in line with NIST's objective to support innovative research and development in the computer sciences.

Internet Standards for General Systems Interoperability

There are numerous Service-Oriented Architecture (SOA) and Semantic Web protocols for identity and privacy matters, but no unifying standard between them. Consequently, a ***meta*-standard** technology is needed to automate the standards development process itself. In the following paragraphs we explain how current protocols ineffectively process service data and attribute metadata, emphasizing the need for alternative solutions:

Service-Oriented Architecture – SOA coupling protocols remain mired in unresolved governance, security, service-level agreement, enterprise service bus, and compatibility issues. Neither SOAP nor JSON can effectively support service data interoperability between Identity Ecosystem participants, as both standards lack the ability to sufficiently *automate* schema lifecycle and systems integration tasks.

Semantic Web – A fundamental requirement for attribute metadata interoperability between Personal Data Ecosystem participants is the *harmonization* of ontologies. Metadata represented in RDF/OWL triples cannot be automatically harmonized because a URL's combination of a concept's expression and representation inhibit the sharing of metadata across triple-stores. For example: two communities express a concept by naming an attribute 'train' and 'tran.' Although both the English and Spanish string representations are valid, sharing their meaning through a URI alias is awkward. It remains exceeding challenging to process billions of triples in heterogeneous triple-stores, and link them in a meaningful way across the cloud.

Introduction to the Federated Web

Cubicon's architecture supports the **Federated Web (fWeb)**: a WWW-compatible internet layer that effectively connects 'everybody' and 'everything' through the use of **context processing**. Context processing allows a machine to 'understand' human intention more clearly by performing an inference process similar to the one used by the brain. For example, the intended meaning of the concept 'fence' becomes more certain as it is related to one or more other concepts:

- Fence: white, picket, post
- Fence: white, uniform, saber
- Fence: white, hot, merchandise

FWeb protocols use community-based ontologies to provide the grounding of concepts, required to help machines understand, and later establish, the connections and relationships between people and between things. In the fWeb, a community is controlled by its membership, comprised of **entities**. We define an entity (NSTIC subject) as an individual, company, organization, or government.

Currently, the IETF is sponsoring projects to support authentication and encryption capabilities in layers 3 and 4 of the OSI model to improve router, domain, and protocol security. To improve security and privacy, Cubicon supports transparency, interoperability, and portability in layers 5 through 7 by placing all machine and entity resources into context. (Figure A)

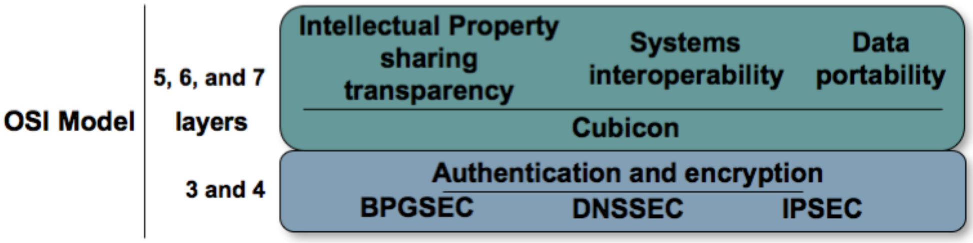


Figure A: OSI Model layers

Impact of Cubicon on the Identity Ecosystem

The Identity Ecosystem Framework is the base structure that incorporates all participating trust frameworks, developed by communities whose members share similar goals and perspectives. (Figure B)

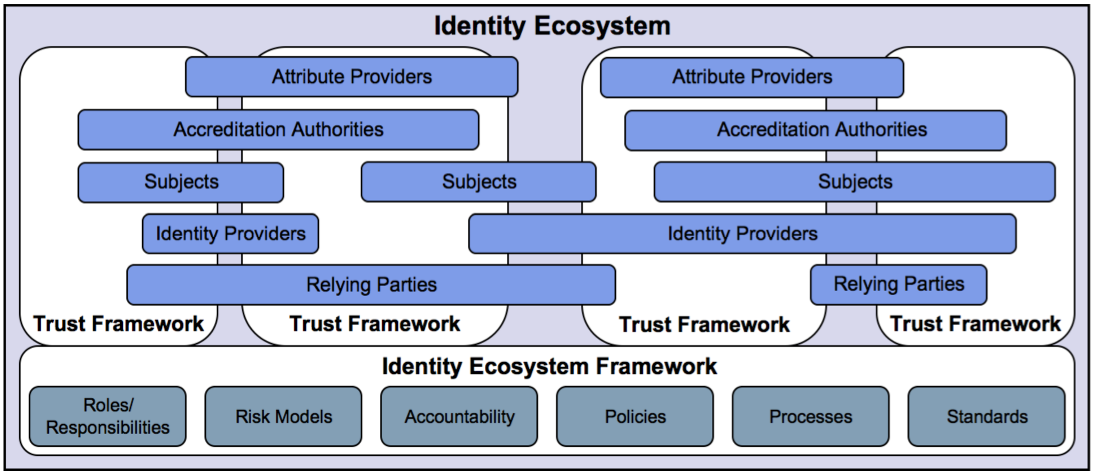


Figure B: Current Identity Ecosystem

Four key benefits are provided to the NSTIC Identity Ecosystem with the introduction of the Cubicon architecture, outlined below. (Figure C)

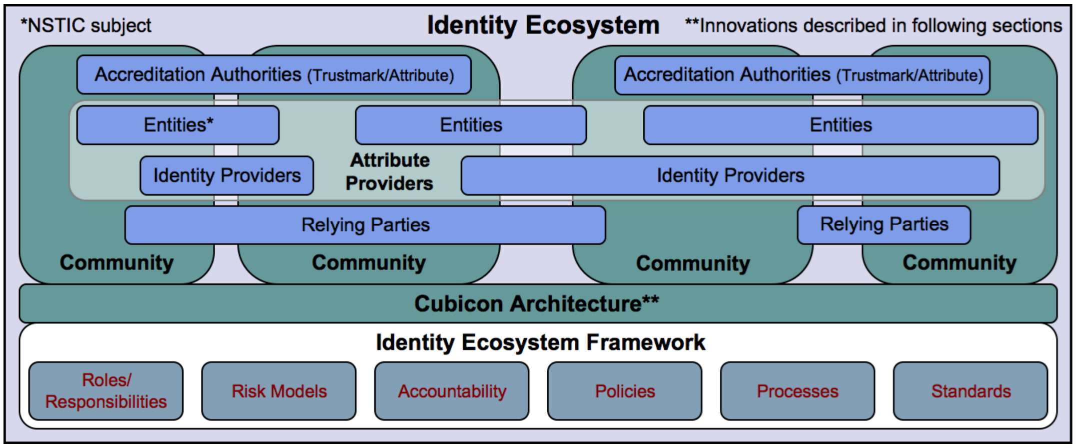


Figure C: Cubicon-enabled Identity Ecosystem

Architecture – To support community governance, Cubicon:

- Automates the community building process to manage a member's **roles and responsibilities**
- Consist of software agents that maintain **risk models** to remotely authenticate and manage digital identities
- Assures service provider **accountability** for online interactions, accessible to the user
- Constrains a user's transactions/interactions with service providers to provide automatic enforcement of legislative **policies** **–** *The code is the law*
- Automates **processes** that manage, authenticate, authorize, and validate identity data
- Unifies identity and privacy **standards** across the Identity and the Personal Data Ecosystems

Community formalization of trust frameworks – **Community** is an architectural tenet of Cubicon. In the fWeb, each community has a repository that provides governance over their members' activities in, and across the Identity and the Personal Data Ecosystems.

Assimilation of the attribute provider role – NSTIC's guidelines specify that an attribute provider should be responsible for establishing and maintaining a user's personal data, however we believe the user should be given the right to control their own data. Users will have the option of hosting their data in an identity provider's cloud or any device under their control.

Expansion of the authentication authority role – The fWeb goes beyond traditional authentication practices by supporting the certification of *trustmarks* and *attributes*. Currently, a **relationship-based identity system** can be used to perform the certification process. These types of systems have not gained widespread adoption, due to their inability to provide provenance for metadata. Cubicon's architecture establishes provenance by permanently fusing a language element to a particular entity at design time.

Cubicon's Architectural Innovations

The Cubicon architecture incorporates twelve innovations that can contribute to transforming NSTIC's *vision* into an actionable *strategy*:

Directory – Architectural Innovations

1) **Language Elements:** *Formalization of software abstractions*
2) **Context:** *Formalization of general systems above the object level*
3) **Community:** *Formalization of the software development life-cycle process*
4) **Micromachines:** *Automation of the programming process*
5) **Relationship-based authentication:** *Supports trustmark and attribute certification*
6) **Globally Typed Identifiers (GTIs):** *Contextual representation of resources*
7) **Secure service interactions:** *Modeling and execution of services*
8) **Topic Map:** *Intuitive modeling of meaningful relationships*
9) **Mobile Agents:** *Perform tasks on behalf of users*

10) Security Model: *Supports trust, behavior, and access safeguards*

11) Open Design: *A viable software market model beyond Open Source*

12) Detecting Privacy Policy Infractions: *Monitors compliance*

1) Language Elements: *Formalization of software abstractions*

Twenty-one types of language elements (constructs) are recombinant across six layers of abstraction to form software components that comprise all fWeb applications and apps. (Figure D) To further clarify: **language elements** combine to form **contexts**; contexts combine to form a **framework**; and frameworks combine to form **applications** and **apps**. All components are represented as **icons**. Icons provide a visual artifact that will allow software systems to be simulated much like mechanical systems are simulated today.

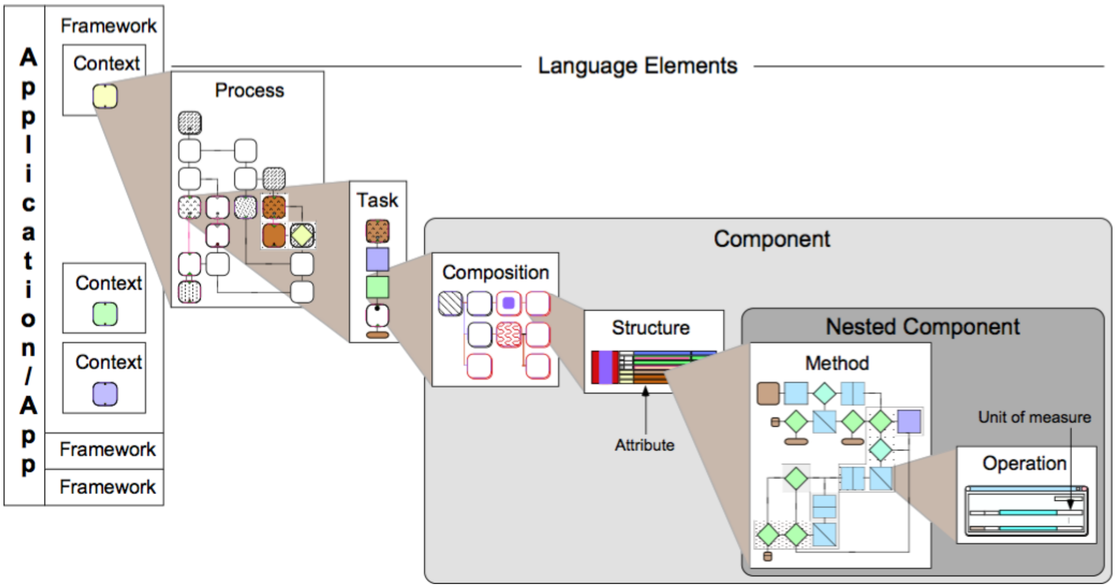
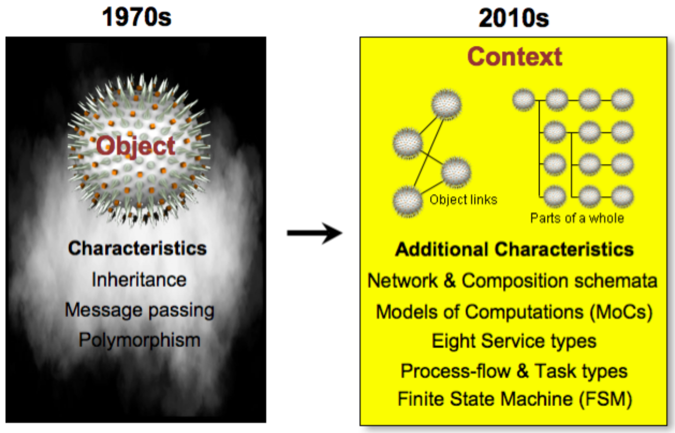


Figure D: Layers of a Cubicon software component

Cubicon software components remain recombinant in and across all fWeb communities. The recombinant nature of Cubicon software components will support the adoption of applications/apps, content, and services between Trust Frameworks (communities).

2) Context: *Formalization of general systems above the object level*

In Cubicon, the semantics (meaning) of an object is placed into a container called a **context**. Context formally captures five additional characteristics of general systems above the object level. (Figure E) These characteristics make it easier to understand another developer's intent, identify relevance of existing work to new efforts, and pinpoint redundancies. **Trusted computing** is the next paradigm leap since the introduction of object-orientation programming in the 1970s.


1970s
Object
Characteristics
Inheritance
Message passing
Polymorphism
2010s
Context
Object links
Parts of a whole
Additional Characteristics
Network & Composition schemata
Models of Computations (MoCs)
Eight Service types
Process-flow & Task types
Finite State Machine (FSM)

Weak organizing principles beyond an object's attributes and methods

Strong organizing principles package objects into general systems

Figure E: Cubicon context container

3) Community: *Formalization of the software development life-cycle process*

In the fWeb, a community is formed by a single entity who can later invite others to join as members. The community's collective intellectual property (IP) is represented in language elements, stored as meta-objects in a repository. (Figure F) **Each meta-object**, composed of multiple attribute slots, is an **executable software design**. At software release, instances of meta-objects stored in the **Community Repository** are compiled into instructions for download into any member's device running a **Cubicon Virtual Machine (VM)**. When data is moved to a device under the control of a different user, the device can request processing instructions for the data's type from the originating Community Repository down to the granularity level of an attribute and a unit of measure.

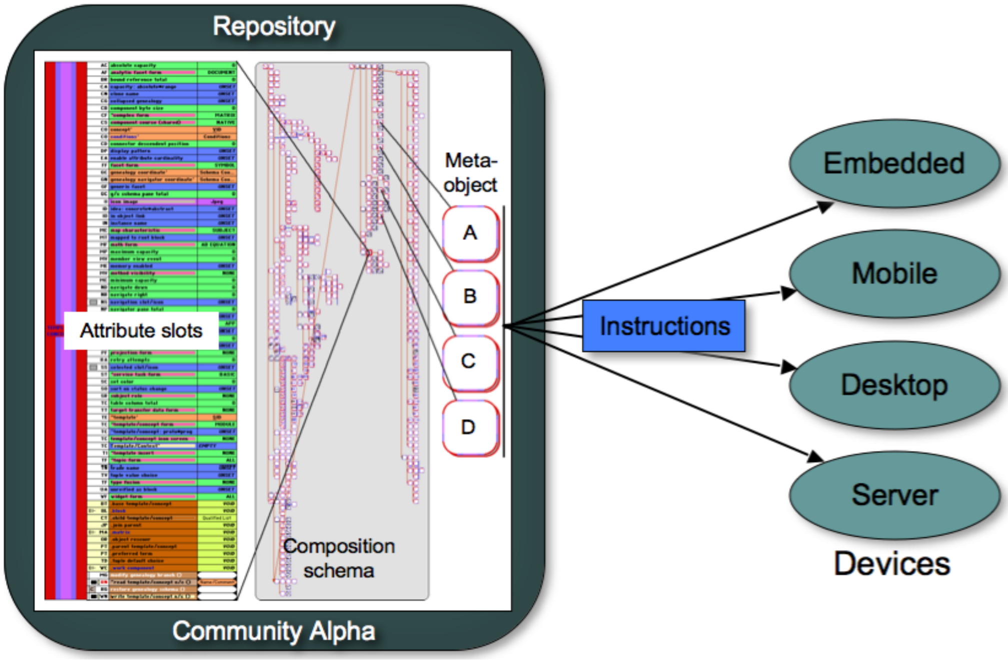


Figure F: Community Repository

The **formalization of community** supports *IP sharing transparency*, *systems interoperability*, and *data portability* without compromising security and privacy. Community is the missing piece of the Semantic Web.

4) Micromachines: *Automation of the programming process*

The idea of a **micromachine** is borrowed from the nanotechnology field where micro-mechanical machines perform some useful task. During programming, micromachines automatically execute repetitive tasks that are currently a programmer's responsibility. Micromachines can also greatly increase productivity, and expand the population able to develop software. (Figure G)

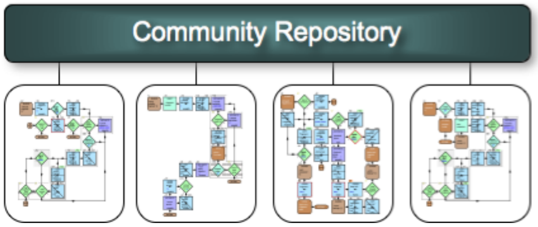


Figure G: Micromachine tasks

5) Relationship-based authentication: *Supports trustmark and attribute certification*

The fWeb provides the technology to **establish and trace provenance** for an entity's identity and intellectual property (IP). The fusion of trustmark and attribute language elements to a particular entity forms the foundation for our relationship-based identity system. Trustmarks are used to indicate a service provider has met the minimum reputation requirements set by a particular community. A Cubicon-supported trustmark is called an **iSeal**. (Figure H)

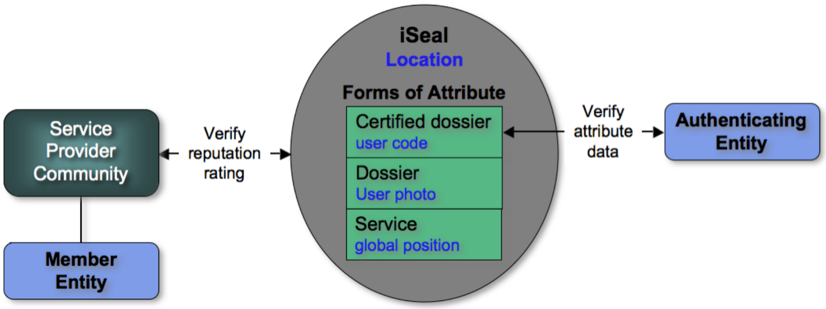


Figure H: Relationship-based identity system

An example of a type of an iSeal is 'Location.' The Location iSeal consists of user code, user photo, and global position, attributes. iSeals rate a member's reputation, based on requirements established by peers and expressed as an algorithm. Entities may hold an iSeal from each community they are an active member of.

A personal data locker contains two forms of attributes: **dossier** and **service**. A *dossier* attribute represents a persistent value about the user (such as user photo), whereas a *service* attribute represents a transactional value (such as global position). A user's self-asserted attribute can be submitted to an authenticator for *certification* (such as user code). Subsequently, any third party viewing the attribute can decide if they trust the authenticator's certification. As an alternative to a self-asserted attribute, a relying party may write an attribute into a user's dossier (such as healthcare data).

6) Globally Typed Identifiers (GTIs): *Contextual representation of resources*

Currently, Semantic Web linked data is represented by Uniform Resource Identifiers (URIs). In the fWeb, concepts are represented by Globally Typed Identifiers (GTIs). (Figure I) Each GTI is expressed as an icon and name, easily read by humans in graphical and text patterns. These *expressions* are bound to an identifier *representation* composed of locator, identity, and type integers. This binding is used for all twenty-one language element types.

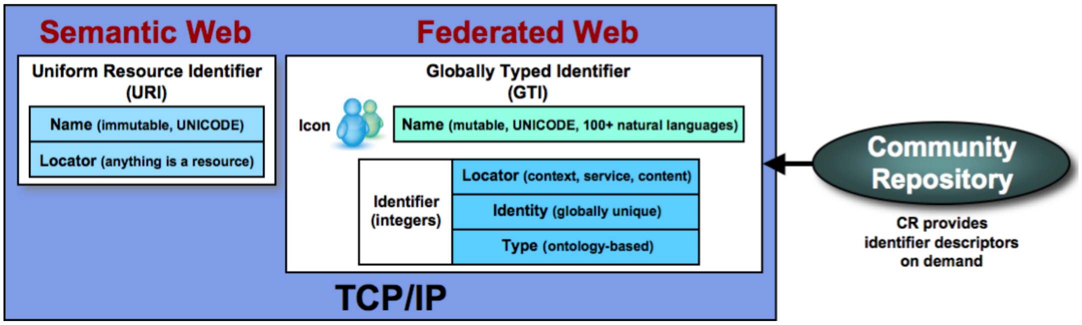

Semantic Web
Federated Web
Uniform Resource Identifier (URI)
Name (immutable, UNICODE)
Locator (anything is a resource)
Globally Typed Identifier (GTI)
Icon
Name (mutable, UNICODE, 100+ natural languages)
Identifier (integers)
Locator (context, service, content)
Identity (globally unique)
Type (ontology-based)
TCP/IP
Community Repository
CR provides identifier descriptors on demand

Figure I: Globally Typed Identifiers

The processing of GTIs is several orders-of-magnitude more efficient than the processing of URIs because each *numeric-based* identifier requires far fewer machine cycles to resolve a value than a *string-based* one: it is the difference between executing a **compiled network system** versus an interpreted network system.

GTIs allow machines to perform concept harmonization by tracing each concept's genealogy back to its origin Community Repository. Harmonization is the method used to contextualize a semantic network of concepts.

7) Secure service interactions: *Modeling and execution of services*

The fWeb replace XML, a verbose string-based schema, with a **service**: a compact binary composite schema. A composite schema is composed of nested collections of record structures that may contain strings. (Figure J) A service supports the effective exchange of data between Identity Ecosystem participant's systems.

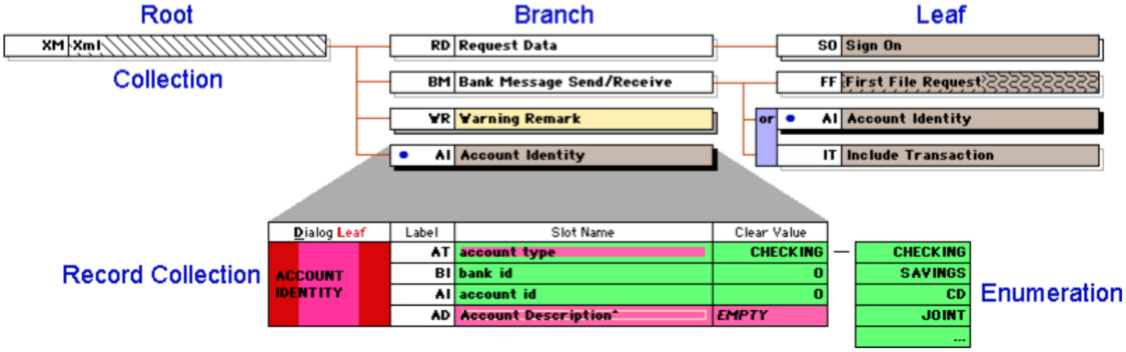


Figure J: Composite schema

An fWeb service:

- Requires community membership to read
- Improves data stream throughput over low bandwidth channels
- Applies partial or whole encryption

- Requires limited memory allocation to contain the service during service processing
- Performs application-level Deep Packet Inspection (DPI) with no parsing

Domain experts can collectively develop a service schema through community consensus in the **Cubicon Integrated Development Environment (IDE)**. Using this method, programmers are essentially removed from the development process, except to code connectors in and out of the schema.

8) Topic Maps: *Intuitive modeling of meaningful relationships*

While a desktop's directory tree is a useful way to *organize* things in files, it does not organize the *relationships* between people and between things in a meaningful way. To overcome this limitation, we use **Cubicon Topic Maps*** in a substrate that is integrated into the fabric of the fWeb. Each Topic Map contains topics that are associated by their meaning (semantics). Topics convey **universal knowledge**: defined as 'a priori' knowledge - "before the fact." Several examples of universal knowledge are E=MC^2, a user's birthday, and the length of a meter.

There are five forms of resources in the fWeb, and four forms of resources in the Web. (Figure K) Each topic can have zero or more links to an fWeb/Web resource, called an **occurrence**.

*not ISO standard Topic Map

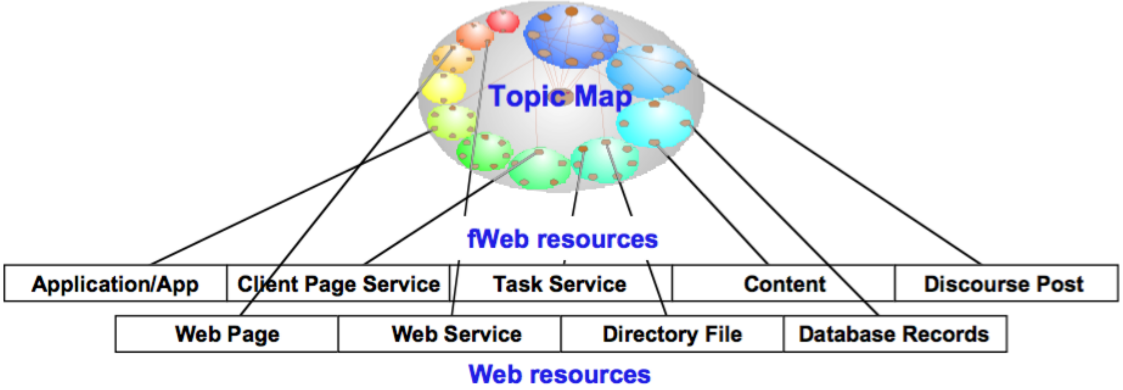


Figure K: Topic Map occurrence of resources

As opposed to a directory tree with a single root, any topic can be a root by moving a particular topic into a map's center. The center topic is now the current subject of study, and the map is automatically redrawn to depict the other topics according to their new degree of indirection from the study topic. (Figure L)

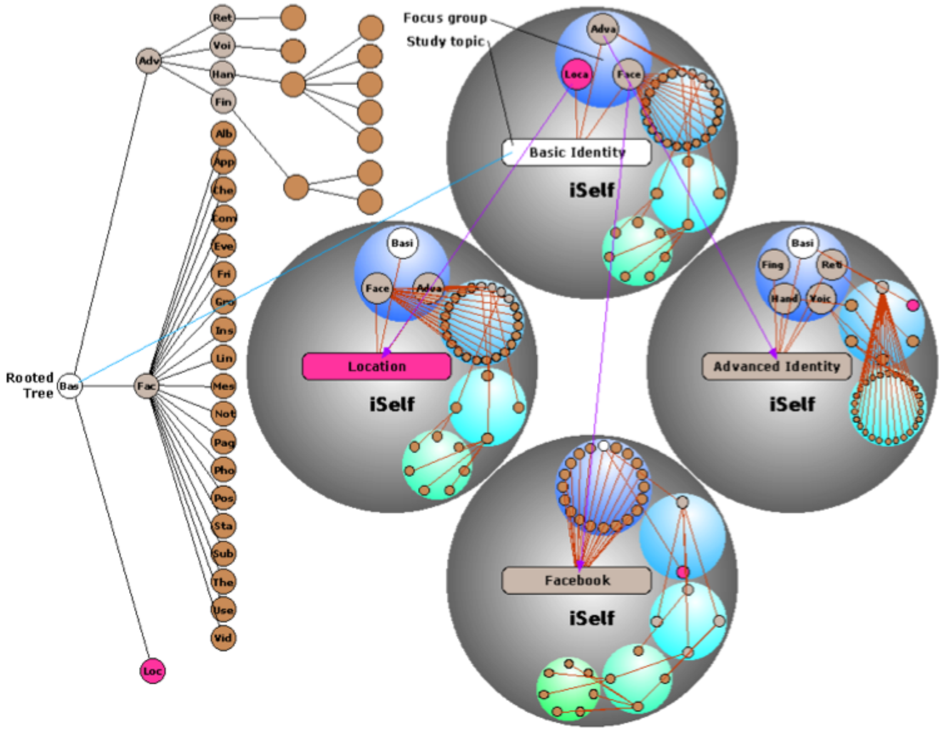


Figure L: Multiple Topic Map views

One special type of Topic Map is called an **iSelf**. An iSelf is similar to a 'data locker.' iSelfs contain a 'Basic Identity' topic, consisting of the user's dossier attributes. Additional topics can be added to the map that represents a user's biometrics, social interpretation, healthcare records, and ultimately DNA. Each iSelf is expressed in a graphical layout that is consistent in all Topic Maps. iSelfs allow a user to easily declare which personal data they choose to make available to others, overcoming the challenge of deciphering a myriad of privacy setting configurations offered at most social media sites.

9) Mobile Agents: *Perform tasks on behalf of users*

Social media relies on processing personal data that is currently exploding in size and complexity, exponentially increasing information overload. Mobile agents can manage information overload by performing useful tasks on the user's behalf by traveling across the Topic Map substrate. This machine behavior is a necessary to enable the fWeb to boost user productivity, a key attribute for creating sufficient benefit to spark viral adoption of the Personal Data Ecosystem.

Topic, **association**, and **occurrence** are three of the six **forms of concept language elements** used in a Topic Map. (Figure M) An entity can control one or more Topic Maps that may be interconnected with Topic Maps controlled by other entities.

All concepts are sourced from genealogies to provide the context that will allow mobile agents to perform inference across the fWeb. Mobile agents travel in and across Topic Maps, to determine topic meaning, processing universal data, and return occurrences of resources back to the originating system. The originating system can use an occurrence (link) to access and copy a remote resource for processing.

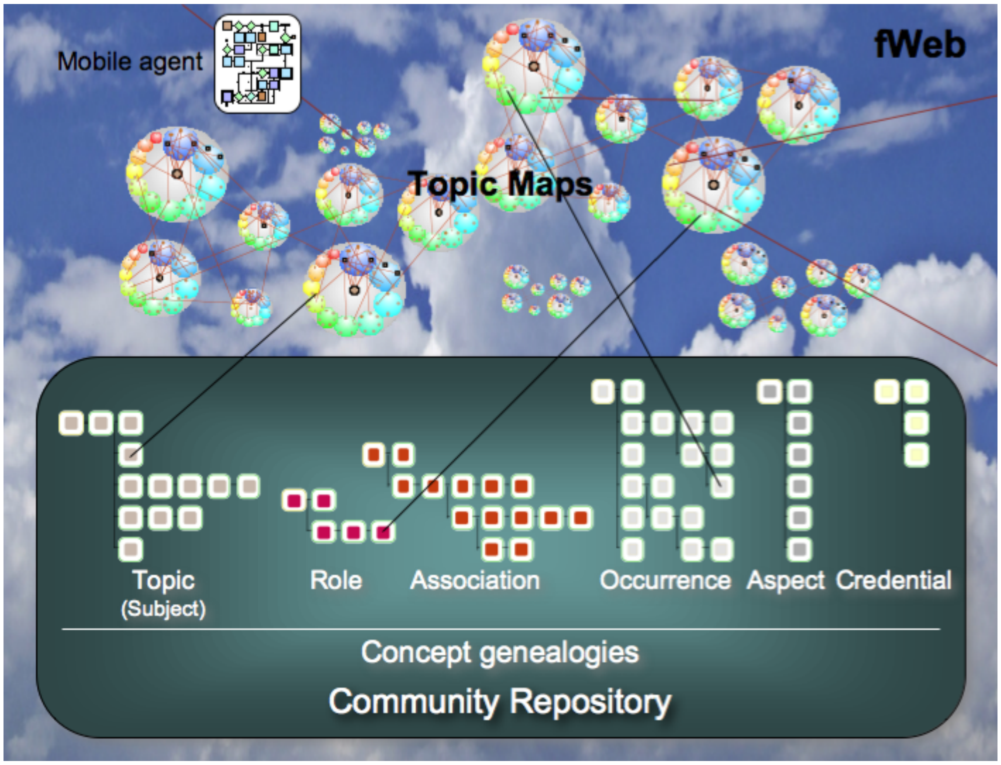


Figure M: Repositories provide context for map concepts

10) Security Model: *Supports trust, behavior, and access safeguards*

Trust – To verify the value of an attribute, Cubicon uses **decentralized trust**. This approach allows an entity to verify any attribute's value using relationship-based authentication (see Innovation 5).

Behavior – The use of 'keyed' software will prohibit pathogenic behaviors from proliferating across devices running a Cubicon VM. (Figure N) The Community Repository keys instructions that will only execute in a pre-authorized device.

Access – Without first receiving credentials, an identity provider's server has no practical way to directly access an iSelf. This barrier provides an additional level of protection on top of individually encrypted topics.

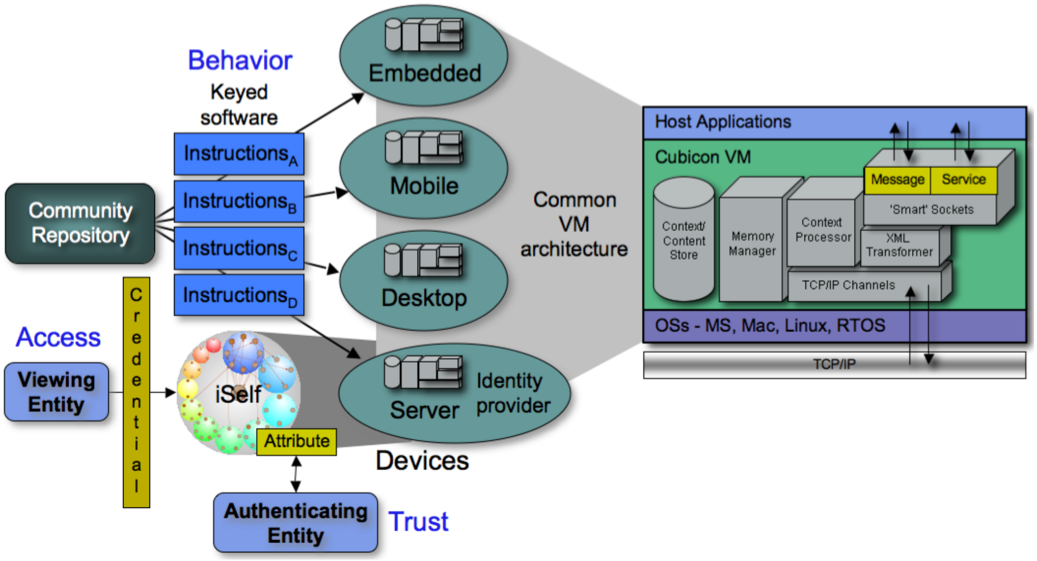


Figure N: Trust, behavior and access safeguard

11) Open Design: *A viable software market model beyond Open Source*

Open Design is a new software market model that supports an 'eBay for IP' called the **Intellectual Property (IP) Exchange**. The IP Exchange puts the *user* in the driver seat: allowing **direct compensation** for their personal data access provided to advertisers and other interested third parties.

Sources of additional revenue include:

- Yearly fees for fWeb registration (similar to domain registration)
- Fees for attribute certification
- Fees for third party application/app certification (limited software warranty)
- Metering of access to Topic Maps and user-generated content on the fWeb
- Metering of mobile agent usage
- Micropayment fee for services, transactions, and queries exchanged over the fWeb
- Fee for IP tracking to enforce copyright on software components

The IP Exchange will support virtual supply chains; distributing royalties back to the developers of software components nested in applications and apps. (Figure O)

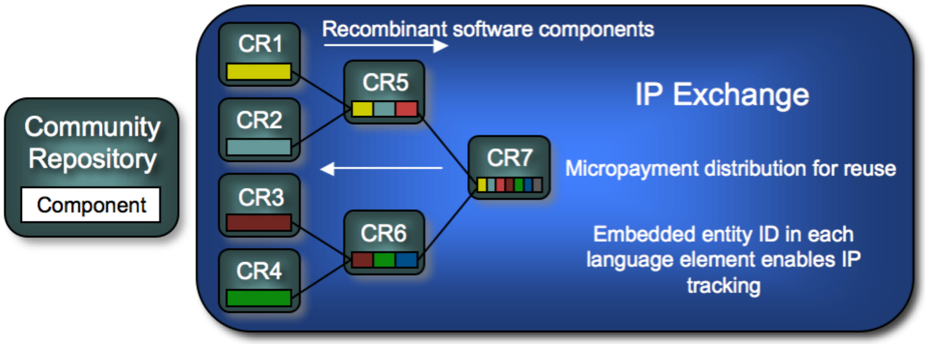

Community Repository
Component
Recombinant software components
CR1
CR2
CR3
CR4
CR5
CR6
CR7
IP Exchange
Micropayment distribution for reuse
Embedded entity ID in each language element enables IP tracking

Figure O: Virtual supply chain

12) Detecting Privacy Policy Infractions: *Monitors compliance*

The harmonization of topics and their attributes will result in the emergence of standard iSeals for many commercial and non-commercial services. An iSelf manages how a user shares their dossier and service data with specific relying/third parties.

Each iSeal contains attribute-sharing permission status to relying/third parties, which may be withdrawn by the user in the future. The FTC can monitor permissioned data activities in order to measure tracking infractions against new privacy laws similar to the Fair Credit Reporting Act.

The Science of Cubicon

Cubicon is the result of over two decades of applied research and development. It is a new branch of computer science that is described in graphical blueprints and animations. Blueprints are complete for the entire Cubicon Architecture, articulated in over 4,000 **core components** using icon-based syntax. More than 12,000 frames of animation precisely detail every interaction scenario the user can perform. The Memory Manager module has already been translated from its graphical blueprint into 'C,' and is shown to significantly outperform contemporary VMs. The garbage collection event has been benchmarked at less than one microsecond. Fully specified and diagrammatically interconnected, the architecture can be encoded without the communications overhead typically associated with a large programming project.

Cubicon: Infrastructure Development Strategy

The Cubicon architecture will be developed into the **Cubicon Trusted Infrastructure for the Ambient Cloud** consisting of four elements. (Figure P)

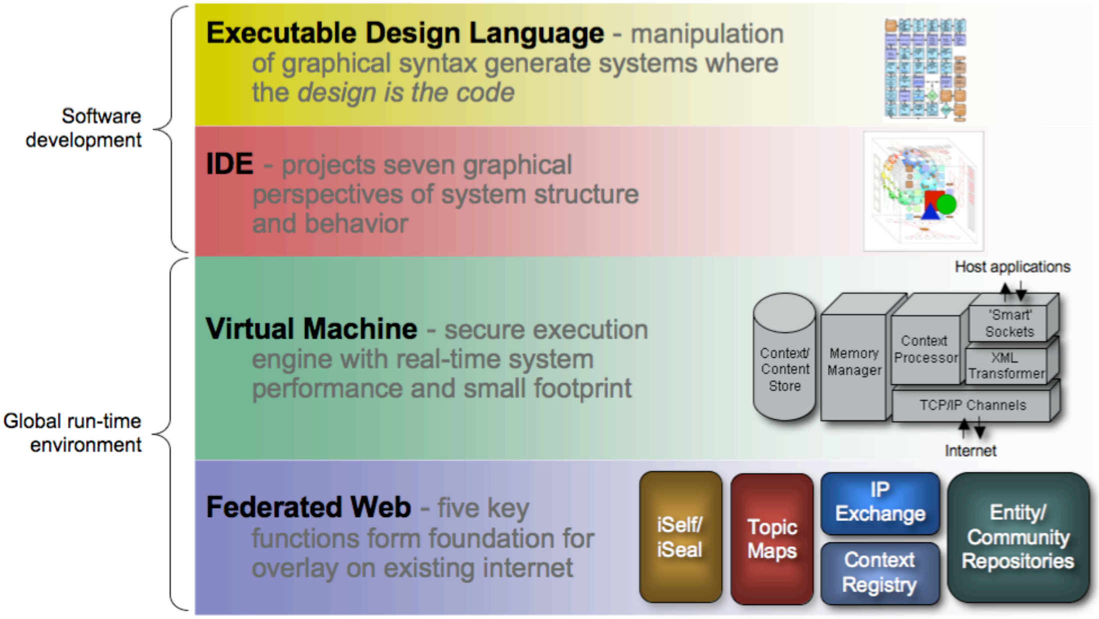

Software development
Global run-time environment
Executable Design Language - manipulation of graphical syntax generate systems where the *design is the code*
IDE - projects seven graphical perspectives of system structure and behavior
Virtual Machine - secure execution engine with real-time system performance and small footprint
Host applications
Context/ Content Store
Memory Manager
Context Processor
'Smart' Sockets
XML Transformer
TCP/IP Channels
Internet
Federated Web - five key functions form foundation for overlay on existing internet
iSelf/ iSeal
Topic Maps
IP Exchange
Context Registry
Entity/ Community Repositories

Figure P: Cubicon: Infrastructure

Development of the Cubicon Infrastructure can best be accomplished through a 'skunkworks' project supported by a single commercial company. This autonomy will ensure design continuity and accelerated deployment.

Moderate resources are required to complete two milestones within a year, consisting of:

1) Demonstrating trusted computing over TCP/IP
2) Benchmarking the production line efficiency to optimize core component Gantt/Pert scheduling

Sufficient core component assemblies will be completed within 1.5 years to support pilot projects that lead to field trials. First NSTIC participant engagement is with the commercial anchor identity provider to demonstrate automation of identity infrastructure core functions. Engagement with first relying parties will commence in 2.0 years with a software development kit (SDK) for the iSelf/iSeal tool. (Figure Q)

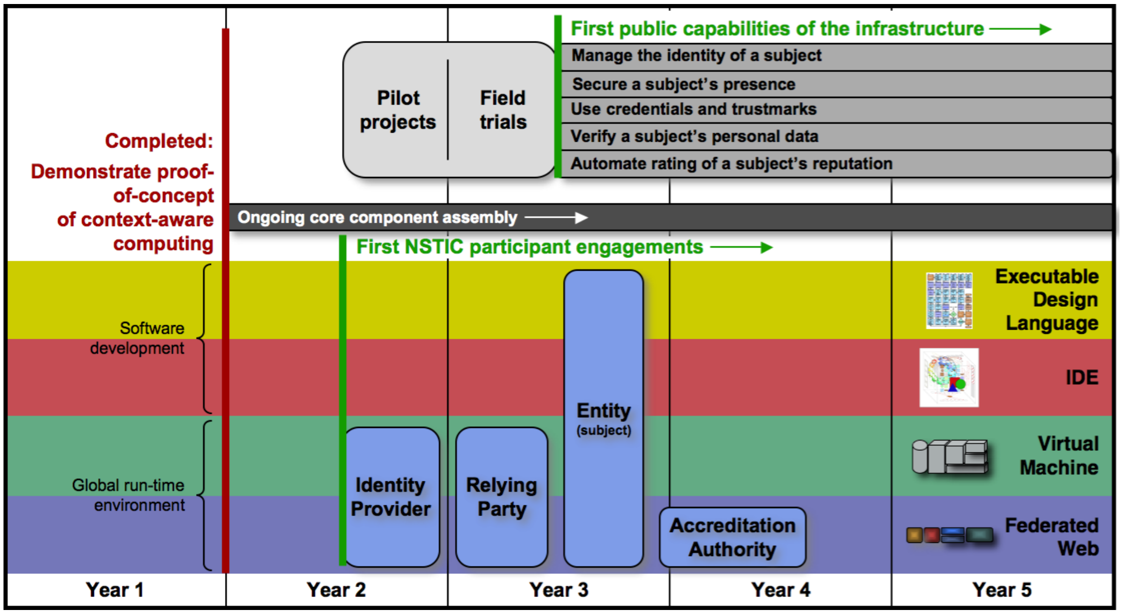

First public capabilities of the infrastructure
Manage the identity of a subject
Secure a subject's presence
Use credentials and trustmarks
Verify a subject's personal data
Automate rating of a subject's reputation
Pilot projects
Field trials
Completed:
Demonstrate proof-of-concept of context-aware computing
Ongoing core component assembly
First NSTIC participant engagements
Software development
Global run-time environment
Identity Provider
Relying Party
Entity (subject)
Accreditation Authority
Executable Design Language
IDE
Virtual Machine
Federated Web
Year 1
Year 2
Year 3
Year 4
Year 5

Figure Q: Cubicon: Development Strategy

Cubicon: Optimal Adoption Strategy

Cubicon can provide an ideal technological base to ensure a more secure and private cyberspace. We believe NSTIC is the ideal federal initiative to sponsor Cubicon. In Figure R, we outline a strategy to adopt Cubicon based on federal, corporate, and user involvement.

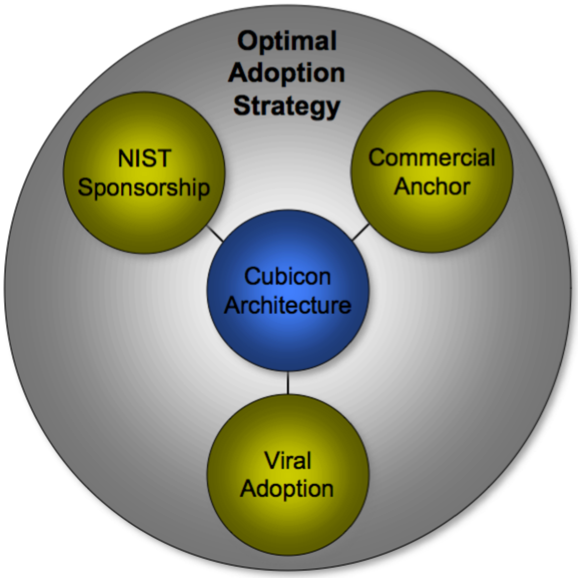


Figure R: Optimal Adoption Strategy

NIST Sponsorship

- Cubicon automation can be used to govern each participant's behavior in both ecosystems.
- The Cubicon architecture provides an alternative to UnitsML by automating SI (System of Units) with the direct binding of units to numeric-typed values, ideal for use in general computing practices.
- Cubicon's community-based development process can be used to harmonize Smart Grid standards, supporting interoperability between devices in the ambient environment and the emerging 'Internet of Things.'

Commercial Anchor

- Early access to the architecture's IP enables a company to lead the build-out of the Cubicon Infrastructure from a competitive position.
- Venture participation can result in substantial capital gains based on taking an equity position in the CoreTalk venture.
- Internal adoption of the **Cubicon software development life-cycle process** can lead to increased employee productivity and shorter time to market across product lines and service offerings.

Viral Adoption

The collective awareness of rising identity and asset theft is creating market demand for new security and privacy technologies. The Cubicon architecture includes the iSelf/iSeal tool to be widely deployed through mobile devices and desktop Web browsers. (Figure S) The iSelf manages how a user shares their dossier data with others. The iSeal provides the option to record service transactions into their dossier. This tool will help address consumer privacy concerns, as well as train them on how to properly safeguard their personal data through a highly automated and intuitive interface.

The tool provides a Cubicon adoption path for users into increasing infrastructure capabilities specified in the blueprints and animations.

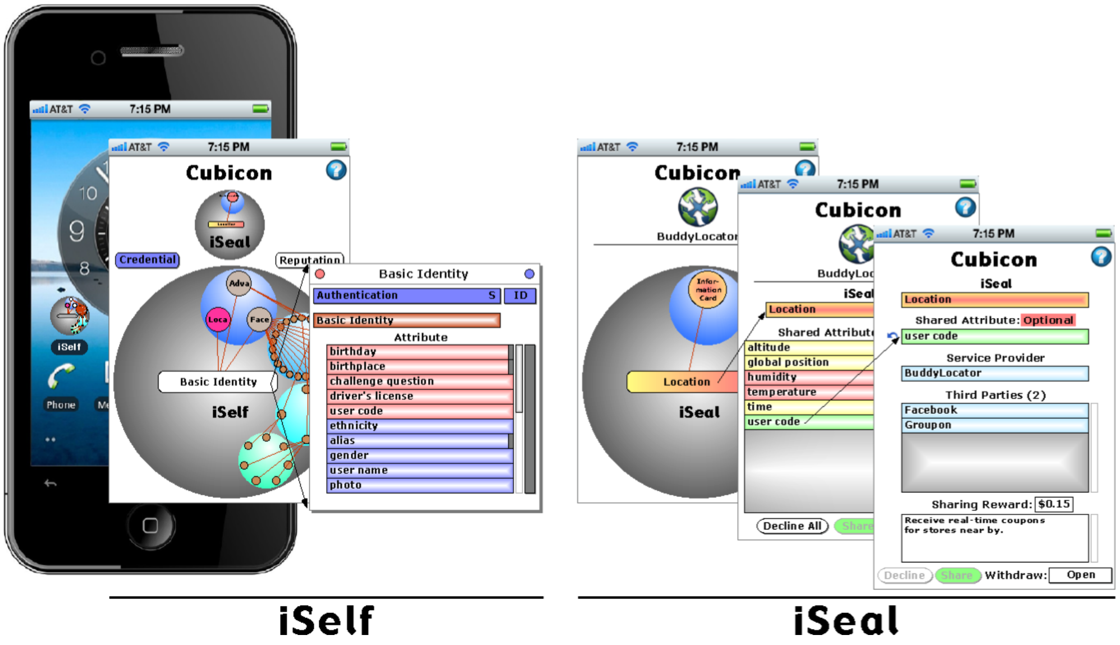


Figure S: iSelf/iSeal tool

Conclusion

The Cubicon architecture is the best holistic match for NSTIC Objective 4.1: Drive innovation through aggressive science and technology (S&T) and research and development (R&D) efforts that will transform vision into sustainable strategy.

Sanford B. Klausner
Founder & CEO
CoreTalk Corporation
klausner@coretalk.net
408.621.4709