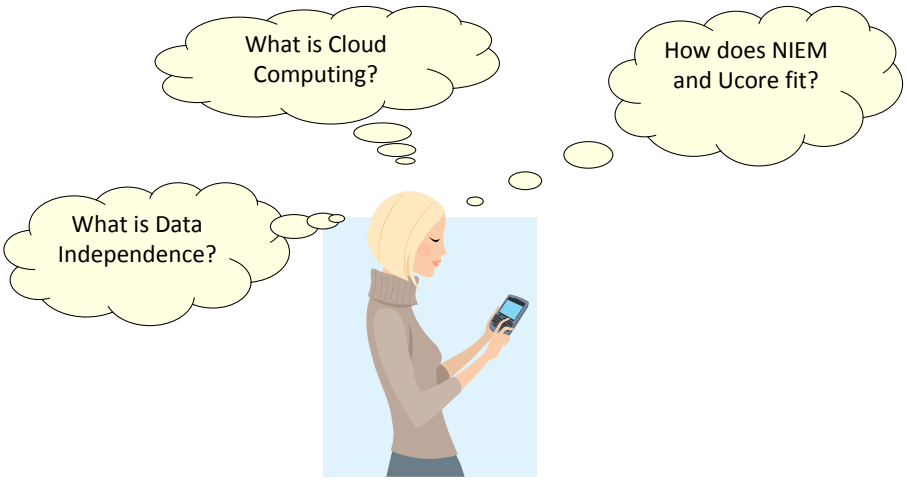
Future of NIEM

Cloud Computing and the Importance of Data Independence

Future of NIEM Vision



As the lead of a enterprise data management organization, there are two mantras we need to constantly talk about…

1) Software changes all of the time. Hardware changes even faster. Data, on the other hand, is the most lasting part of any IT architecture. Managing data at the enterprise level is important.

2) The cyber threat in not getting any smaller! Building security at the data level as it moves is more important than ever.

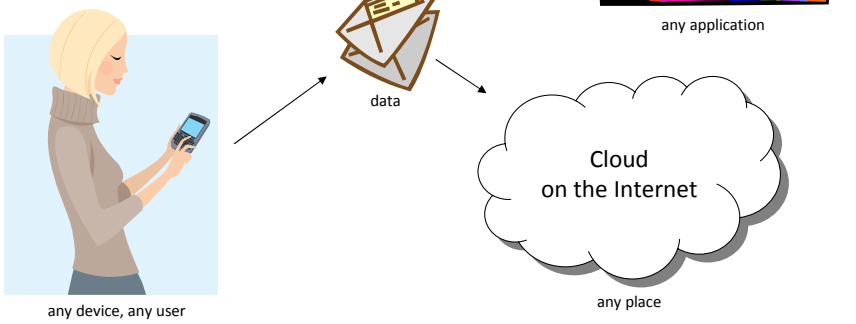
As the lead of a enterprise data management organization, NIEM is one of my biggest tools to help combat both…. NIEM is what makes information sharing work better, and what allows for cost avoidance / cost reduction in government IT budgets.

As the Executive Director of NIEM, the program must be able to rapidly adapt and evolve to changes in the computing environment in order to continue to act as the BRIDGE FOR INFORMATION SYSTEMS. More and more, looking into the future means addressing issues of secure and trusted Information sharing in the Cloud Computing environment.

So, I am going to talk today about Data Independence, how NIEM and UCore are part of the solution for achieving data independence: a principal that is becoming increasingly important in the shift towards cloud-computing. I am hoping that you walk away asking yourselves how you are addressing data independence and how NIEM/UCore fits in the picture for you.

What is Data Independence?

Data moving to any authorized place you need it, in a form that any authorized application can use it, by any authorized user, on any authorized device.



My current definition of data independence is:

> " Data moving to any authorized place you need it, in a form that any authorized application can use it, by any authorized user, on any authorized device."

Email is an example of something in our everyday lives that is gaining data independence. You can email from virtually any device, from multiple types of applications, securely to users. For a long time, companies like AOL did not support standards that would allow their mail to be sent or received by other applications.

Definition of data independence has evolved from the 1980s definition that produced the concept of a DBMS, where data structures were separate from data, and from the code that manipulated it. It changes for each evolution or revolution that happens in computing history. XML, Web 2.0 and Cloud computing are the most recent changes that have caused a rethinking of the definition again.

(see pages 14-16 for more on Data Independence)

What is Cloud Computing?

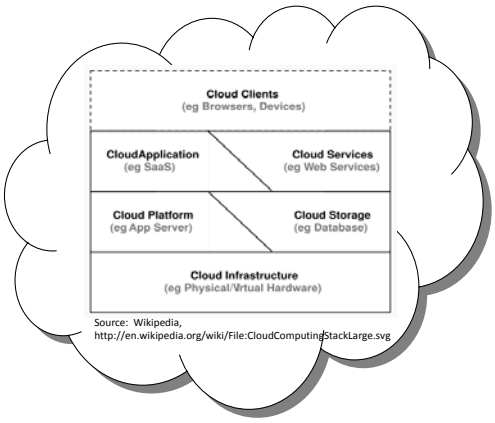
The *cloud* is a metaphor for the Internet, based on how it is depicted in computer network diagrams, and is an abstraction for the complex infrastructure it conceals.

Cloud Computing is about the delivery of computing resources from a location other than that from the user.

In its most used context it is Internet-based ("*cloud*") development and use of computer technology ("computing").

Most risks of cloud computing associated with "Data Security" and "Data Privacy".

"Who controls access to data I put into the cloud?"



Source: Wikipedia, http://en.wikipedia.org/wiki/File:CloudComputingStackLarge.svg

---

For me, Cloud Computing is about the delivery of computing resources from a location other than that from the user or device by leveraging various types of transparent architectures.

While the architecture of a cloud might be very different, clouds themselves are depicted as oversimplifications of an entire ecosystem of computing resources. Software as a Service (SaaS) is one component of cloud computing.

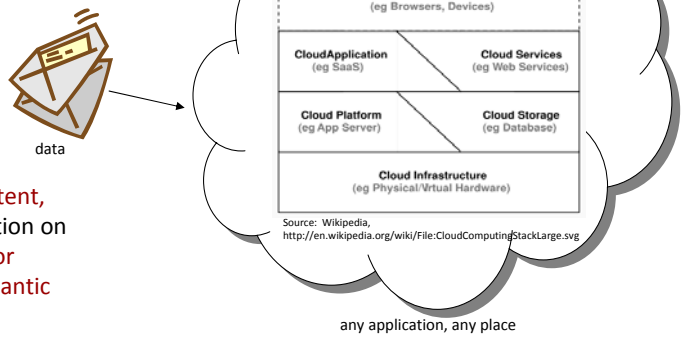
The architecture of computing has evolved from DBMSs of the 80's to well known Service Oriented and Event Driven Architectures (SOA and EDA) being implemented in many government computing environments. The success of the cloud is also a result of lesser known, but successful architectures of Space Based and Share Nothing architectures used and developed by powerful industry giants such as Google and Amazon.

(see pages 16-20 for more on SOA, EDA, SBA and SNA)

(see pages 22-23 for more on Cloud Computing)

Impact of Cloud Computing on Data Independence?

Data moving to any authorized place you need it, in a form that any authorized application can use it, by any authorized user, on any authorized device.



In addition to the content, must include information on the structure, access or security info, and semantic understanding.

"How do I [as an IT leader] continue to achieve data independence in today's computing environment?" Building on the idea about the RDBMS as mechanism to achieve data independence in the 80's and 90's computing environment, we are continually challenged by and at the mercy of vendors implementation of data storage beneath applications. As we move toward Cloud computer, these challenges are amplified.

Achieving Data Independence in Cloud Computing makes it necessary for data, as it moves between clouds, to include context in addition to content, meaning structure, access or security info, and common semantic understanding in order to ensure interoperability. With these attributes, the vision " data, in any authorized place you need it, in a form that any authorized application can use it, by any authorized user, on any authorized device" is possible.

Applications in the cloud need to understand data!!! Data Independence will not be achieved by all applications uniformly. The success drivers for data independence include:

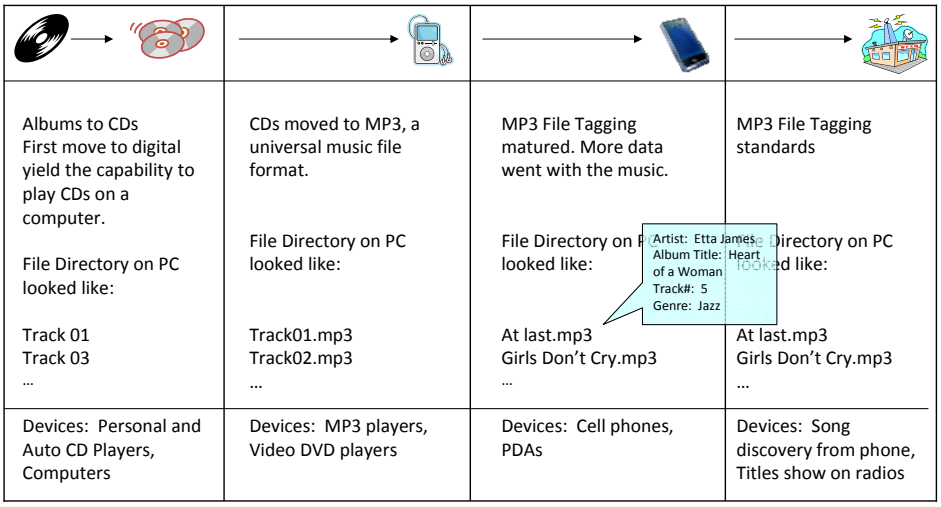
- Data is reliably delivered from distributed sources (data centers/servers) using virtualized technologies containing trusted, reliable data
- Services are secure and accessible with a single or logical point of access
- Market driven services need to meet the quality of service requirements of customers
- Open standards are critical (NIEM, UCORE, GML, SAML, XACML,…) with shared or common semantics

Imagine clicking on content you find through internet search, and since the data has security info embedded with it, the application (browser) knows when to ask for additional information before showing more it to the user. This already happens today, with secure Music files and protected images.

Data independence is the best path to mitigating the growing risk of cyber security. Architecting for data independence preserves your investment in data.

Digital Music: Data Independence Success?

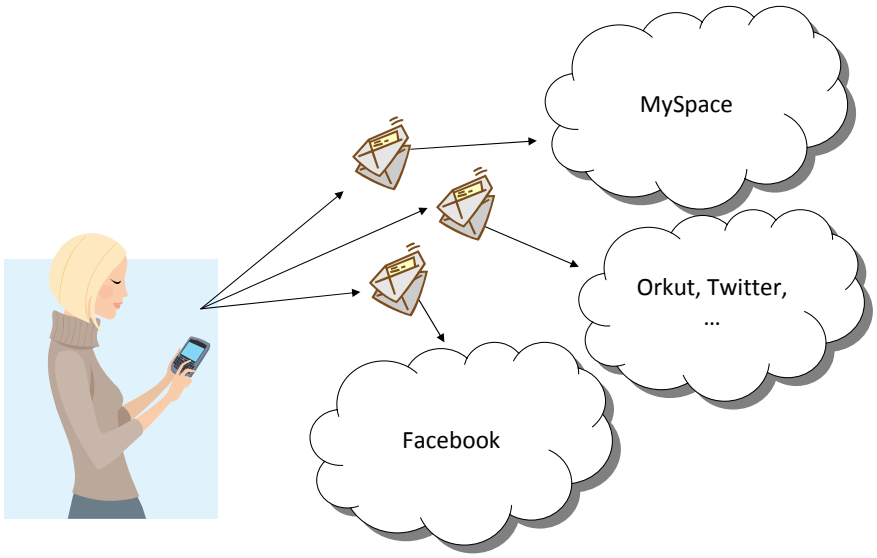
Digital Music



Moving from albums to CDs started the data independence evolution for music. Along the way, additional elements were added to the data that allows today's version of data independence, where you can hold a phone to a speaker, and the application on the phone recognizes the song, downloads the song identification information back to the phone, and allows the user to order and download a song that they had no information about. Music files move well! Any music, any place, any device, any user, as authorized.

Analog to digital All data about songs printed on album cover now printed on CD cover and label. ---Still Visual	MP3 as a file format was very portable. Data about the song (or file) was stored in database or software outside of MP3 file	MP3 tagging more standardized, down to a few definitions (ID2, ID3 RealID). Data about the song is embedded in the file. Viewable in windows file viewer	MP3 tagging mostly standardized. Devices can listen to songs, and look up data from central service, and store info for user to download, purchase and store.
Can copy CD as is, but CD without visual marking requiring playing and human recognition to determine the content	Ripping required re-keying of data about the song. Painful, but easy to overcome Retagging across software required since data was not kept with music. (example Real Player, Windows Media Player….)	•Minimal if none retagging required to move files across media players. •Security at file level, tagged to a device •Applications uses common services to retrieve tags during "ripping process"	Data Independence? YES!

Social Networking: Data Independence Success?



Let's look at cloud computing application such as social networking. There are lots of different types of information exchanges that happen between application and user on numerous types of devices. It is a complex, and continually evolving environment with constantly changing requirements as applications are used more and more. Right now, there is lots of re-keying to stay current in multiple applications.

There are some types of information exchange within Social Networking sites that are relatively consistent - such as reviews, ratings, and feedback – which currently deliver high value (AMAZON started this).

Social sites are evolving from silo applications to social platforms, aggregators are developing, and context brokers are now emerging. What is happening today parallels the financial service companies trying to aggregate visibility to people's financial situation. Inter-Cloud exchange standards are under development for social networking, but Data Independence remains relatively immature due to lack of open based standards, lack of driving force for interoperability, and lack of discretely defined exchanges.

Data generally not secure!!!!! NOT yet achieving full success. Social Networking sites therefore, are not useable in production environments where security is critical.

Success Factors For Data Independence

- Market demand pushed
- Distinct exchanges for discrete purposes
- Open standards-based
- Security and Privacy
- Applications and services (cloud or not) need to be built to accept the distinct, open standards-based exchange

---

For social networking, vendors are still building SILOS. Myspace has largest user base, with Facebook seemingly at second, and large number of smaller, niche sites trailing behind. The market may not be mature enough to demand interoperability as each social networking site has their core set of users, and it is relatively unknown the size of the subset of multiple.

Social networking, for example, still a relatively moving target of discretely defined info exchanges.

Open, standards based allows for broader vendor adoption!!!! Look what happened with MP3.

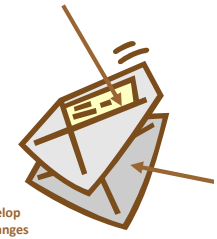
Vendors need to see value in adopting.

Data Independence means data moves with it's context (or metadata).

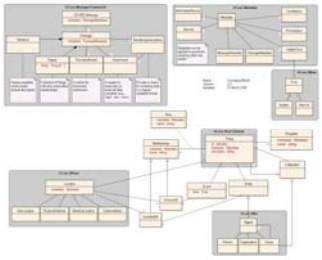
What are NIEM / Ucore / LEXS?

NIEM: information exchange model that is used to structure of the payload of the information exchange, using common - shared semantics.

Lifecycle Approach to develop discrete information exchanges



Ucore/ LEXS: information exchange specification and implementation profile with shared semantics for a small set of elements (who, what, when & where) with security markings to permit access control, providing a messaging framework.



Both are open standards built by practitioners for information exchange of distinct packages of information

---

What is NIEM:

NIEM provides two core products:

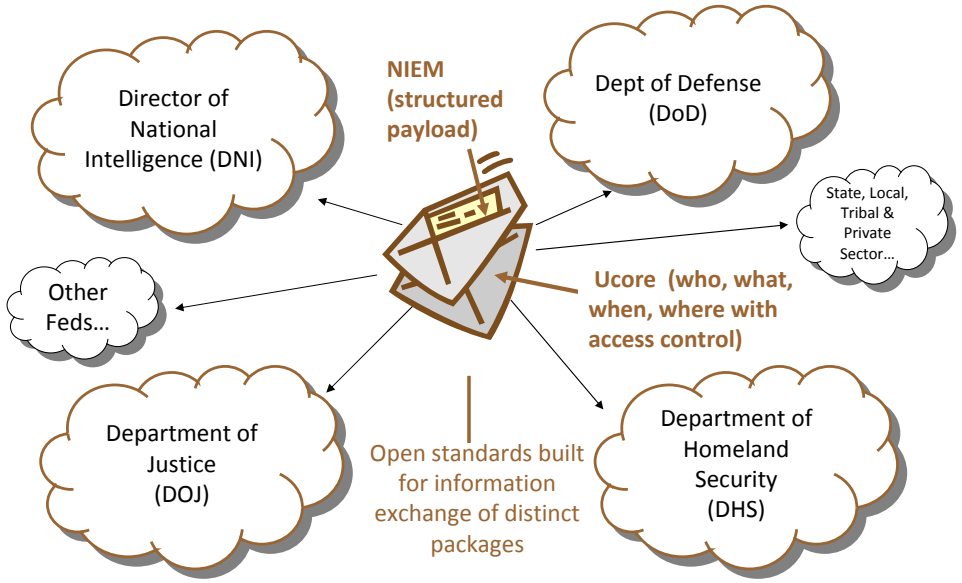
1) Data dictionary of commonly agreed upon elements, built for information exchange, governed by communities of interest.
2) A life cycle approach to develop discrete information exchanges, focused on reusability, used by practitioners at the federal, state, local, tribal and private sector agencies.

What is Ucore?

Developed and governed by DOD/DNI/DOJ and DHS, it is an information exchange specification and implementation profile with

- Vocabulary of most commonly exchanged concepts (Who, What, When, Where)
- XML representation of the concepts
- Extension rules to allow tailoring to specific mission areas
- Security markings to permit controlled access, electronic tear lines
- Messaging framework to package and un-package the content consistently

How does NIEM / Ucore fit?



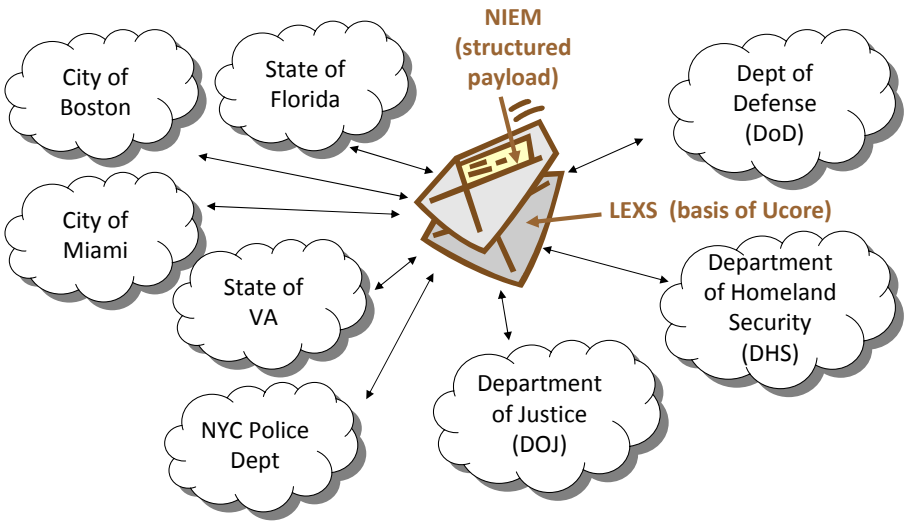
Ucore and NIEM provide part of the success factors to start achieving data independence in government computing.

Ucore provides the common message format, with information such as security, and structure of the content to allow for applications in the cloud to distinguish which messages or data packages they need to open. A basic semantic understanding of what is in the envelope about "who, what, when and where" is included in Ucore. NIEM can be used to provide structure for the payload or the contents within the envelope allows for authorized applications to "Do Something Relevant" with the data since there is additional detailed semantic understanding included.

Ucore and NIEM together, need to continue to evolve as open standards to allow for data independence thus allowing or advancing inter-cloud computing for secure and safe cross-government/private sector information exchange.

A significant missing link is a common access control schema that can describe how the data should be protected, so that applications can be built to open, check for authorization, and protect data between and across the clouds. The solution for common access control needs to be as repeatable, common and extensible as the IP schema that makes the internet work.

SAR: Data Independence Success?



Market demand pushed:

- For DOD/DHS/DOJ/DNI, this is the War on Terror.
- For Federal Health IT, this the reduction in costs associated with maintaining health electronic records.

Distinct exchanges for discrete purposes: Not all information to all users at once. Must be for a distinct purpose by which access can be decided and attributed to data. Examples: SAR, TWPDES, AIS, ANOA, SILO. The SAR includes attribution of privacy related data so that it can be protected.

SAR is Open standards-based: LEXS 3.1 and NIEM 2.x are mature, and capable. They are governed to ensure semantic understanding by community. LEXS carries in the message structure common elements similar to Ucore, specific for law enforcement and justice information. Ucore was architected from LEXS specifically for DOJ/DHS/DOD/DNI information sharing.

Applications (cloud or not) need to be built to the distinct, open standards based exchanges: Vendor adoption in COTS, GOTS and other applications is ultimate key to success. Examples: OneDOJ/NDEX, DHS LEISS, LEO, RMS vendors exchanging information in SAR format.

Gaps in the Road to Data Independence using NIEM /UCore / LEXS

- Lack of depth in common data access schemas addressing privacy and security
- Lack of documented distinct exchanges for discrete purposes
- Open standards lack capability for element level security marking
- Vendor adoption of Ucore / NIEM needs improvement

Lack of common data access schemas to allow for data to fly between clouds and still be protected. But, it can be accomplished within a community, or for a specific type of information to protected. Data architects and security architects need to work together to extend the concept of security architecture by "data type" to produce element level data security (or security in depth).

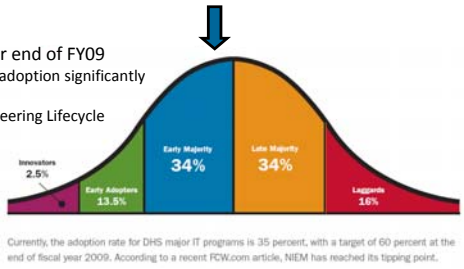
Focus on development of high value, cross government definitions of distinct exchanges for discrete purposes: SAR is a great example of work that needs to be completed for additional exchanges. Where are the high value exchanges in Health IT? Homeland Defense? Intelligence?

Ucore and NIEM both lack capability for element level security markings. Solution needs to stands up to transactional volume required by cloud computing. Is XML the answer?

Vendor adoption of Ucore / NIEM needs improvement. Some vendor adoption is happening now, but we need to reach out and develop stronger relationships in the vendor community

Current Success with NIEM / Ucore / LEXS

- DHS adoption at 35% of Major IT programs, 60% for end of FY09
  - FY08 focus on discrete Info Exchanges has increased adoption significantly
  - Measured compliance in investment review process
  - Built into Enterprise SOA Guidance and System Engineering Lifecycle
- DOJ adoption
  - CJIS System of Systems
  - Terrorism Screening Center (TSC)
  - FBI's Guardian
  - NDEX / One DOJ
  - Future Development of NGI



Currently, the adoption rate for DHS major IT programs is 35 percent, with a target of 60 percent at the end of fiscal year 2009. According to a recent FCW.com article, NIEM has reached its tipping point.

- DOD/DNI adoption through Maritime Domain Awareness and Ucore 2.0
- PM ISE adoption of the TSC Terrorist Watch Listing and Suspicious Activity Reporting Functional Standards

- 39 of 50 States using NIEM
  - 39 states represent approx 75% of U.S. population
  - Mostly in Justice-oriented applications, some state wide(TX, NY, FL)
  - Bolstered by DOJ/DHS Grants like Real ID implementation

The content within NIEM, the Domains are expanding / getting refined based on the current level of adoption to include most recent work in the following areas: Emergency Management, Geospatial, Immigration, Infrastructure Protection, International Trade, Rad-Nuc (CBRN), Screening and most recently Maritime

Indications of reaching the Tipping point:

- The use of NIEM-ify and NIEM-ifications are common across DHS
- Web searches return Job Descriptions for NIEM developers
- More requests about joining governance than outgoing requests to join
- Tools are being developed, on spec
- Integrators are training themselves

How can you help? Ask yourselves… what does NIEM and Ucore mean to me as an integrator? As a product developer? As a government users?

The NIEM program office is hosting a Vendor Day in combination with a Request for Information to support the growth of NIEM on Feb 17th. Additional information will be out soon on the details for the event.

Background on Data Independence

Principles of Data Independence

data independence
Circa 1970's

The separation of data from the programs that use the data. Nearly all modern applications are based on the principle of data independence. In fact, the whole concept of a database management system (DBMS) supports the notion of data independence since it represents a system for managing data separately from the programs that use the data. In contrast, it is possible to write applications in which the data being processed is actually represented in the program's source code. This *data-dependent* approach is very inflexible because it makes it difficult to modify the data and it also makes the data inaccessible to other programs.

Source: http://www.webopedia.com/TERM/D/data_independence.html

Principles of Data Independence

data independence for data
Circa 1990's

Techniques that allow data to be changed without affecting the applications that process it. There are two kinds of data independence. The first type is data independence for data, which is accomplished in a database management system (DBMS). It allows the database to be structurally changed without affecting most existing programs. Programs access data in a DBMS by field and are concerned with only the data fields they use, not the format of the complete record. Thus, when the record layout is updated (fields added, deleted or changed in size), the only programs that must be changed are those that use those new fields.

data independence for processing

The second type of data independence relates to processing and refers to miscellaneous data used in programs that might change in the future, such as discount rates, product descriptions and error messages. Such data should be stored in a database and not "hard wired" into the code of the program. When values change, only the database item is altered, a much simpler task than recompiling numerous programs.

Source: Computer Desktop Encyclopedia, http://www.answers.com/topic/data-independence

Principles of Data Independence

data independence: first level
Circa EARLY 2000's

The logical structure of the data is known as the **schema definition**. In general, if a user application operates on a subset of the attributes of a relation, it should not be affected later when new attributes are added to the same relation. Logical data independence indicates that the conceptual schema can be changed without affecting the existing schemas..

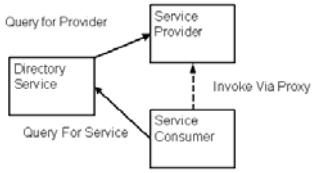
data independence: second level

The physical structure of the data is referred to as **physical data description**. Physical data independence deals with hiding the details of the storage structure from user applications. The application should not be involved with these issues since, conceptually, there is no difference in the operations carried out against the data.

Source: Wikipedia, http://en.wikipedia.org/wiki/Data_independence

Principles of SOA

Service Oriented Architecture
Circa 2003



SOA achieves interoperability between different systems and programming languages provides the basis for integration between applications on different platforms through a communication protocol. One example of such communication is based on the concept of messages. Using messages across defined message channels decreases the complexity of the end application thereby allowing the developer of the application to focus on true application functionality instead of the intricate needs of a communication protocol.

Allows new functionality developed to reference a common business format for each data element.

Source: Wikipedia, http://en.wikipedia.org/wiki/Service-Oriented_Architecture#Principles

Principles of EDA

Event Driven Architecture
Circa 2003

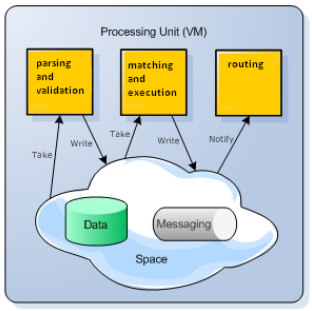
EDA is a software architecture pattern promoting the production, detection, consumption of, and reaction to events.

An event driven architecture is extremely loose coupled and well distributed. The great distribution of this architecture exists because an event can be almost anything and exist almost anywhere. The architecture is extremely loose coupled because the event itself doesn't know about the consequences of its cause. e.g. If we have an alarm system that records information when the front door opens, the door itself doesn't know that the alarm system will add information when the door opens, just that the door has been opened

Source: Wikipedia, http://en.wikipedia.org/wiki/Event_Driven_Architecture

Principles of SBA

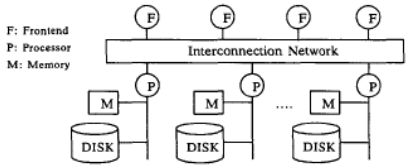
Space Based Architecture
Circa 2002



SBA is a software architecture pattern for achieving linear scalability of stateful, high-performance applications, following many of the principles of Representational State Transfer, Service-Oriented Architecture and Event-Driven Architecture, as well as elements of grid computing. With a space-based architecture, applications are built out of a set of self-sufficient units, known as processing-units (PU). These units are independent of each other, so that the application can scale by adding more units.

Source: Wikipedia, http://en.wikipedia.org/wiki/Space-based_architecture

Principles of SN



Shared Nothing Architecture

Circa 1986

SN is a distributed computing architecture in which each node is independent and self-sufficient, and there is no single point of contention across the system. People typically contrast SN with systems that keep a large amount of centrally-stored state information, whether in a database, an application server, or any other similar single point of contention. While SN is best known in the context of web development, the concept predates the web.

Shared Nothing is popular for web development because of its scalability. As Google has demonstrated, a pure SN system can scale almost infinitely simply by adding nodes in the form of inexpensive computers, since there is no single bottleneck to slow the system down. Google calls this *sharding*. An SN system typically partitions its data among many nodes on different databases (assigning different computers to deal with different users or queries), or may require every node to maintain its own copy of the application's data, using some kind of coordination protocol.

Source: Wikipedia, http://en.wikipedia.org/wiki/Shared_nothing_architecture

Background on Cloud Computing

Principles of Cloud Computing

Cloud computing

Circa 2000's

Cloud computing refers to the delivery of computational resources from a location other than that from the user. In its most used context it is Internet-based (*"cloud"*) development and use of computer technology ("computing"). The *cloud* is a metaphor for the Internet, based on how it is depicted in computer network diagrams, and is an abstraction for the complex infrastructure it conceals.

It is a style of computing in which IT-related capabilities are provided "as a service", allowing users to access technology-enabled services from the Internet ("in the cloud") without knowledge of, expertise with, or control over the technology infrastructure that supports them. According to a 2008 paper published by *IEEE Internet Computing* "Cloud Computing is a paradigm in which information is permanently stored in servers on the Internet and cached temporarily on clients that include desktops, entertainment centers, tablet computers, notebooks, wall computers, handhelds, sensors, monitors, etc."

Cloud computing is a general concept that incorporates software as a service (SaaS), Web 2.0 and other recent, well-known technology trends, in which the common theme is reliance on the Internet for satisfying the computing needs of the users. For example, Google Apps provides common business applications online that are accessed from a web browser, while the software and data are stored on the servers.

Source: Wikipedia, http://en.wikipedia.org/wiki/Cloud_computing

Architecture of Cloud Computing

Cloud computing: Architecture
Circa 2000's

The majority of cloud computing infrastructure currently consists of reliable services delivered through data centers that are built on servers with different levels of virtualization technologies.

The services are accessible anywhere in the world, with *The Cloud* appearing as a single point of access for all the computing needs of consumers.

[Commerical] Offerings need to meet the quality of service requirements of customers and typically offer service level agreements.

Open standards and open source software are also critical to the growth of cloud computing.

Source: Wikipedia, http://en.wikipedia.org/wiki/Cloud_computing