Interoperability in Systems Engineering

Lahoma Sells

First Edition, 2012

ISBN 978-81-323-3075-2

Published by:
Research World
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Chapter 1

Interoperability

Interoperability is a property referring to the ability of diverse systems and organizations to work together (inter-operate). The term is often used in a technical systems engineering sense, or alternatively in a broad sense, taking into account social, political, and organizational factors that impact system to system performance.

Definition

While interoperability was initially defined for IT systems or services and only allows for information to be exchanged, a more generic definition could be this one :

Interoperability is a property of a product or system, whose interfaces are completely understood, to work with other products or systems, present or future, without any restricted access or implementation.

This generalized definition can then be used on any system, not only information technology system. It defines several criteria that can be used to discriminate between systems that are "really" inter-operable and systems that are sold as such but are not because they don't respect one of the aforementioned criteria, namely :

- non-disclosure of one or several interfaces
- implementation or access restriction built in the product/system/service

The IEEE Glossary defines interoperability as:

the ability of two or more systems or components to exchange information and to use the information that has been exchanged.

James A. O'Brien and George M. Marakas define interoperability as:

Being able to accomplish end-user applications using different types of computer systems, operating systems, and application software, interconnected by different types of local and wide area networks.

Syntactic interoperability

If two or more systems are capable of communicating and exchanging data, they are exhibiting syntactic interoperability. Specified data formats, communication protocols and the like are fundamental. XML or SQL standards are among the tools of syntactic interoperability. This is also true for lower-level data formats, such as ensuring alphabetical characters are stored in ASCII format in all the communicating systems.

Syntactical interoperability is a necessary condition for further interoperability.

Semantic interoperability

Beyond the ability of two or more computer systems to exchange information, semantic interoperability is the ability to automatically interpret the information exchanged meaningfully and accurately in order to produce useful results as defined by the end users of both systems. To achieve semantic interoperability, both sides must refer to a common information exchange reference model. The content of the information exchange requests are unambiguously defined: what is sent is the same as what is understood.

Interoperability and Open Standards

Interoperability must be distinguished from Open Standards. Although the goal of each is to provide effective and efficient exchange between computer systems, the mechanism for accomplishing that goal is very different. Open Standards imply interoperability *ab-initio,* i.e., by definition, while interoperability does not, by itself, imply wider exchange between a range of products, or similar products from several different vendors, or even past future revisions of the same product. Interoperability may be developed *post-facto,* as a special measure between two products, while excluding the rest, or even when the vendors of the two 'interoperable' products are forced into a dominant/submissive or exploitative relationship vis-a-vis themselves or their customers.

Open Standards

Open Standards rely on a broadly consultative and inclusive group including representatives from vendors, academicians and others holding a stake in the development. That discusses and debates the technical and economic merits, demerits and feasibility of a proposed common protocol. After the doubts and reservations of all members are addressed, the resulting common document is endorsed as a *common standard.* This document is subsequently released to the public, and henceforth becomes an *open standard.* It is usually published and is available freely or at a nominal cost to any and all comers, with *no further encumbrances.* Various vendors and individuals (even those who were not part of the original group) can use the standards document to

make products that implement the common protocol defined in the standard, and are thus **interoperable by design,** with no specific liability or advantage for any customer for choosing one product over another on the basis of standardised features. The vendors' products compete on the quality of their implementation, user interface, ease of use, performance, price, and a host of other factors, while keeping the customers data intact and transferable even if he chooses to switch to another competing product for business reasons.

Post Facto Interoperability

Post-facto interoperability may be the result of the absolute market dominance of a particular product in contravention of any applicable standards, or if any effective standards were not present at the time of that product's introduction. The vendor behind that product can then choose to *ignore* any forthcoming standards and not co-operate in any standardisation process at all, using its near-monopoly to insist that its product sets the *de-facto* standard by its very market dominance. This is not a problem if the product's implementation is open *and* minimally encumbered. However, this is not often the case and the product is both closed and heavily encumbered (e.g. by patent claims). Achieving interoperability with such a product is both critical for any other vendor if it wishes to remain relevant in the market, and extremely difficult to accomplish because of lack of co-operation on equal terms with the original vendor, who may well see the new vendor as a potential competitor and threat. The newer implementations often rely on clean-room reverse engineering in the absence of technical data to achieve interoperability. The original vendors can provide such technical data to others, often in the name of 'encouraging competition,' but such data are invariably encumbered, and may be of limited use. Availability of such data is *not* equivalent to an open standard, because:

1. The data are provided by the original vendor on a discretionary basis, who has every interest in blocking the effective implementation of competing solutions, and may subtly alter or change its product, often in newer revisions, so that competitors' implementations are almost, but not quite completely interoperable, leading customers to consider them unreliable or of a lower quality. These changes can either not be passed on to other vendors at all, or passed on after a strategic delay, maintaining the market dominance of the original vendor.
2. The data themselves may be encumbered, e.g. by patents or pricing, leading to a dependence of all competing solutions on the original vendor, and possibly leading a revenue stream from the competitors' customers back to the original vendor. This revenue stream is only a result of the original product's market dominance and not a result of any innate superiority.
3. Even when the original vendor is genuinely interested in promoting a healthy competition (so that he may also benefit from the resulting innovative market), post-facto interoperability may often be undesirable as it is not the result of a collaborative process and thus may contain many defects or quirks that can be directly traced back to the original implementation's technical limitations, but affect the customer negatively. In an open process, such limitations would have been easily identified and corrected, and the resulting cleaner specification used

by all vendors for higher-quality implementations. This is not possible post-facto, as customers already have valuable information and processes encoded in the faulty but dominant product, and other vendors are forced to replicate those faults and quirks even if they could design better solutions, for the sake of preserving interoperability.

4. Lack of an open standard can also become problematic for the customers, as in case of the original vendor's inability to fix a certain problem that is an artifact of technical limitations in the original product. The customer wants that fault fixed, but the vendor has to maintain that faulty state, even across newer revisions of the same product, because that behaviour is a de-facto standard and many more customers would have to pay the price of any break in interoperability caused by fixing the original problem and introducing new behaviour.

Telecommunications

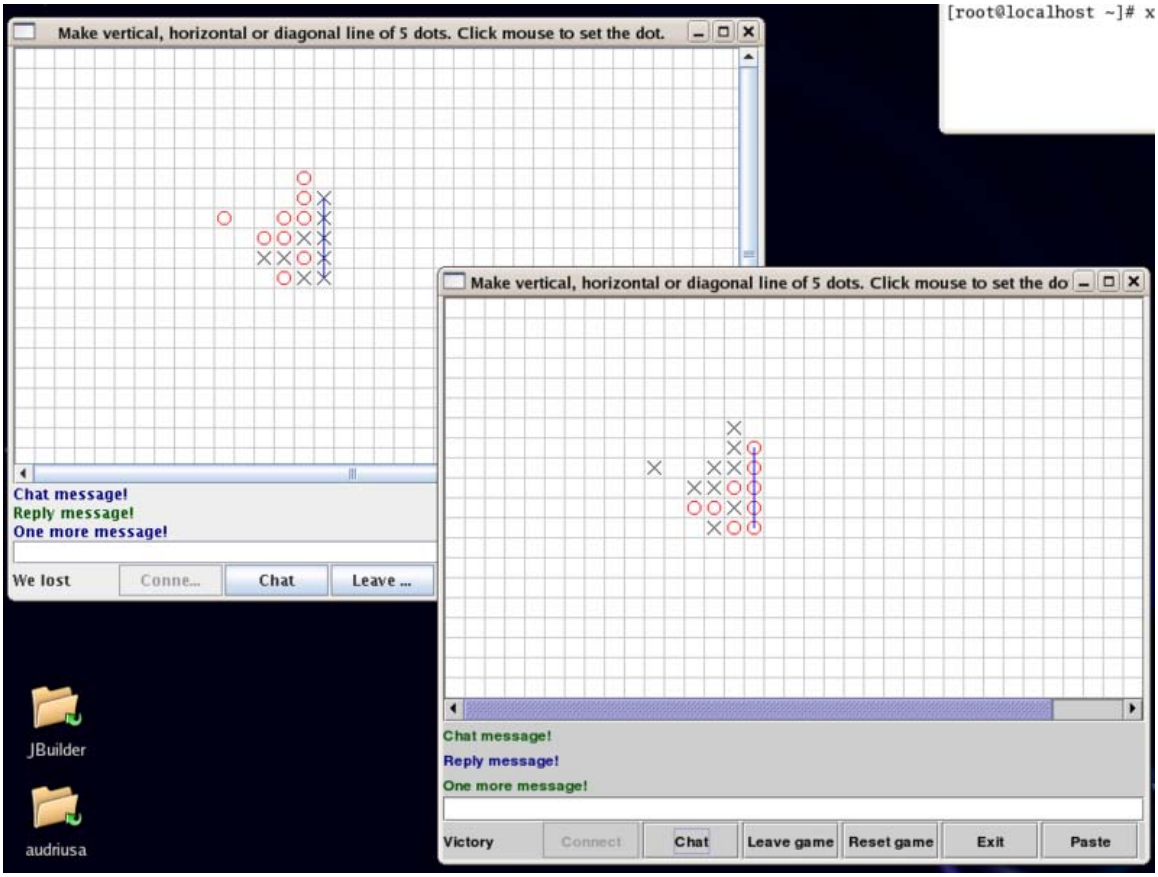
In telecommunication, the term can be defined as:

1. The ability of systems, units, or forces to provide services to and accept services from other systems, units or forces and to use the services exchanged to enable them to operate effectively together.
2. The condition achieved among communications-electronics systems or items of communications-electronics equipment when information or services can be exchanged directly and satisfactorily between them and/or their users. The degree of interoperability should be defined when referring to specific cases.

In two-way radio, interoperability is composed of three dimensions:

- compatible communications paths (compatible frequencies, equipment and signaling),
- radio system coverage or adequate signal strength, and;
- scalable capacity.

Software



Interoperability: playing the two role network game, when one of the player clients (top left) runs under Sun Microsystems and another under GNU Classpath with JamVM. The applications execute the same bytecode and interoperate using the standard RMI-IIOP messages for communication

With respect to software, the term interoperability is used to describe the capability of different programs to exchange data via a common set of exchange formats, to read and write the same file formats, and to use the same protocols. (The ability to execute the same binary code on different processor platforms is 'not' contemplated by the definition of interoperability.) The lack of interoperability can be a consequence of a lack of attention to standardization during the design of a program. Indeed, interoperability is not taken for granted in the non-standards-based portion of the computing world.

According to ISO/IEC 2382-01, *Information Technology Vocabulary, Fundamental Terms*, interoperability is defined as follows: *"The capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units".*

Note that the definition is somewhat ambiguous because the *user* of a program can be another program and, if the latter is a portion of the set of program that is required to be

interoperable, it might well be that it does need to have knowledge of the characteristics of other units. This definition focuses on the technical side of interoperability, while it has also been pointed out that interoperability is often more of an organizational issue: often interoperability has a significant impact on the organizations concerned, raising issues of ownership (do people want to share their data?), labor relations (are people prepared to undergo training?) and usability. In this context, a more apt definition is captured in the term "business process interoperability".

Interoperability can have important economic consequences, such as network externalities. If competitors' products are not interoperable (due to causes such as patents, trade secrets or coordination failures), the result may well be monopoly or market failure. For this reason, it may be prudent for user communities or governments to take steps to encourage interoperability in various situations. In the United Kingdom, for example, there is an eGovernment-based interoperability initiative called e-GIF while in the United States there is the NIEM initiative. Standards Defining Organizations (SDOs) provide open public software specifications to facilitate interoperability, and an example is Oasis-Open organization. As far as user communities, Neutral Third Party is creating standards for business process interoperability. Another example of a neutral party is the RFC documents from the Internet Engineering Task Force (IETF).

Medical industry

New technologies are being introduced in hospitals and labs at an ever-increasing rate, and many of these innovations have the potential to interact synergistically if they can be integrated effectively. The need for “plug-and-play” interoperability – the ability to take a medical device out of its box and easily make it work with one’s other devices – has attracted great attention from both healthcare providers and industry.

Interoperability helps patients get the most out of technology, and it also encourages innovation in the industrial sphere. When different products can be combined without complicated and expensive interfaces, small companies can enter a field and make specialized products. Without interoperability, hospitals are forced to turn to large vendors that provide suites of compatible devices but that do not specialize in any one area. Interoperability promotes competition, and competition encourages innovation and quality.

From the perspective of Intel, a major producer of consumer healthcare devices, there are six major factors that affect an industry’s ability to achieve interoperability. First there needs to be a demand for interoperable products. Second, there must be standards, or rules, defining what interoperability means in the field. Third, business conditions must encourage manufacturers to make their products interoperable. Fourth, guidelines must exist that make the often-complicated standards easier for companies to interpret. Fifth, compliance must be verified by independent testing; and finally, interoperability must be actively promoted. The rapid rise of wireless technology illustrates that interoperability is attainable.

Conditions in the biomedical industry are still in the process of becoming conducive to the development of interoperable systems. A potential market of interested hospitals exists, and standards for interoperability are being developed. Nevertheless, it seems that current business conditions do not encourage manufacturers to pursue interoperability. Only sixteen to twenty percent of hospitals, for example, use electronic medical records (EMR). With such a low rate of EMR adoption, most manufacturers can get away with not investing in interoperability. In fact, not pursuing interoperability allows some of them to tout the inter-compatibility of their own products while excluding competitors. By promoting EMR adoption, companies such as Intel hope to create an environment in which hospitals will have the collective leverage to demand interoperable products.

eGovernment

Speaking from an eGovernment perspective, interoperability refers to the collaboration ability of cross-border services for citizens, businesses and public administrations. Exchanging data can be a challenge due to language barriers, different specifications of formats and varieties of categorisations. Many more hindrances can be identified.

If data is interpreted differently, collaboration is limited, takes longer and is not efficient. For instance if a citizen of country A wants to purchase land in country B, the person will be asked to submit the proper address data. Address data in both countries include: Full name details, street name and number as well as a post code. The order of the address details might vary. In the same language it is not an obstacle to order the provided address data; but across language barriers it becomes more and more difficult. If the language requires other characters it is almost impossible, if no translation tools are available.

Hence eGovernment applications need to exchange data in a semantically interoperable manner. This saves time and money and reduces sources of errors. Fields of practical use are found in every policy area, be it justice, trade or participation etc. Clear concepts of interpretation patterns are required.

Many organizations are dedicated to interoperability. All have in common that they want to push the development of the World Wide Web towards the semantic web. Some concentrate on eGovernment, eBusiness or data exchange in general. In Europe, for instance, the European Commission and its IDABC programme issue the European Interoperability Framework. They also initiated the Semantic Interoperability Centre Europe (SEMIC.EU). A European Land Information Service (EULIS) was established in 2006, as a consortium of European National Land Registers. The aim of the service is to establish a single portal through which customers are provided with access to information about individual properties, about land and property registration services, and about the associated legal environment. In the United States, the government's CORE.gov service provides a collaboration environment for component development, sharing, registration, and reuse and related to this is the National Information Exchange Model (NIEM) work and component repository.

Public safety

Interoperability is an important issue for law enforcement, fire fighting, EMS, and other public health and safety departments, because first responders need to be able to communicate during wide-scale emergencies. Traditionally, agencies could not exchange information because they operated widely disparate hardware that was incompatible. Agencies' information systems such as computer-aided dispatch systems (CAD) and records management systems (RMS) functioned largely in isolation, so-called "information islands." Agencies tried to bridge this isolation with inefficient, stop-gap methods while large agencies began implementing limited interoperable systems. These approaches were inadequate and the nation's lack of interoperability in the public safety realm become evident during the 9/11 attacks on the Pentagon and World Trade Center structures. Further evidence of a lack of interoperability surfaced when agencies tackled the aftermath of the Hurricane Katrina disaster.

In contrast to the overall national picture, some states, including Utah, have already made great strides forward. The Utah Highway Patrol and other departments in Utah have created a statewide data-sharing network using technology from a company based in Bountiful, Utah, FATPOT Technologies.

The Commonwealth of Virginia is one of the leading states in the United States when it comes to improving interoperability and is continually recognized as a National Best Practice by Department of Homeland Security (DHS). Virginia's proven practitioner-driven Governance Structure ensures that all the right players are involved in decision making, training & exercises, planning efforts, etc. The Interoperability Coordinator leverages a regional structure to better allocate grant funding around the Commonwealth so that all areas have an opportunity to improve communications interoperability. Virginia's strategic plan for communications is updated yearly to include new initiatives for the Commonwealth - all projects and efforts are tied to this plan, which is aligned with the National Emergency Communications Plan, authored by Department of Homeland Security's Office of Emergency Communications (OEC).

The State of Washington seeks to enhance interoperability statewide. The State Interoperability Executive Committee (SIEC), established by the legislature in 2003, works to assist emergency responder agencies (police, fire, sheriff, medical, hazmat, etc.) at all levels of government (city, county, state, tribal, federal) to define interoperability for their local region.

Washington recognizes collaborating on system design and development for wireless radio systems enables emergency responder agencies to efficiently provide additional services, increase interoperability, and reduce long-term costs.

This important work saves the lives of emergency personnel and the citizens they serve.

The U.S. government is making a concerted effort to overcome the nation's lack of public safety interoperability. The Department of Homeland Security's Office for

Interoperability and Compatibility (OIC) is pursuing the SAFECOM and CADIP programs, which are designed to help agencies as they integrate their CAD and other IT systems.

The OIC launched CADIP in August 2007. This project will partner the OIC with agencies in several locations, including Silicon Valley. This program will use case studies to identify the best practices and challenges associated with linking CAD systems across jurisdictional boundaries. These lessons will create the tools and resources public safety agencies can use to build interoperable CAD systems and communicate across local, state, and federal boundaries.

Achieving software interoperability

Software Interoperability is achieved through five interrelated ways:

1. Product testing

   Products produced to a common standard, or to a sub-profile thereof, depend on clarity of the standards, but there may be discrepancies in their implementations that system or unit testing may not uncover. This requires that systems formally be tested in a production scenario – as they will be finally implemented – to ensure they actually will intercommunicate as advertised, i.e. they are interoperable. Interoperable product testing is different from conformance-based product testing as conformance to a standard does not necessarily engender interoperability with another product which is also tested for conformance.

2. Product engineering

   Implements the common standard, or a sub-profile thereof, as defined by the industry/community partnerships with the specific intention of achieving interoperability with other software implementations also following the same standard or sub-profile thereof.

3. Industry/community partnership

   Industry/community partnerships, either domestic or international, sponsor standard workgroups with the purpose to define a common standard that may be used to allow software systems to intercommunicate for a defined purpose. At times an industry/community will sub-profile an existing standard produced by another organization to reduce options and thus making interoperability more achievable for implementations.

4. Common technology and IP

   The use of a common technology or IP may speed up and reduce complexity of interoperability by reducing variability between components from different sets of

separately developed software products and thus allowing them to intercommunicate more readily. This technique has some of the same technical results as using a common vendor product to produce interoperability. The common technology can come through 3rd party libraries or open source developments.

5. Standard implementation

   Software interoperability requires a common agreement that is normally arrived at via a industrial, national or international standard.

Each of these has an important role in reducing variability in intercommunication software and enhancing a common understanding of the end goal to be achieved.

Interoperability as a question of power and market dominance

Interoperability tends to be regarded as an issue for experts and its implications for daily living are sometimes underrated. The European Union Microsoft competition case shows how interoperability concerns important questions of power relationships. In 2004, the European Commission found that Microsoft had abused its market power by deliberately restricting interoperability between Windows work group servers and non-Microsoft work group servers. By doing so, Microsoft was able to protect its dominant market position for work group server operating systems, the heart of corporate IT networks. Microsoft was ordered to disclose complete and accurate interface documentation, which will enable rival vendors to compete on an equal footing ("the interoperability remedy"). As of June 2005 the Commission is market testing a new proposal by Microsoft to do this, having rejected previous proposals as insufficient.

Interoperability has also surfaced in the Software patent debate in the European Parliament (June/July 2005). Critics claim that because patents on techniques required for interoperability are kept under RAND (reasonable and non discriminatory licensing) conditions, customers will have to pay license fees twice: once for the product and, in the appropriate case, once for the patent protected programme the product uses.

Railways

Railways have greater or lesser interoperability depending on conforming to standards of gauge, couplings, brakes, signalling, communications, loading gauge, operating rules, to mention a few parameters. North American railroads are highly interoperable, Europe, Asia, Africa, Central and South America, and Australia much less so. The parameter most difficult to overcome (at reasonable cost) is incompatibility of gauge, though variable gauge axle systems such as the SUW 2000 are starting to come to the rescue.

Chapter 2

Backward Compatibility

In the context of telecommunications and computing a device or technology is said to be **backward** or **downward compatible** if it can work with input generated by an older device. If products designed for the new standard can receive, read, view or play older standards or formats, then the product is said to be backward-compatible; examples of such a standard include data formats and communication protocols. Jocularly referred to as "hysterical raisins" i.e., a homophone like phrase for "historical reasons".

The reverse is forward compatibility, which implies that old devices allow (or are expected to allow) data formats generated by new (or future) devices, perhaps without supporting all new features. A standard supports forward compatibility if older product versions can receive, read, view or play the new standard.

For example, the introduction of FM stereo transmission allowed backward compatibility since new FM radio receivers could receive monaural signals generated by old transmitters. It also allowed forward compatibility, since old monaural FM radio receivers still could receive a signal from a new transmitter.

In programming languages, backward compatibility refers to the ability of a compiler for version N of the language to accept programs or data that worked under version N - 1. (By this definition, if previous versions (N - 1, N - 2, etc.) were also backward compatible, which is often the case, then, by induction, version N will also accept input that worked under any prior version after, and including, the latest one that was not backward compatible. However, in practice, features are often deprecated and support is dropped in a later release, which is yet thought of as backward compatible.)

In other contexts, a product or a technology is said to be **backward compatible** when it is able to fully take the place of an older product, by inter-operating with products that were designed for the older product.

Description

Backward compatibility is a relationship between two components, rather than being an attribute of just one of them. More generally, a new component is said to be backward compatible if it provides all of the functionality of the old component.

Backward compatibility is the special case of compatibility in which the new component has a direct historical ancestral relationship with the old component. If this special relationship does not exist then it not usually spoken of as "backward" compatibility but is instead just "compatible"—a consistent interface allowing interoperability between components and products that were each developed separately.

Data does nothing in the absence of an interpreter, so the notion of compatibility does not apply to document files, it only applies to software. In the case of a program that creates document files, a new version of that program ("v2") is said to be backward compatible with the old version of the program ("v1") when it can both read and write documents that work with v1. Everything that v1 could do must also be possible with v2, including saving documents that can be read by v1 (which is something that v1 could do.)

If a newer software version cannot save files that can be read by the older version, it is not backward compatible with the older version, although it may provide an irreversible upgrade capability for the old files. This situation has often been used strategically by software vendors to force customers to purchase upgrades since, over time, the number of data files usable by an old version diminishes at a rate proportional to the number of other customers that have upgraded (assuming that all customers generate files at the same the average rate.)

Levels of compatibility vary. In software, *binary compatibility* and *source compatibility* are distinguishable. Binary compatibility means that programs can work correctly with the new version of this library without requiring recompilation. Source compatibility requires recompilation but no changes to the source code.

Many platforms rely on emulation, the simulation of an older platform in software, to achieve backward compatibility.

Examples

- The NTSC color broadcast system was engineered by RCA to be backward compatible with black-and-white NTSC television sets.
- Most Blu-ray disc drives are able to play standard CD and DVD discs and most DVD drives are able to play standard CDs.
- Numerous video game consoles are backward compatible and are able to play the games created for predecessor consoles:
    - The Atari 7800 is backward compatible with almost all Atari 2600 games.

    - The Game Boy Advance line (except the Game Boy Micro) is backward compatible with previous Game Boy systems, meaning all Game Boy and Game Boy Color titles are playable on this system.
    - The Nintendo DS and the Nintendo DS Lite are backward compatible with all Game Boy Advance games.
    - The Nintendo 3DS is backward compatible with the Nintendo DS and Nintendo DSi software.
    - The PlayStation 2 is backward compatible with most of the original PlayStation library. Additionally, the initial PlayStation 3 model is backward compatible with most PlayStation and PlayStation 2 games. This is provided by the inclusion of the original Emotion Engine chip that is built inside the PS2. However, subsequent models removed this and the "Graphics Synthesizer" GPU, thus removing support for PS2 titles, but still able to play most PSX games.
    - The Xbox 360 is backward compatible with some Xbox games via software emulation.
    - The Wii is backward compatible with all games from the previous Nintendo system, the Nintendo GameCube, due to it being based on the PowerPC, the same base as the latter.
- Microsoft Windows contains application compatibility shims to make the platform compatible with most software from earlier 32-bit and 16-bit versions (e.g. *Civilization* (circa 1991, designed for Windows 3.0) running on Windows Vista).
- Microsoft Word 2000 was backward compatible with Word 97 because it could read and write files in Word 97 format, with the understanding that features unique to Word 2000 would not appear in Word 97.
- Several computer operating systems have various methods of running software originally designed for older versions or other OSs:
    - Windows NT and successors have various subsystems to run legacy applications. MS-DOS and Win16 subsystems (only on i386) can run some applications for those platforms, and it has an OS/2 subsystem for running CLI OS/2 applications.
    - With the introduction of Windows 7 (Business, Ultimate, or Enterprise editions), Windows XP Mode enables full compatibility with older programs supported under Windows XP via Microsoft Virtual PC.
- The modern Nikon F Mount SLR camera lenses from the late 1970s - present (its design dates back to 1959) can function on the newer Nikon DSLR cameras with some limitations.
- PCI Express 2.0 is backward compatible with PCI Express 1.1.
- The IBM 7080 transistorized computer was backward compatible with all models of the IBM 705 vacuum tube computer.

Chapter 3

Conceptual Interoperability & Business Process Interoperability

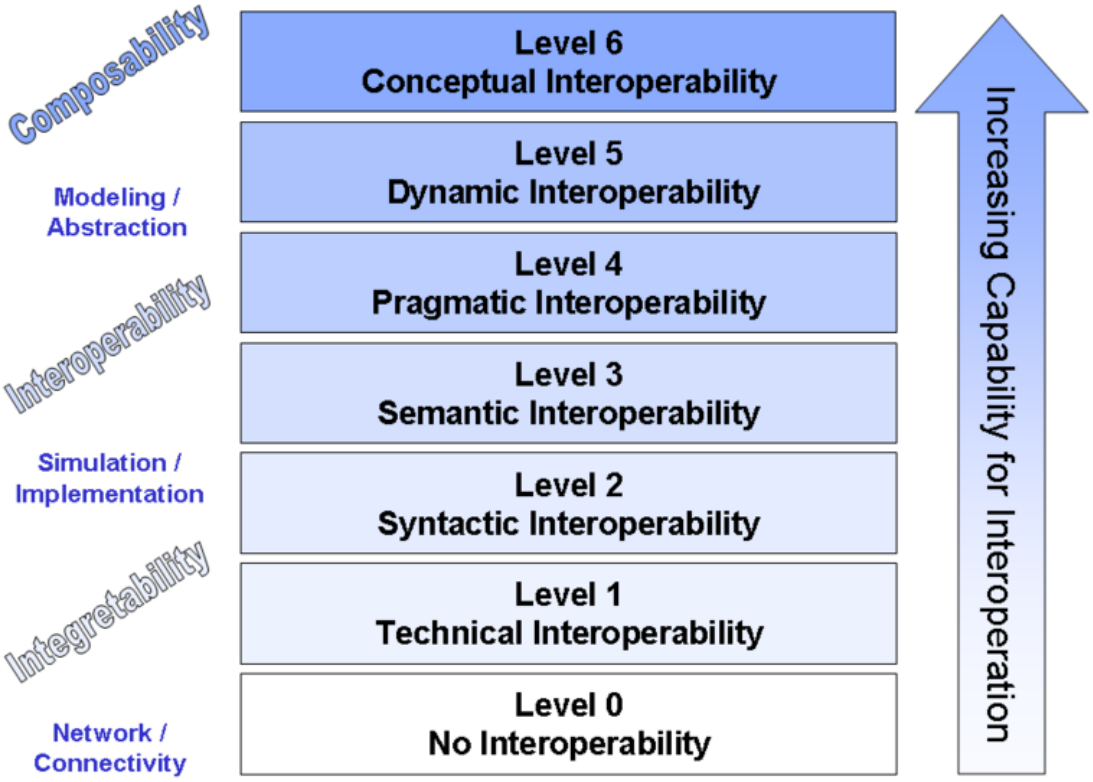
Conceptual Interoperability

Conceptual interoperability is a concept in simulation theory.

From the early ideas of Harkrider and Lunceford, simulation composability has been studied in more detail. Petty and Weisel formulated the current working definition: "Composability is the capability to select and assemble simulation components in various combinations into simulation systems to satisfy specific user requirements. The defining characteristic of composability is the ability to combine and recombine components into different simulation systems for different purposes." A recent RAND study provided a coherent overview of the state of composability for military simulation systems within the U.S. Department of Defense; many of its findings have much broader applicability.

Levels of conceptual interoperability

The resulting challenges have produced layered views. Petty and Weisel distinguish between the idea of *interoperability*, coping with the technical challenges, and *composability*, dealing with modeling issues. Research at the Virginia Modeling, Analysis and Simulation Center (VMASC) refined these layers to define the "Levels of Conceptual Interoperability Model (LCIM)," This definition has undergone gradual improvement since the first discussion in. The current version of LCIM was first documented in.



The different levels are characterized as follows:

- **Level 0:** Stand-alone systems have **No Interoperability**.
- **Level 1:** On the level of Technical Interoperability, a **communication protocol** exists for exchanging data between participating systems. On this level, a communication infrastructure is established allowing systems to exchange bits and bytes, and the underlying networks and protocols are unambiguously defined.
- **Level 2:** The Syntactic Interoperability level introduces a common structure to exchange information; i.e., a **common data format** is applied. On this level, a common protocol to structure the data is used; the format of the information exchange is unambiguously defined. This layer defines structure.
- **Level 3:** If a common information exchange reference model is used, the level of Semantic Interoperability is reached. On this level, the **meaning of the data** is shared; the content of the information exchange requests are unambiguously defined. This layer defines (word) meaning. There is a related but slightly different interpretation of the phrase semantic interoperability, which is closer to what is here termed *Conceptual Interoperability*, i.e. information in a form whose meaning is independent of the application generating or using it.
- **Level 4:** Pragmatic Interoperability is reached when the interoperating systems are aware of the **methods and procedures** that each system is employing. In other words, the use of the data – or the context of its application – is understood

by the participating systems; the context in which the information is exchanged is unambiguously defined. This layer puts the (word) meaning into context.

- **Level 5:** As a system operates on data over time, the state of that system will change, and this includes the **assumptions and constraints** that affect its data interchange. If systems have attained Dynamic Interoperability, they are able to comprehend the state changes that occur in the assumptions and constraints that each is making over time, and they are able to take advantage of those changes. When interested specifically in the effects of operations, this becomes increasingly important; the effect of the information exchange within the participating systems is unambiguously defined.
- **Level 6:** Finally, if the conceptual model – i.e. the assumptions and constraints of the meaningful abstraction of reality – are aligned, the highest level of interoperability is reached: Conceptual Interoperability. This requires that conceptual models are documented based on engineering methods enabling their interpretation and evaluation by other engineers. In essence, this requires a "fully specified, but **implementation independent model**" as requested by Davis and Anderson; this is not simply text describing the conceptual idea.

The LCIM shows that a layered approach to support composable services is necessary. The WS standards described earlier are not able to manage all levels, in particular not with the M&S specific upper layers. It is worth mentioning, however, that the LCIM focuses on technical support by information systems, such as command and control information systems in the military context. As Alberts and Hayes point out in, the organizational and social aspects are often even more important. Tolk proposes such a layered framework for measures of merits dealing with questions like tactical or strategic alignment of objectives or even political will of coalition partners in. Within this contribution, however, the focus will be on the information system aspects.

Page et al. suggest defining composability as the realm of the model and interoperability as the realm of the software implementation of the model. In addition, their research introduces integratability coping with the hardware-side and configuration side of connectivity. The author supports this categorization and recommends the following distinction when dealing with issues of simulation system interoperability, to include meaningful simulation-to-simulation system interoperation:

- **Integratability** contends with the physical/ technical realms of connections between systems, which include hardware and firmware, protocols, etc.
- **Interoperability** contends with the software- and implementation details of interoperations, including exchange of data elements based on a common data interpretation, etc.
- **Composability** contends with the alignment of issues on the modeling level. The underlying models are purposeful abstractions of reality used for the conceptualization being implemented by the resulting simulation systems.

This ideas complement the LCIM. The LCIM has been successfully applied not only in the domain of Modeling & Simulation, but generally in model-based interoperability

challenges. It should be pointed out that the LCIM can be used in descriptive and in prescriptive mode.

Business Process Interoperability

Business process interoperability (BPI) is a property referring to the ability of diverse business processes to work together, to so called "inter-operate".

It is a state that exists when a business process can meet a specific objective automatically utilizing essential human labor only. Typically, BPI is present when a process conforms to standards that enable it to achieve its objective regardless of ownership, location, make, version or design of the computer systems used.

Overview

The main attraction of BPI is that a business process can start and finish at any point worldwide regardless of the types of hardware and software required to automate it. Because of its capacity to offload human "mind" labor, BPI is considered by many as the final stage in the evolution of business computing. BPI's twin criteria of *specific objective* and *essential human labor* are both subjective.

Currently, business process interoperability is limited to enterprise software systems in which functions are designed to work together, such as a payroll module and a general ledger module that are part of the same program suite, and in controlled software environments that use EDI. Interoperability is also present between incompatible systems where middleware has been applied. In each of these cases, however, the processes seldom meet the test of BPI because they are constrained by information silos and the systems' inability to freely communicate among each other.

The objectives of BPI vary, but tend to fall into the following categories:

- Enable end-to-end straight-through processing ("STP") by interconnecting data and procedures trapped in information silos
- Let systems and products work with other systems or products without special effort on the part of the customer
- Increase productivity by automating human labor
- Eliminate redundant business processes and data replications
- Minimize errors inherent in manual processes
- Introduce mainstream enterprise software-as-a-service
- Give top managers a practical means of overseeing processes used to run business operations
- Encourage development of innovative Internet-based business processes
- Place emphasis on business processes rather than on the systems required to operate them
- Strengthen security by eliminating gaps among proprietary software systems
- Improve privacy by giving users complete control over their data

- Enable realtime enterprise scenarios and forecasts

History

The term "Business process interoperability" (BPI) is coined in the 1990s, mostly in connection with the value chain in electronic commerce. BPI has been utilized in promotional materials by various companies, and appears as a subject of research at organizations concerned with computer science ontologies. Yet despite the attention it has received, business process interoperability has not been applied outside of limited information system environments. A possible reason is that BPI requires universal conformance to standards so that a business process can start and finish at any point worldwide. The standards themselves are fairly straightforward—organizations use a finite set of shared processes to manage most of their operations. Bringing enterprises together to create and adopt the standards is another matter entirely. The world of management systems is, after all, characterized by information silos. Moving away from silos requires organizations to deal with cultural issues such as ownership and sharing of processes and data, competitive forces and security, not to mention the effect of automation on their work forces.

While the timetable or adoption of BPI cannot be predicted, it remains a subject of interest in organizations and think tanks alike.

Testing for BPI

To test for BPI, an organization analyzes a business process to determine if it can meet its *specific objective* utilizing *essential human labor* only.

The *specific objective* must be clearly defined from start to finish. Start and finish are highly subjective, however. In one organization, a process may start when a customer orders a product and finish when the product is delivered to the customer. In another organization, the same process may be preceded with product manufacture and distribution, and may be followed by management of after-sale warranty and repairs.

Essential human labor includes:

- Tasks that must be performed by people because no practical means of automation is available. Examples include fighting a fire, driving a bus and preparing a meal.
- Tasks that, in the opinion of management, are more effectively performed by people. Examples include answering a telephone call with a human voice and offering investment advice in person.
- Tasks where the cost of automation is greater than the cost of human labor.

To qualify for BPI, every process task must be taken into account from start to finish, including the labor that falls between the cracks created by incompatible software applications, such as gathering data from one system and re-inputting it in another, and

preparing reports that include data from disparate systems. The process must flow uninterrupted regardless of the underlying computerized systems used. If non-essential human labor exists at any point, the process fails the test of BPI.

Achieving BPI

To assure that business processes can meet their specific objectives automatically utilizing essential human labor only, BPI takes a “service-oriented architecture“ (SOA) approach, which focuses on the processes rather than on the technologies required to automate them. A widely-used SOA is an effective way to address the problems caused by information silos.

SOA makes practical sense because organizations cannot be expected to replace or modify their current enterprise software to achieve BPI, regardless of the benefits involved. Many workers' jobs are built around the applications they use, and most organizations have sizable investments in their current information infrastructures which are so complex that even the smallest modification can be very costly, time-consuming and disruptive. Even if software makers were to unite and conform their products to a single set of standards, the problem would not be solved. Besides software from well-known manufacturers, organizations use a great many legacy software systems, custom applications, manual procedures and paper forms. Without SOA, streamlining such a huge number of disparate internal processes so that they interoperate across the entire global enterprise spectrum is simply out of the question.

To create an SOA for widespread use, BPI relies on a centralized database repository containing shared data and procedures common to applications in every industry and geographical area. In essence, the repository serves as a top application layer, enabling organizations to export their data to its distributed database and obtain the programs they need by simply logging on via a portal. To assure security and commercial neutrality, the repository conforms to standards promulgated by the community of BPI stakeholders.

Organizations and interest groups that wish to achieve business process interoperability begin by establishing one or more BPI initiatives.

Chapter 4

Semantic Interoperability

Semantic Interoperability is a term used in computer science as a synonym for "Computable Semantic Interoperability". In this sense, it is the ability of computer systems to communicate information and have that information properly interpreted by the receiving system in the same sense as intended by the transmitting system. "Proper interpretation" means that the transmitted information will be used appropriately by a receiving computer system because the logical implications derivable from transmitted information will be the same as those that the sending system would derive. Semantic Interoperability requires that any two systems will derive the same inferences from the same information. This term is sometimes also used as a synonym for "General Semantic Interoperability", the ability of computer systems to place information in a public location and have that information properly interpreted by systems whose developers do not know the creators of the information nor the purpose for which it was created.

Semantic Interoperability is also use in a more general sense, as the ability of any communicating entities (not only computers) to share unambiguous meaning. In this broader sense, the **sender** must be able to reliably transmit all sufficient and necessary information; the **receiver** must be able to correctly interpret its interlocutor; and both must be aware of, and agree upon, each other behaviors for given interactions.

Semantic as a function of syntactic and pragmatic interoperability

Syntactic interoperability, provided by for instance XML or the SQL standards, is a pre-requisite to semantic. It involves a common data format and common protocol to structure any data so that the manner of processing the information will be interpretable from the structure. It also allows detection of syntactic errors, thus allowing receiving systems to request resending of any message that appears to be garbled or incomplete. No semantic communication is possible if the syntax is garbled or unable to represent the data. However, information represented in one syntax may in some cases be accurately

translated into a different syntax. Where accurate translation of syntaxes is possible, systems using different syntaxes may also interoperate accurately. In some cases the ability to accurately translate information among systems using different syntaxes may be limited to one direction, when the formalisms used have different levels of *expressivity* (ability to express information).

However, once the syntactical correctness has been verified, the intended meaning of the content of a communication still cannot be judged without *some* commonality in methods and procedures that each system is employing for *semantic* interpretation, which goes beyond the syntactic. In other words, the use of the data or context of application must be understood and unambiguously defined. The task of achieving semantic interoperabiilty among computer systems requires the use of a means to assure that, if there is any context sensitivity to the way terms are used, that the context must also be specified as part of the information using those terms.

Semantic interoperability may be achieved in an interactive manner among a limited group of systems that communicate with each other regularly, so as to allow refinement or clarification of meanings that are unclear, or addition of new meanings. Messages like "this doesn't seem to be an appropriate action at this moment" might provide feedback to the semantic interoperability layer of such interacting systems to suggest it has made errors. Over time, also, state of a system may change or the agreements that govern it may change as more and more systems are aligned. Messages that are semantically unambiguous at one time may be ambiguous if viewed later after the range of possible messages becomes more refined.

Just as computer programs are very often specified both from top-down user requirements or design precedents and bottom-up system capabilities (like shared libraries and APIs), semantics of a local system or group of systems in communication with each other often emerge from compromises between its syntax and its pragmatics.

The more ambitious goal of General Semantic Interoperability presupposes that such interactive refinements or clarifications of meaning are not possible. In that case, the meanings of any information available to multiple remote systems must be specified in sufficient detail to resolve any potential ambiguity. This requires the use of some common standard of meaning. The current best technology for achieving that level of precision in specification of meaning is the use of a logical representation at least as expressive a First-Order Logic (FOL). A common ontology allows all interoperating systems to specify meanings of terms with precision, by linking terms used in specific contexts to the ontology elements that describe the meanings of those terms in logical format.

A single ontology containing representations of every term used in every application is generally considered impossible, because of the rapid creation of new terms or assignments of new meanings to old terms. However, though it is impossible to anticipate *every* concept that a user may wish to represent in a computer, there is the possibility of finding some finite set of "primitive" concept representations that can be combined to

create any of the more specific concepts that users may need for any given set of applications or ontologies. Having a foundation ontology (also called *upper ontology*) that contains all those primitive elements would provide a sound basis for general semantic interoperability, and allow users to define any new terms they need by using the basic inventory of ontology elements, and still have those newly-defined terms properly interpreted by any other computer system that can interpret the basic foundation ontology. Whether the number of such primitive concept representations is in fact finite, or will expand indefinitely, is a question under active investigation. If it is finite, then a stable foundation ontology suitable to support accurate and general semantic interoperability can evolve after some initial foundation ontology has been tested and used by a wide variety of users. At the present time, no foundation ontology has been adopted by a wide community, so such a stable foundation ontology is still in the future.

Words and Meanings

One persistent misunderstanding recurs in discussion of semantics - the confusion of words and meanings. The meanings of words change, sometimes rapidly. But a formal language such as used in an ontology can encode the meanings (semantics) of concepts in a form that does not change. In order to determine what is the meaning of a particular word (or term in a database, for example) it is necessary to label each fixed concept representation in an ontology with the word(s) or term(s) that may refer to that concept. When multiple words refer to the same (fixed) concept, in language this is called synonymy; when one word is used to refer to more than one concept, that is called ambiguity. Ambiguity and synonymy are among the factors that make computer understanding of language very difficult. The use of words to refer to concepts (the meanings of the words used)is very sensitive to the context and the purpose of any use for many human-readable terms. The use of ontologies in supporting semantic interoperability is to provide a fixed set of concepts whose meanings and relations are stable and can be agreed to by users. The task of determining which terms in which contexts (each database is a different context) then is separated from the task of creating the ontology, and must be taken up by the designer of a database, or the designer of a form for data entry, or the developer of a program for language understanding. When a word used in some interoperability context changes its meaning, then to preserve interoperability it is necessary to change the pointer to the ontology element(s) that specifies the meaning of that word.

An example may clarify the point above. An initial ontology used for interoperability may specify that every instance of type 'Automobile' must have four supporting wheels. Later it is learned that automobiles exist or are developed that have only three wheels. Such three-wheeled automobiles will **not** be instances of that initial ontology concept. There are two ways to remedy the problem. The original ontology concept may be changed to include both three or four-wheeled versions, but that could cause errors in legacy systems that depend on the original meaning of the concept. Instead, one would preferably create a parent (more general) type in the ontology that includes both three and four-wheeled vehicles, and the original 'Automobile' type would then be made a subtype of that new type. To avoid name clashes, that new concept may be called, e.g.

"GenericAutomobile". The documentation of the original concept would be changed to alert users that a broader similar concept representation ('GenericAutomobile')is now available that includes three-wheeled vehicles. The linguistic term 'automobile' in any local terminology could be mapped to the more general type (if three-wheeled automobiles were intended), or optionally to the more restricted type having only four wheels, if that local community only wants to talk about four-wheeled automobiles. Making any change in an ontology can cause problems for legacy users; in this case, some users may want the meaning of 'automobile' to include three-wheeled automobiles, but used the original ontology concept because it was the closest concept available. When the new more general concept is added, all users need to be informed of the change so they can determine if they need to change the pointers from their local terms or data elements to the new ontology element.

This scenario illustrates the importance of trying to achieve stability in the ontology used to specify the meanings of terms used in applications, so as to achieve accurate semantic interoperability. For situations where an ontology is developed for use by a closely interacting group of users, direct communication of needs and requirements can assure that the ontology contains representations of all the concepts required by the users. For the general case of semantic interoperability among a wide range of users, who do not have direct contact with each other, the need to anticipate the requirements of users with many different purposes suggests the need to develop some common ontology, in consultation with a broad diversity of potential users, that has the capability of representing any of the concepts those users need, by combining the primitive concept representations. That is the purpose of a Foundation Ontology with a complete set of representations of the semantic primitives, as described in the previous section.

Partial semantic interoperability

To achieve perfect semantic interoperability, all communicating systems must use term (or symbol) definitions that are identical or can be accurately interconverted. Thus a common ontology is the ideal situation for semantic interoperability. Where that is impossible, lesser degrees of semantic interoperability may be achieved by techniques that automatically map the definitions used by one system to those of another.

Interoperability is sometimes considered as an all-or-nothing attribute of computer systems - *see upper ontology* - but for complex information, different levels of interoperability can be envisioned; when multiple pieces of information are being transferred, correct interpretation of some fraction of that information may be considered as constituting some level of semantic interoperability. Perfect semantic interoperability would require the correct interpretation of all transferred information, from the point of view of all users.

Knowledge representation requirements and languages

A knowledge representation language may be sufficiently expressive to describe nuances of meaning in well understood fields. There are at least five levels of complexity of these.

For general semi-structured data one may use a general purpose language such as XML.

For structured data with well understood relationships one may use SQL or the relational model.

A description logic (such as the one used in the OWL semantic web ontology language) is more complex.

Languages with the full power of first-order predicate logic may be required for many tasks.

Human languages are highly expressive, but are considered too ambiguous to allow the accurate interpretation desired, given the current level of human language technology. In human languages the same word may be used to refer to different concepts (ambiguity), and the same concept may be referred to by different words (synonymy).

Prior agreement not required

Semantic interoperability may be distinguished from other forms of interoperability by considering whether the information transferred has, in its communicated form, all of the meaning required for the receiving system to interpret it correctly, even when the algorithms used by the receiving system are unknown to the sending system. Consider sending one number:

If that number is intended to be the sum of money owed by one company to another, it implies some action or lack of action on the part of both those who send it and those who receive it.

It may be correctly interpreted if sent in response to a specific request, and received at the time and in the form expected. This correct interpretation does not depend only on the number itself, which could represent almost any of millions of types of quantitative measure, rather it depends strictly on the circumstances of transmission. That is, the interpretation depends on both systems expecting that the algorithms in the other system use the number in exactly the same sense, and it depends further on the entire envelope of transmissions that preceded the actual transmission of the bare number. By contrast, if the transmitting system does not know how the information will be used by other systems, it is necessary to have a shared agreement on how information with some specific meaning (out of many possible meanings) will appear in a communication. For a particular task, one solution is to standardize a form, such as a request for payment; that request would have to encode, in standardized fashion, all of the information needed to evaluate it, such as: the agent owing the money, the agent owed the money, the nature of the action giving rise to the debt, the agents, goods, services, and other participants in that action; the time of the action; the amount owed and currency in which the debt is reckoned; the time allowed for payment; the form of payment demanded; and other information. When two or more systems have agreed on how to interpret the information in such a request, they can achieve semantic interoperability *for that specific type of transaction*. For semantic

interoperability generally, it is necessary to provide standardized ways to describe the meanings of many more things than just commercial transactions, and the number of concepts whose representation needs to be agreed upon are at a minimum several thousand.

Ontology research

How to achieve semantic interoperability for more than a few restricted scenarios is currently a matter of research and discussion. For the problem of General Semantic Interoperability, some form of foundation ontology ('upper ontology') is required that is sufficiently comprehensive to provide the defining concepts for more specialized ontologies in multiple domains. Over the past decade more than ten foundation ontologies have been developed, but none have as yet been adopted by a wide user base.

The need for a single comprehensive all-inclusive ontology to support Semantic Interoperability can be avoided by designing the common foundation ontology as a set of basic ("primitive") concepts that can be combined to create the logical descriptions of the meanings of terms used in local domain ontologies or local databases. This tactic is based on the principle that:

If:

```
(1) the meanings and usage of the primitive ontology elements in the
foundation ontology are agreed on, and
(2) the ontology elements in the  domain ontologies are constructed as
logical
combinations of the elements in the foundation ontology,
```

Then:

```
The intended meanings of the domain ontology elements can be computed
automatically using an FOL reasoner, by any system that accepts the
meanings of the elements in the foundation ontology, and has both the
foundation ontology and the logical specifications of the elements in
the domain ontology.
```

Therefore:

```
Any system wishing to interoperate accurately with another system need
transmit only the data to be communicated, plus any logical
descriptions of terms used in that data that were created locally and
are not already in the common foundation ontology.
```

This tactic then limits the need for prior agreement on meanings to only those ontology elements in the common Foundation Ontology (FO). Based on several considerations, this is likely to be fewer than 10,000 elements (types and relations).

In practice, together with the FO focused on representations of the primitive concepts, a set of domain extension ontologies to the FO with elements specified using the FO

elements will likely also be used. Such pre-existing extensions will ease the cost of creating domain ontologies by providing existing elements with the intended meaning, and will reduce the chance of error by using elements that have already been tested. Domain extension ontologies may be logically inconsistent with each other, and that needs to be determined if different domain extensions are used in any communication.

Whether use of such a single foundation ontology can itself be avoided by sophisticated mapping techniques among independently developed ontologies is also under investigation.

Importance

The practical significance of semantic interoperability has been measured by several studies that estimate the cost (in lost efficiency) due to lack of semantic interoperability. One study, focusing on the lost efficiency in the communication of healthcare information, estimated that US$77.8 billion per year could be saved by implementing an effective interoperability standard in that area. Other studies, of the construction industry and of the automobile manufacturing supply chain, estimate costs of over US$10 billion per year due to lack of semantic interoperability in those industries. In total these numbers can be extrapolated to indicate that well over US$100 billion per year is lost because of the lack of a widely used semantic interoperability standard in the US alone.

There has not yet been a study about each policy field that might offer big cost savings applying semantic interoparability standards. But to see which policy fields are capable of profiting from semantic interoperability see 'Interoperability' in general. Such policy fields are eGovernment, health, security and many more. The EU also set up the Semantic Interoperability Centre Europe in June 2007.

Chapter 5

Forward Compatibility & Spatial Archive and Interchange Format

Forward Compatibility

Forward compatibility or **upward compatibility** (sometimes confused with extensibility) is the ability of a system to gracefully accept input intended for later versions of itself. The introduction of a forward compatible technology implies that old devices partly can understand data generated by new devices. The concept can be applied to electrical interfaces, telecommunication signals, data communication protocols, file formats, and computer programming languages. A standard supports forward compatibility if older product versions can receive, read, view, play or execute the new standard.

Although forward compatibility and extensibility are similar, they are not the same. A forward compatible system can accept data from a future version of itself and pick out the "known" part of the data. An example is a text-only word processor ignoring picture data from a future version. An extensible system is one that can be upgraded to fully handle the new data in the newer input format. An example is a text-only word processor that can be upgraded to handle picture data.

A forward-compatible system is expected to "gracefully" handle input which is intended for a newer version, by ignoring the unknowns and selecting the known subset of the data that the system is capable of handling. Forward compatibility is harder to achieve than backward compatibility because a system needs to cope gracefully with an unknown future data format or requests for unknown future features. Backward compatibility does not have this issue because it accepts a known data format.

Examples

Telecommunication standards

The introduction of FM stereo transmission, or color television, allowed forward compatibility, since monophonic FM radio receivers and black-and-white TV sets still could receive a signal from a new transmitter. It also allowed backward compatibility since new receivers could receive monophonic or black-and-white signals generated by old transmitters.

Another example of forward compatibility is a VHS or Betamax videocassette recorder, which ignores the high resolution information stored on a S-VHS or Super Betamax tape.

Video gaming

The Game Boy system was forward compatible with most games created for the Game Boy Color. (Clear-colored cartridges however housed games not backward compatible.) It is so far the only handheld system to feature forward compatibility.

Systems architecture

Many application software systems are designed with a robust and self-sufficient systems architecture so that they can operate adequately even when input for a more advanced version is entered.

In all cases, when the application system accepts the input data or operating system is not as expected, it will produce an output that will identify the problem accurately for the user.

Document formats

An example of forward compatibility is with a word processor. Assume that Version 1 of a word processor only allows text, and no graphics. It saves files with only information about the text typed, and the font, color, and size of the text. Let's say that the program adds the mark [VERSION1 END] to denote the end of the file. However, next year Version 2 is released that accepts graphics. However, the new word processor saves all of the text at the beginning of the file, puts the [VERSION1 END] mark, and then stores the picture data next, and puts a [VERSION2 END] mark after the picture data. The Version 1 word processor would still be able to read the text data up to the [VERSION1 END] mark, but would ignore the picture data afterward. When Version 3 is released that allows videos in the word processor file, it would save in this format: text data [VERSION1 END], picture data [VERSION2 END], and video data [VERSION3 END].

Adobe Reader / Adobe Acrobat

Although the above file design allows forward compatibility, there are additional features that can be added to be more useful to the user. One would be if Version 1 of a program printed a message that the file was created with a newer version of the software, and that some data was not available. The Adobe Reader program generates a message notifying the user of a PDF file that it was created in a newer version of Adobe Acrobat, and some features will not be available.

MS Word

Another useful feature is if Version 1 of a program offered to download a viewer or converter that allows the user to at least read files from newer versions of the program, even though the user may not be able to edit them. In the above word processor example, Version 1 would download updates from the internet that allows the user to see the pictures and videos in later versions of the word processor, even though the user cannot add, edit, or modify the multimedia data, due to that functionality not being in Version 1. An example of this functionality is Microsoft Word. When a document is created in Microsoft Word 2007 and opened in an earlier version (like Microsoft Word 2003), Microsoft Word 2003 tells the user it can download a converter to read files in the newer Microsoft Word 2007 format. This allows Microsoft Word 2003 to read data created by Microsoft Word 2007, even though the user cannot use Microsoft Word 2003 to build new data in the advanced format.

PNG

An example of forward compatibility is the Portable Network Graphics (PNG) format, which divides data into "chunks", and indicates whether these are "critical" or "ancillary", where ancillary chunks can be ignored by programs that do not understand them.

More subtly, it also indicates whether chunks are safe to copy by readers that do not recognize them – thus ensuring that data does not become out of sync.

Spatial Archive and Interchange Format

The Spatial Archive and Interchange Format (SAIF, pronounced *safe*) was defined in the early 1990s as a self-describing, extensible format designed to support interoperability and storage of geospatial data.

SAIF Dataset

SAIF has two major components that together define SAIFtalk. The first is the Class Syntax Notation (CSN), a data definition language used to define a dataset's schema. The second is the Object Syntax Notation (OSN), a data language used to represent the object data adhering to the schema. The CSN and OSN are contained in the same physical file,

along with a directory at the beginning of the file. The use of ASCII text and a straightforward syntax for both CSN and OSN ensure that they can be parsed easily and understood directly by users and developers. A SAIF dataset, with a .saf or .zip extension, is compressed using the zip archive format.

Schema Definition

SAIF defines 285 classes (including enumerations) in the Class Syntax Notation, covering the definitions of high-level features, geometric types, topological relationships, temporal coordinates and relationships, geodetic coordinate system components and metadata. These can be considered as forming a base schema. Using CSN, a user defines a new schema to describe the features in a given dataset. The classes belonging to the new schema are defined in CSN as subclasses of existing SAIF classes or as new enumerations.

A *ForestStand::MySchema* for example could be defined with attributes including age, species, etc. and with *ForestStand::MySchema* specified as a subclass of *GeographicObject*, a feature defined in the SAIF standard. All user defined classes must belong to a schema, one defined by the user or previously existing. Different schemas can exist in the same dataset and objects defined under one schema can reference those specified in another.

Inheritance

SAIF supports multiple inheritance, although common usage involved single inheritance only.

Object Referencing

Object referencing can be used as a means of breaking up large monolithic structures. More significantly it can allow objects to be defined only once and then referenced any number of times. A section of the geometry of the land-water interface could define part of a coastline as well as part of a municipal boundary and part of a marine park boundary. This geometric feature can be defined and given an object reference, which is then used when the geometry of the coastline, municipality and marine park are specified.

Multimedia

Multimedia objects can also be objects in a SAIF dataset and referenced accordingly. For example, image and sound files associated with a given location could be included.

Model Transformations and Related Software Applications

The primary advantage of SAIF was that it was inherently extensible following object oriented principles. This meant that data transfers from one GIS environment to another did not need to follow the lowest common denominator between the two systems.

Instead, data could be extracted from a dataset defined by the first GIS, transformed into an intermediary, i.e., the semantically rich SAIF model, and from there transformed into a model and format applicable to the second GIS.

This notion of model to model transformation was deemed to be realistic only with an object oriented approach. It was recognized that scripts to carry out such transformations could in fact add information content. When Safe Software developed the Feature Manipulation Engine (FME), it was in large measure with the express purpose of supporting such transformations. The FMEBC was a freely available software application that supported a wide range of transformations using SAIF as the hub. The FME was developed as a commercial offering in which the intermediary could be held in memory instead of as a SAIF dataset.

History

The SAIF project was established as a means of addressing interoperability between different geographic information systems. Exchange formats of particular prominence at the time included DIGEST (Digital Geographic Information Exchange Standard) and SDTS (Spatial Data Transfer Specification, later accepted as the Spatial Data Transfer Standard). These were considered as too inflexible and difficult to use. Consequently, the Government of British Columbia decided to develop SAIF and to put it forward as a national standard in Canada.

SAIF became a Canadian national standard in 1993 with the approval of the Canadian General Standards Board. The last version of SAIF, published in January 1995, is designated as CGIS-SAIF Canadian Geomatics Interchange Standard: Spatial Archive and Interchange Format: Formal Definition (Release 3.2), issue CAN/CGSB-171.1-95, catalogue number P29-171-001-1995E.

The work on the SAIF modeling paradigm and the CSN classes was carried out principally by Mark Sondheim, Henry Kucera and Peter Friesen, all with the British Columbia government at the time. Dale Lutz and Don Murray of Safe Software developed the Object Syntax Notation and the Reader and Writer software that became part of the Feature Manipulation Engine.

SAIF was brought to the attention of Michael Stonebraker and Kenn Gardels of the University of California at Berkeley, and then to those working on the initial version of the Open Geospatial Interoperability Specification (OGIS), the first efforts of what became the Open Geospatial Consortium (OGC). A series of 18 submissions to the ISO SQL Multimedia working group also helped tie SAIF to the original ISO work on geospatial features.

Today SAIF is of historical interest only. It is significant as a precursor to the Geography Markup Language and as the formative element in the development of the widely used Feature Manipulation Engine.

Chapter 6

Schools Interoperability Framework & Protocol Converter

Schools Interoperability Framework

The **Schools Interoperability Framework**, **Systems Interoperability Framework** (UK), or **SIF**, is a data sharing open specification for academic institutions from kindergarten through twelfth grade (K-12). Until recently, it has been used primarily in the United States alone; however, it is increasingly being implemented in Australia, the UK, India and elsewhere.

The specification is composed of two parts: an XML specification for modeling educational data, and a Service-Oriented Architecture (SOA) specification for sharing that data between institutions.

SIF is not a product, but an industry initiative that enables diverse applications to interact and share data. As of March 2007, SIF is estimated to have been used in more than 48 states and 6 countries, supporting five million students.

The specification is actively maintained by its specification body, the Schools Interoperability Framework Association.

History and background

Traditionally, the standalone applications used by public school districts have the limitation of data isolation; that is, it is difficult to access and share their data. This often results in redundant data entry, data integrity problems, and inefficient or incomplete reporting. In such cases, a student's information can appear in multiple places but may not be identical, for example, or decision makers may be working with incomplete or inaccurate information. Many district and site technology coordinators also experience an

increase in technical support problems from maintaining numerous proprietary systems. SIF was created to solve these issues.

The Schools Interoperability Framework (SIF) began as an initiative chiefly championed initially by Microsoft to create "a blueprint for educational software interoperability and data access." It was designed to be an initiative drawing upon the strengths of the leading vendors in the K-12 market to enable schools IT professionals to build, manage and upgrade their systems. It was endorsed by close to 20 leading K-12 vendors of student information, library, transportation, food service applications and more. The first pilot sites began in the summer of 1999, and the first SIF-based products began to show up in 2000.

In the beginning it was not clear which approach would become the national standard in the United States. Both SIF and EDI were vying for the position in 2000 but SIF began taking the lead in 2002 or so.. In 2000, the National School Boards Association held a panel discussion during its annual meeting on the topic of SIF.

In 2007 in the United Kingdom Becta has championed the adoption of SIF as a national standard for schools data interchange. .

In 2008 it was announced that in the UK the standard will become known as the "Systems Interoperability Framework". This reflects the intention in the UK to develop SIF to be used in other organizations beyond just schools.

Strengths

Some features of SIF that make it well-suited for data interoperability are:

- It is an XML standard that exists built entirely and specifically for the exchange of K-12 education-related information;
- Case studies show significant dollar savings for schools and districts;
- Many school districts require that K-12 data vendors use SIF, and some states like Oklahoma are even legislating its use;
- Many vendors in the K-12 space already have SIF agents that have met SIF certification criteria and several Zone Integration Server products "ZIS" are available in the market. With many SIF agents available & capable of interacting with the ZIS, barriers for school district use have been significantly reduced;

Criticism

SIF has all the pains and challenges that come with any SOA specification and data model. When building specifications via consensus not everyone is always happy and sometimes the end product isn't perfect. Also given all the moving parts in modeling the entire K12's enterprise the specification has many points of possible failure. This is not particular to SIF but to any record-level, automated system moving standardized data from one source to another in a heterogeneous environment. Out-of-the-box

interoperability and ease of use and implementation are part of a 12-18 month focus from 2007 and through 2009.

How SIF works

Rather than have each application vendor try to set up a separate connection to every other application, SIF has defined the set of rules and definitions to share data within a "SIF Zone"—a logical grouping of applications in which software application agents communicate with each other through a central communication point. Zones are managed by a piece of software called a Zone Integration Server (ZIS). A single ZIS can manage multiple Zones.

Data travels between applications as a series of standardized messages, queries, and events written in XML and sent using Internet protocols. The SIF specification defines such events and the "choreography" that allows data to move back and forth between the applications.

SIF Agents are pieces of software that exist either internal to an application or installed next to it. The SIF Agents function as extensions of each application and serve as the intermediary between the software application and the SIF Zone. The ZIS keeps track of the Agents registered in the Zone and manages transactions between Agents, enabling them to provide data and respond to requests. The ZIS controls all access, routing, and security within the system. Standardization of the behavior of the Agents and ZIS means that SIF can add standard functionality to a Zone by simply adding SIF-enabled applications over time.

Vertical interoperability

"Vertical interoperability" is a situation in which SIF agents at different levels of an organization communicate using a SIF Zone. Vertical interoperability involves data collection from multiple agents (upward) or publishing of information to multiple agents (downward). For example, a state-level data warehouse may listen for changes in district-level data warehouses and update its database accordingly. Or a state entity may wish to publish teacher certification data to districts. The three pieces of the SIF specification that deal directly with vertical interoperability are the Student Locator object, the Vertical Reporting object, and the Data Warehouse object.

SIF in relation to other standards

SIF was designed before SOAP, namespaces, and web service standards were as mature as they are today. As a result it has a robust SOA that is more vetted than the current SOAP specifications but does not use the SOAP or WS standards. The 2.0 SIF Web Services specification begins the process of joining these two worlds.

The 2.0 Web Services specification allows for more generalized XML messaging structures typically found in enterprise messaging systems that use the concept of an

enterprise service bus. Web service standards are also designed to support secure public interfaces and XML appliances can make the setup and configuration easier. The SIF 2.0 Web Services specification allows for the use of Web Services to communicate in and out of the Zone.

SIFA is also working closely with the Post-Secondary Education Standards Council (PESC), SCORM, and other standards organizations.

Future versions of SIF may integrate more standards and multiple namespaces.

Versions

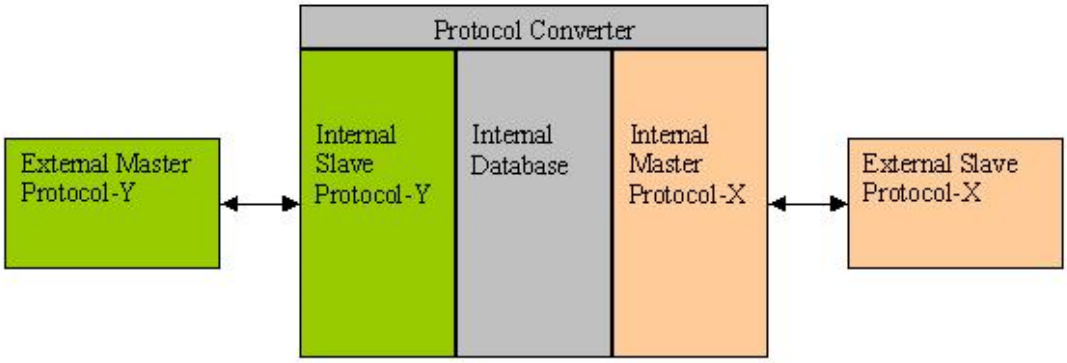
The version 2.4 specification is the latest version of SIF. It is expected that 2.x application environments will arise in the near future as vendors start designing, developing, and implementing 2.x agents, and states, districts, and schools start adopting them. Most of the SIF Zone Integration Server vendors are currently or will be putting 1.5r1 to 2.x migration functionality in place for their clients.

A future major version of SIF is being worked on right now, code-named "Columbus". Its emphasis will be on ease of use and implementation.

Protocol Converter

A **Protocol Converter** is a device used to convert standard or proprietary protocol of one device to the protocol suitable for the other device or tools to achieve the interoperability. Protocols are software installed on the routers which convert the data formats, data rate and protocols of one network into the protocols of the network in which data is navigating.There are varieties of protocols used in different fields like Power Generation, Transmission & Distribution, Oil & Gas, Automaton, Utilities, AMR, and Remote Monitoring applications. The major protocol translation messages involve conversion of data messages, events, commands and time synchronization.

General Architecture



The general architecture of a protocol converter include an internal master protocol communicating to the external slave devices and the data collected is used to update the internal database of the converter. When the external master requests for data, the internal slave collects the same from the database and send it to the external master. There will be different schemes for handling the spontaneous reporting of events and commands. There can be different physical medium for communication on protocol-X & Y which include RS232, RS485, Ethernet, etc.

Applications of Protocol Converters

Protocol Converter applications vary from industry to industry. The protocol converter can be a software converter, hardware converter, or an integrated converter depending on the protocols.

- Some of the key applications are:
  - Substation Automation
  - Building Automation
  - Process Automation

The major protocols used in each area of application are listed under List of automation protocols.

Latency & Engineering Issues in using Protocol Converters

Protocol Converters are generally used for transforming data and commands from one device or application to another. This necessarily involves transformation of data, commands, their representation, encoding and framing to achieve the conversion.

There are simple and complex types of conversions depending on the application and domain in which this is being used. The simplest and most commonly used conversion is protocol conversion between Modbus RTU and Modbus TCP. In this conversion, there is no change in the overall framing. Hence it is easy to take the Serial Modbus RTU frame and encapsulate it in a TCP/UDP Socket and send it over Ethernet. Since both the protocol framings are the same, except for the actual physical layer transmission, both the application layers will interpret data similarly as long as the communication interfaces are made transparent.

However, there do exist very complex conversions, for example: where the data is formatted, the data types supported, the object models, etc. They are so different that the conversion engine needs to make modifications not only in framing, but in mapping information for each type of data, command, and in some cases, the object models. Also, there might be user configurations required in defining the mapping of supported and non-supported data types

These transformations, however, bring about conversion advantages, communication delay, processing latency, and an overall end to end processing time which is finite and needs to be considered in all solution designs.

The latency of end to end communication depends on the processing delay of the hardware and/or software being used, the protocol & conversion complexity, and the solution architecture. These latencies can vary for typical industrial and energy automation applications from 10-20 milli-seconds to as high as 1 second. Solution architectures using protocol converters need to consider this latency and how it will impact the project for which converters are being considered.

Also, the majority of such architectures would involve configuration and mapping which both require considerable engineering effort and engineering time. These need to be considered while defining project schedules.

Chapter 7

Web Services Interoperability Technology, Web Services Interoperability & Web Interoperability

Web Services Interoperability Technology

Web Services Interoperability Technology (WSIT) is an open-source project started by Sun Microsystems to develop the next-generation of web service technologies.

It consists of Java programming language APIs that enable advanced WS-* features to be used in a way that is compatible with Microsoft's Windows Communication Foundation (WCF) as used by .NET.

WSIT is distributed under the terms of the CDDL open-source license, and is currently under development as part of project Metro.

WSIT is a series of extensions to the basic SOAP protocol, and so uses JAX-WS and JAXB. It is not a new protocol such as the binary DCOM.

WSIT implements the WS-I specifications, including:

- Metadata
  - WS-MetadataExchange
  - WS-Transfer
  - WS-Policy
- Security
  - WS-Security
  - WS-SecureConversation
  - WS-Trust
  - WS-SecurityPolicy
- Messaging
  - WS-ReliableMessaging

    - WS-RMPolicy
- Transactions
    - WS-Coordination
    - WS-AtomicTransaction

Web Services Interoperability

The **Web Services Interoperability Organization** (WS-I) is an industry consortium chartered to promote interoperability amongst the stack of web services specifications. WS-I does not define standards for web services; rather, it creates guidelines and tests for interoperability.

It is governed by a Board of Directors consisting of the founding members (IBM, Microsoft, BEA Systems, SAP, Oracle, Fujitsu, Hewlett-Packard, and Intel) and two elected members (currently, Sun Microsystems and webMethods).

The organization's deliverables include profiles, sample applications that demonstrate the profiles' use, and test tools to help determine profile conformance.

WS-I Profiles

According to WS-I, a profile is

A set of named web services specifications at specific revision levels, together with a set of implementation and interoperability guidelines recommending how the specifications may be used to develop interoperable web services.

- WS-I Basic Profile
- WS-I Basic Security Profile
- Simple Soap Binding Profile

WS-I Test tools

The WS-I issued a test tool suite in 2004. Two tools are included :

- A monitor designed to intercept live SOAP messages and the associated headers during a test session. This functionality is ensured through the use of the man in the middle principle.
- An analyzer designed to analyze profile conformance of a Web Service artifacts. The profile is chosen with a Test Assertion Document (*.TAD) file. The different artifacts are :
    - The Web Service description file, actually a WSDL file

- The Web Service discovery artifact, actually a UDDI entry
- The messages and associated envelopes exchanged during a test session and captured with the test tools

These test tools are not designed to be used as a full certification tool. As stated in the user guide (*WS-I Testing Tools version 1.1 User Guide*) bundled with the test tools., they can only be used as indicator of profile compliance:

Question: Can testing tools certify that a Web Service is conforming to the Profile?
Answer: The tools can only verify the conformance of Web Service artifacts that are produced during a testing session. Some artifacts belong to the definition of the Web Service (WSDL); some others result from the observable behavior of the Web Service at run-time. It is rather difficult to test all possible behaviors that a Web Service can exhibit, mostly because exercising these behaviors is application-dependent and requires an application-level understanding of the Web Service. For these reasons, the Testing WS-I working group has not attempted to provide certification criteria.

The testing tools are then an indicator of conformance of a Web Service to the Profiles selected, based on the artifacts produced. In turn, this is an indicator of interoperability with other business partners who also have tested as conforming to the Profiles.

WS-I Profile Compliance

The WS-I is not a certifying authority thus every vendor can claim to be compliant to a profile. However the use of the test tool is required before a company can claim a product to be compliant.

In a 2003 interview , the WS-I spokesman said even if every companies are free to claim compliance unfaithfully, he expects companies to be honest:

"We expect enforcement of that brand to be market-driven. We suspect no one wants to be the first person to be called on for making a bad claim."

Web Interoperability

Web interoperability means producing web pages viewable in standard compatible web browsers, various operating systems such as Windows, Macintosh and Linux and devices such as PC, PDA and mobile phone based on the latest web standards.

History

This term was originated by the Web Interoperability Pledge that is a promise to adhere to current HTML Recommendations as promulgated by the World Wide Web Consortium (W3C). The WIP was not a W3C initiative. but it was started by and has been run by ZDNet AnchorDesk quite independently.

This issue was known by cross browsing in browser war between Internet Explorer and Netscape. Windows Internet Explorer was the dominant browser after that, but modern web browsers such as Mozilla Firefox, Opera and Safari have supported web standards. Because of backward compatibility of Internet Explorer, many web pages has supported non-standard HTML tags and DOM handling script yet as well platform-dependent techniques such as ActiveX. These are very harmful for Web accessibility and Device Independence.

Elements of Web interoperability

- Structural and semantic markup with XHTML.
- CSS based layout with layout elements such as position and float.
- Separating among structure, presentation and behavior in web pages.
- DOM scripting based on W3C DOM Standard and ECMAScript.

Activities

It has been various activities, for example Web Standards Project, Mozilla's Technology Evangelism and Web Standards Group. Also there are educational activities such as Web Essential Conference.

Chapter 8

System Integration & Deprecation

System Integration

In engineering, **system integration** is the bringing together of the component subsystems into one system and ensuring that the subsystems function together as a system. In information technology, **systems integration** is the process of linking together different computing systems and software applications physically or functionally, to act as a coordinated whole.

The system integrator brings together discrete systems utilizing a variety of techniques such as computer networking, enterprise application integration, business process management or manual programming.

Overview

A system is an aggregation of subsystems cooperating so that the system is able to deliver the over-arching functionality. System integration involves integrating existing (often disparate) subsystems. The subsystems will have interfaces. Integration involves joining the subsystems together by "gluing" their interfaces together. If the interfaces don't directly interlock, the "glue" between them can provide the required mappings. System integration is about determining the required "glue".

System integration is also about adding value to the system, capabilities that are possible because of interactions between subsystems.

In today's connected world, the role of system integration engineers is becoming more and more important: more and more systems are designed to connect together, both within the system under construction and to systems that are already deployed.

Required skills

A system integration engineer needs a broad range of skills and is likely to be defined by a breadth of knowledge rather than a depth of knowledge. In an effort to better quantify the required skillset of a competent integrator by today's standards, Eric Wick developed a matrix template in 2010 to assess the capabilities of integrators interested in pursuing careers in the field.

These skills are likely to include software and hardware engineering, interface protocols, and general problem solving skills. It is likely that the problems to be solved have not been solved before except in the broadest sense. They are likely to include new and challenging problems with an input from a broad range of engineers where the system integration engineer "pulls it all together."

Methods of integration

Vertical Integration (as opposed to "horizontal") is the process of integrating subsystems according to their functionality by creating functional entities also referred to as silos. The benefit of this method is that the integration is performed quickly and involves only the necessary vendors, therefore, this method is cheaper in the short term. On the other hand, cost-of-ownership can be substantially higher than seen in other methods, since in case of new or enhanced functionality, the only possible way to implement (scale the system) would be by implementing another silo. Reusing subsystems to create another functionality is not possible.

Star Integration or also known as **Spaghetti Integration** is a process of integration of the systems where each system is interconnected to each of the remaining subsystems. When observed from the perspective of the subsystem which is being integrated, the connections are reminiscent of a star, but when the overall diagram of the system is presented, the connections look like spaghetti, hence the name of this method. The cost varies due to the interfaces which subsystems are exporting. In a case where the subsystems are exporting heterogeneous or proprietary interfaces, the integration cost can substantially rise. Time and costs needed to integrate the systems increase exponentially when adding additional subsystems. From the feature perspective, this method often seems preferable, due to the extreme flexibility of the reuse of functionality.

Horizontal Integration or **Enterprise Service Bus** (ESB) is an integration method in which a specialized subsystem is dedicated to communication between other subsystems. This allows cutting the number of connections (interfaces) to only one per subsystem which will connect directly to the ESB. The ESB is capable of translating the interface into another interface. This allows cutting the costs of integration and provides extreme flexibility. With systems integrated using this method, it is possible to completely replace one subsystem with another subsystem which provides similar functionality but exports different interfaces, all this completely transparent for the rest of the subsystems. The only action required is to implement the new interface between the ESB and the new subsystem.

The horizontal scheme can be misleading, however, if it is thought that the cost of intermediate data transformation or the cost of shifting responsibility over business logic can be avoided.

Example

The process of systems integration is successfully done in construction engineering. Broad arrays of different engineering fields are simultaneously considered in the planning stage of a project. But all of these areas are adjacent to each other. For instance the structural analysis, construction physics, inherent material science, actual construction course and logistics are all members of a rather small group. A difference would be the simultaneous consideration of areas distributed over the professions of the engineers involved and the architects, landscape architects, maintenance specialist, utility-provider and an scientist rendering the lifecycle balance of the project. Such an Multidisciplinary approach cross all borders of different professions could be regarded as Multi-disciplinary Engineering (MDE). Such a level of integration has the potential to solve problems on an adequate level, achieving much more than financial gain. The number of areas covered in a MDE process is much higher, too.

Deprecation

In computer software or authoring programs standards and documentation, the term **deprecation** is applied to software features that are superseded and should be avoided. Although deprecated features remain in the current version, their use may raise warning messages recommending alternative practices, and deprecation may indicate that the feature will be removed in the future. Features are deprecated—rather than being removed—in order to provide backward compatibility and give programmers who have used the feature time to bring their code into compliance with the new standard.

Reasons for deprecation

Programmers or standards-makers may choose to deprecate a feature for any number of reasons. Some common cases are:

- *The feature has been replaced by a more powerful, alternative feature.* For instance, the Linux kernel contains two modules to communicate with Windows networks — `smbfs` and `cifs`. The latter provides better security, supports more protocol features and integrates better with the rest of the kernel. Since the inclusion of `cifs`, `smbfs` has been deprecated.
- *The feature contains a design flaw—frequently a security flaw—and so should be avoided, but existing code depends upon it.* The C standard function `gets()` is an example of this. Using this function can introduce a buffer overflow into the program that uses it. However, it cannot be removed as it is part of the C standard, and a compiler which lacks it would be out of conformance. Therefore, compilers raise warnings when it is used.

- *The feature is considered extraneous, and will be removed in the future in order to simplify the system as a whole.* Early versions of the Web markup language HTML included a `FONT` element, to allow page designers to specify the font in which text should be displayed. With the release of Cascading Style Sheets and HTML 4.0, the `FONT` element became extraneous, and detracted from the benefits of noting structural markup in HTML and graphical formatting in CSS. Thus, the `FONT` element was deprecated in the *Transitional* HTML 4.0 standard, and eliminated in the *Strict* variant.
- *A future version of the software is planned to make major structural changes, which make it impossible (or impractical) to support older features.* For instance, when Apple Inc. planned the transition from Mac OS 9 to Mac OS X, it created a subset of the older system's API which would support most programs with minor changes. This became the Carbon library, available in both Mac OS 9 and Mac OS X. Programmers who were, at the time, chiefly using Mac OS 9, could ensure that their programs would run natively on Mac OS X by using only the API functions in Carbon. Other Mac OS 9 functions were deprecated, and were never supported natively in Mac OS X.
- *Standardization or increased consistency in naming.* Projects that are developed over long periods of time, or by multiple individuals or groups, can contain inconsistencies in the naming of various items. These can be the result of a lack of foresight, changes in nomenclature over time, or personal, regional or educational differences in terminology. Since merely renaming an item would break backwards compatibility, the existing name must be left in place. The original name will likely remain indefinitely, but will be deprecated to encourage use of the newer, more consistent naming convention. An example would be an API that alternately used the spelling "color" and "colour". Standardization would result in the use of only one of the regional spellings throughout, and all occurrences of the other spelling would be deprecated.
- A feature that once was only available independently is now combined with its co-feature. An example being VLC Media Player, VLC used to stand for 'VideoLan Client' and a 'VideoLan Server' was available as its co-feature. Both the client and server became available in the same package, and as such, getting one independently would be impractical.

Etymology

In mainstream English, the infinitive "to deprecate" means, simply, "to strongly disapprove of *(something)*". It derives from the Latin verb *deprecare*, meaning "to ward off (a *disaster*) by prayer". Thus, for a standard document to state that a feature is deprecated is merely a recommendation against using it.

Other usage

A particular term or expression may be *deprecated* when the term becomes obsolete, essentially superfluous and either has no meaning or serves no purpose and becomes essentially empty verbiage.

For example, in copyright, it was common to comply with the terms of the Buenos Aires Convention treaty by including a statement of rights such as *all rights reserved*, however, once every country that was a signatory to the Buenos Aires Convention also became a signatory to the Berne Convention treaty (because Berne does not require any form of notice), the use of the term "all rights reserved" has been essentially deprecated, as it no longer serves any purpose and grants no additional right or protection not already available if the statement were never used. In this instance, **deprecated** is a better term to describe the condition than **obsolete**.

An example in paleontology would be *Brontosaurus*, a synonym of the genus *Apatosaurus*.

Chapter 9

Collaboration

Collaboration is working together to achieve a goal, but in its negative sense it is working as a traitor. It is a recursive process where two or more people or organizations work together to realize shared goals, (this is more than the intersection of common goals seen in co-operative ventures, but a deep, collective, determination to reach an identical objective) — for example, an intruiging endeavor that is creative in nature—by sharing knowledge, learning and building consensus. Most collaboration requires leadership, although the form of leadership can be social within a decentralized and egalitarian group. In particular, teams that work collaboratively can obtain greater resources, recognition and reward when facing competition for finite resources. Collaboration is also present in opposing goals exhibiting the notion of adversarial collaboration, though this is not a common case for using the word.

Structured methods of collaboration encourage introspection of behavior and communication. These methods *specifically* aim to increase the success of teams as they engage in collaborative problem solving. Forms, rubrics, charts and graphs are useful in these situations to objectively document personal traits with the goal of improving performance in current and future projects.

Since the Second World War the term "Collaboration" acquired a very negative meaning as referring to persons and groups which help a foreign occupier of their country—due to actual use by people in European countries who worked with and for the Nazi German occupiers. Linguistically, "collaboration" implies more or less equal partners who work together—which is obviously not the case when one party is an army of occupation and the other are people of the occupied country living under the power of this army.

In order to make a distinction, the more specific term Collaborationism is often used for this phenomenon of collaboration with an occupying army. However, there is no water-tight distinction; "Collaboration" and "Collaborator", as well as "Collaborationism" and "Collaborationist", are often used in this pejorative sense—and even more so, the

equivalent terms in French and other languages spoken in countries which experienced direct Nazi occupation.

Classical examples of collaboration

Following are some examples of successful collaboration efforts in the past.

Trade

The trade of goods is an economic activity providing mutual benefit

Trade originated with the start of communication in prehistoric times. Trading was the main facility of prehistoric people, who bartered goods and services from each other when there was no such thing as the modern day currency. Peter Watson dates the history of long-distance commerce from circa 150,000 years ago. Trade exists for many reasons. Due to specialisation and division of labor, most people concentrate on a small aspect of production, trading for other products. Trade exists between regions because different regions have a comparative advantage in the production of some tradable commodity, or because different regions' size allows for the benefits of mass production. As such, trade at market prices between locations benefits both locations.

Community organization

Organization and cooperation between community members provides economic and social benefits

The members of an intentional community typically hold a common social, political or spiritual vision. They also share responsibilities and resources. Intentional communities include cohousing, residential land trusts, ecovillages, communes, kibbutzim, ashrams, and housing cooperatives. Typically, new members of an intentional community are selected by the community's existing membership, rather than by real-estate agents or land owners (if the land is not owned by the community).

Hutterite, Austria (16th century)
: Housing units are built and assigned to individual families but belong to the colony and there is very little personal property. Meals are taken by the entire colony in a common long room.

Oneida Community, Oneida, New York (1848)
: The Oneida Community practiced *Communalism* (in the sense of communal property and possessions) and *Mutual Criticism*, where every member of the community was subject to criticism by committee or the community as a whole, during a general meeting. The goal was to eliminate bad character traits.

Early Kibbutz settlements founded near Jerusalem (1890)
: A Kibbutz is an Israeli collective community. The movement combines socialism and Zionism in a form of practical Labor Zionism, founded at a time when independent farming was not practical or perhaps more correctly—not practicable. Forced by necessity into communal life, and inspired by their own ideology, the kibbutz members developed a pure communal mode of living that attracted interest from the entire world. While the kibbutzim lasted for several generations as utopian communities, most of today's kibbutzim are scarcely

different from the capitalist enterprises and regular towns to which the kibbutzim were originally supposed to be alternatives.

Game theory

The prisoner's dilemma is an example of game theory

Game theory is a branch of applied mathematics and economics that looks at situations where multiple players make decisions in an attempt to maximize their returns. The first documented discussion of it is a letter written by James Waldegrave, 1st Earl Waldegrave in 1713. Antoine Augustin Cournot's *Researches into the Mathematical Principles of the Theory of Wealth* in 1838 provided the first general theory. It was not until 1928 that this became a recognized, unique field when John von Neumann published a series of papers. Von Neumann's work in game theory culminated in the 1944 book The *Theory of Games and Economic Behavior* by von Neumann and Oskar Morgenstern. In 1950, the first discussion of the prisoner's dilemma appeared, and an experiment was undertaken on this game at the RAND corporation.

Military-industrial complex

The term **military-industrial complex** refers to a close and symbiotic relationship among a nation's armed forces, its private industry, and associated political and commercial interests. In such a system, the military is dependent on industry to supply material and other support, while the defense industry depends on government for revenue.

Skunk Works
: Skunk Works is a term used in engineering and technical fields to describe a group within an organization given a high degree of autonomy and unhampered by bureaucracy, tasked with working on advanced or secret projects. Founded at Lockheed Martin in 1943, the team developed highly innovative aircraft in short time frames, even beating its first deadline by 37 days. Creator of the organization, Kelly Johnson is said to have been an 'organizing genius' and had fourteen basic operating rules.

Manhattan Project

The Manhattan Project was the project to develop the first nuclear weapon (atomic bomb) during World War II by the United States, the United Kingdom and Canada. Formally designated as the Manhattan Engineer District, it refers specifically to the period of the project from 1941–1946 under the control of the U.S. Army Corps of Engineers, under the administration of General Leslie R. Groves. The scientific research was directed by American physicist J. Robert Oppenheimer.

While the aforementioned persons were influential in the project itself, the value of this project as an influence on organized collaboration is better attributed to Vannevar Bush. In early 1940, Bush lobbied for the creation of the National Defense Research Committee. Frustrated by previous bureaucratic failures in implementing technology in World War I, Bush sought to organize the scientific power of the United States for greater success.

The project succeeded in developing and detonating three nuclear weapons in 1945: a test detonation of a plutonium implosion bomb on July 16 (the Trinity test) near Alamogordo, New Mexico; an enriched uranium bomb code-named "Little Boy" on August 6 over Hiroshima, Japan; and a second plutonium bomb, code-named "Fat Man" on August 9 over Nagasaki, Japan.

Project management

The 2,751 Liberty ships built in four years by the United States during World War II required new approaches in organization and manufacturing

As a discipline, Project Management developed from different fields of application including construction, engineering, and defense. In the United States, the forefather of project management is Henry Gantt, called the father of planning and control techniques, who is famously known for his use of the "bar" chart as a project management tool, for

being an associate of Frederick Winslow Taylor's theories of scientific management, and for his study of the work and management of Navy ship building. His work is the forerunner to many modern project management tools including the work breakdown structure (WBS) and resource allocation.

The 1950s marked the beginning of the modern project management era. Again, in the United States, prior to the 1950s, projects were managed on an ad hoc basis using mostly Gantt charts, and informal techniques and tools. At that time, two mathematical project scheduling models were developed: (1) the "Program Evaluation and Review Technique" or PERT, developed as part of the United States Navy's (in conjunction with the Lockheed Corporation) Polaris missile submarine program; and (2) the "Critical Path Method" (CPM) developed in a joint venture by both DuPont Corporation and Remington Rand Corporation for managing plant maintenance projects. These mathematical techniques quickly spread into many private enterprises.

In 1969, the Project Management Institute (PMI) was formed to serve the interest of the project management industry. The premise of PMI is that the tools and techniques of project management are common even among the widespread application of projects from the software industry to the construction industry. In 1981, the PMI Board of Directors authorized the development of what has become *A Guide to the Project Management Body of Knowledge* (PMBOK), containing the standards and guidelines of practice that are widely used throughout the profession. The International Project Management Association (IPMA), founded in Europe in 1967, has undergone a similar development and instituted the IPMA Project Baseline. Both organizations are now participating in the development of a global project management standard.

Academia

Black Mountain College

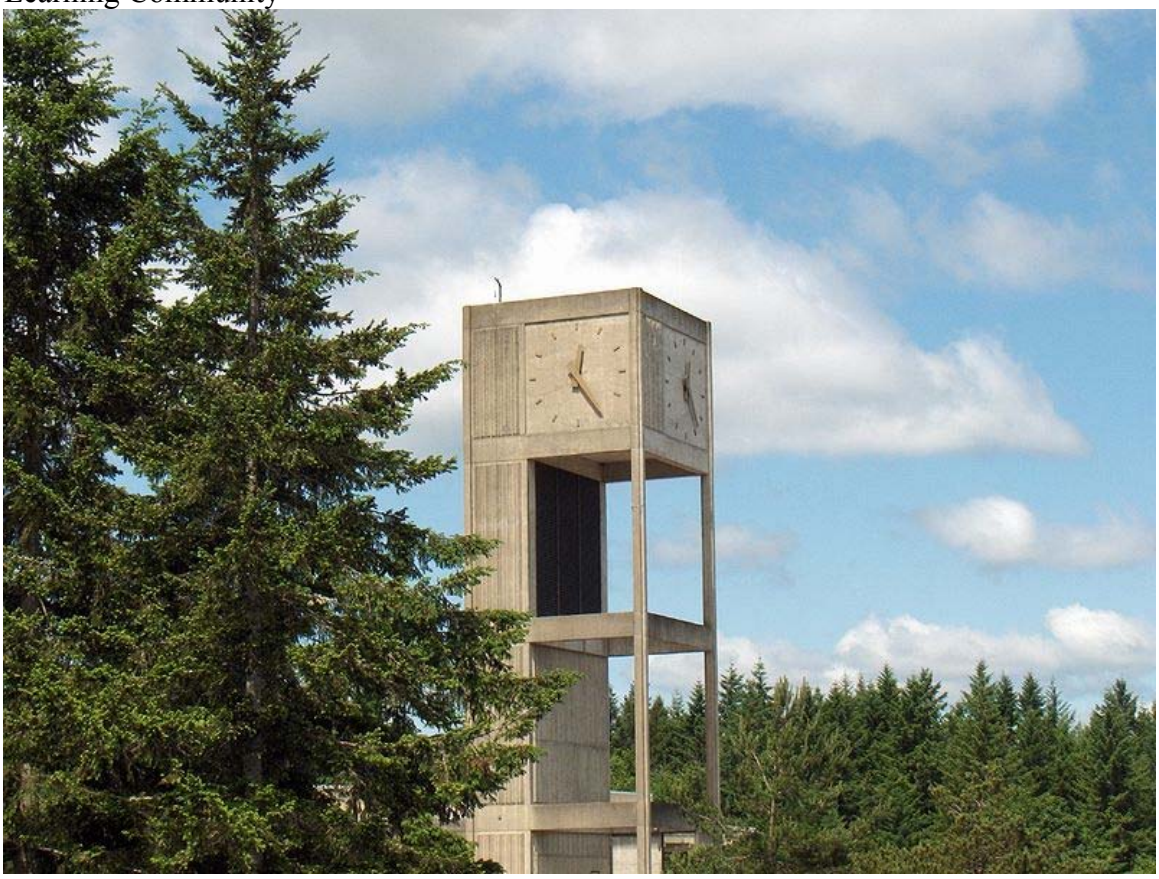
Founded in 1933 by John Andrew Rice, Theodore Dreier and other former faculty of Rollins College, Black Mountain was experimental by nature and committed to an interdisciplinary approach, attracting a faculty which included many of America's leading visual artists, poets, and designers.

Operating in a relatively isolated rural location with little budget, Black Mountain College inculcated an informal and collaborative spirit, and over its lifetime attracted a venerable roster of instructors. Some of the innovations, relationships and unexpected connections formed at Black Mountain would prove to have a lasting influence on the postwar American art scene, high culture, and eventually pop culture. Buckminster Fuller met student Kenneth Snelson at Black Mountain, and the result was the first geodesic dome (improvised out of slats in the school's back yard); Merce Cunningham formed his dance company; and John Cage staged his first happening.

Not a haphazardly conceived venture, Black Mountain College was a consciously directed liberal arts school that grew out of the progressive education movement. In its day it was a unique educational experiment for the artists and writers who conducted it, and as such an important incubator for the American avant garde. Black Mountain proved to be an important precursor to and prototype for many of

the alternative colleges of today ranging from the University of California, Santa Cruz to Hampshire College and Evergreen State College, among others.

Learning Community



The Evergreen signature clock tower

Dr. Wolff-Michael Roth and Stuart Lee of the University of Victoria assert that until the early 1990s the individual was the 'unit of instruction' and the focus of research. The two observed that researchers and practitioners switched to the idea that knowing is 'better' thought of as a cultural practice. Roth and Lee also claim that this led to changes in learning and teaching design in which students were encouraged to share their ways of doing mathematics, history, science, with each other. In other words, that children take part in the construction of consensual domains, and 'participate in the negotiation and institutionalisation of … meaning'. In effect, they are participating in **learning communities.**

This analysis does not take account of the appearance of Learning communities in the United States in the early 1980s. For example, The Evergreen State College, which is widely considered a pioneer in this area, established an intercollegiate learning community in 1984. In 1985, this same college established the Washington Center for Improving the Quality of Undergraduate Education, which focuses on collaborative education approaches, including learning communities as one of its centerpieces.

Classical music

Although relatively rare compared with collaboration in popular music, there have been some notable examples of music written in collaboration between classical composers. Perhaps the best-known examples are:

- *Hexameron*, a set of variations for solo piano on a theme from Vincenzo Bellini's opera *I puritani*. It was written and first performed in 1837. The contributors were Franz Liszt, Frédéric Chopin, Carl Czerny, Sigismond Thalberg, Johann Peter Pixis, and Henri Herz.
- The *F-A-E Sonata*, a sonata for violin and piano, written in 1853 as a gift for the violinist Joseph Joachim. The composers were Albert Dietrich (first movement), Robert Schumann (second and fourth movements), and Johannes Brahms (third movement).

Contemporary examples

Arts

Collaboration—or joint production by two or more artists—is a common style among musicians and performance artists. It has not been so popular, on the other hand, in the world of art, and especially in modern art. But the strong sense of individualism long possessed by artists of fine art began to wane around the 1960s, and some artists working in units have emerged and become widely known along with the development of new media based on the advances in information technology. They have changed the concept of art into something that can be engaged in by more than individual artists alone.

Art groups

Fluxus

> An international network of artists, composers and designers noted for blending different artistic media and disciplines in the 1960s. Fluxus encouraged a do it yourself aesthetic, and valued simplicity over complexity. Like Dada before it, Fluxus included a strong current of anti-commercialism and an anti-art sensibility, disparaging the conventional market-driven art world in favor of an artist-centered creative practice. As Fluxus artist Robert Filliou wrote, however, Fluxus differed from Dada in its richer set of aspirations, and the positive social and communitarian aspirations of Fluxus far outweighed the anti-art tendency that also marked the group.

Just Buffalo Literary Center, CEPA Gallery, and Big Orbit are three nonprofit arts organizations in Buffalo, New York, that have shared space and certain administrative functions since 2005. Just Buffalo offers an array of literary arts and arts-in-education programs. CEPA Gallery presents contemporary photo-related art and supports working artists. And Big Orbit has an art gallery and programs in the fields of experimental theater, literary performance, new music and sound art.

Once they co-located their administrative offices they quickly started to realize a number of advantages. Financial savings was an obvious one (they share equipment, a software contract, phone and Internet services and more). Physical proximity also helped the three executives develop a strong sense of trust and respect, and they soon looked for other ways to collaborate, such as hiring a shared grant writer who brings in grants for all three organizations.

There have been many benefits: financial savings because of their shared space, increased donations, and improved artistic programming. Beyond the tangible benefits, there are important intangibles. The agency directors share information and ideas, and they coordinate mailings. Perhaps most important, the organizations have increased their creativity; being in the same space has led to a "think tank" atmosphere. One of the three directors notes that "We work so closely … it's helped us come up with new thinking to expand our capacity and create a built-in brain trust and support system for problem solving and practical help."

Situationist International

> The **Situationist International (SI)** was a small group of international political and artistic agitators with roots in Marxism, Lettrism and the early 20th century European artistic and political avant-gardes. Formed in 1957, the SI was active in Europe through the 1960s and aspired to major social and political transformations. In the 1960s it split into a number of different groups, including the Situationist Bauhaus, the Antinational and the Second Situationist International. The first SI disbanded in 1972.

Business

Training meeting about sustainable design. The photo shows a training meeting with factory workers in a stainless steel ecodesign company from Rio de Janeiro, Brazil. These types of meetings are important for the collaboration of the team

Collaboration in business can be found both inter- and intra-organization and ranges from the simplicity of a partnership and crowd funding to the complexity of a multinational corporation. Collaboration between team members allows for better communication within the organization and throughout the supply chains. It is a way of coordinating different ideas from numerous people to generate a wide variety of knowledge. The recent improvement in technology has provided the world with high speed internet, wireless connection, and web-based collaboration tools like blogs and has as such created a "mass collaboration." People from all over the world are efficiently able to communicate and share ideas through the internet, or even conferences, without any geographical barriers.

Education

Generally defined, an Educational Collaborative Partnership is ongoing involvement between schools and business/industry, unions, governments and community organizations. Educational Collaborative Partnerships are established by mutual agreement between two or more parties to work together on projects and activities that will enhance the quality of education for students while improving skills critical to success in the workplace.

Collaboration in Education- two or more co-equal individual voluntarily brings their knowledge and experience together by interacting toward a common goal in the best interest of students for the betterment of their education success.

Music

Musical collaboration occurs when musicians in different places or groups work on the same album or song. Collaboration between musicians, especially with regards to jazz, is often heralded as the epitome of complex collaborative practice. Special software has been written to facilitate musical collaboration over the Internet. Websites have also been created to enable creative music collaboration over the Internet.

Several awards exist specifically for collaboration in music:

- Grammy Award for Best Country Collaboration with Vocals—awarded since 1988
- Grammy Award for Best Pop Collaboration with Vocals—awarded since 1995
- Grammy Award for Best Rap/Sung Collaboration—awarded since 2002

Entertainment

Collaboration in entertainment is a relatively new phenomenon brought on with the advent of social media, reality tv, and video sharing sites such as YouTube and Vimeo. Collaboration occurs when writers, directors, actors, producers and other individuals or groups work on the same television show, short film, or feature length film. A revolutionary system has been developed by Will Wright for the production of the TV series title Bar Karma on CurrentTV. Special web-based software, titled Storymaker, has been written to facilitate plot collaboration over the Internet. Organizations such as Orange_County_Screenwriters_Association bring together professional and amateur writers and filmmakers in a collaborative manner for entertainment development.

Publishing

Collaboration in publishing can be as simple as dual-authorship or as complex as commons-based peer production. Technological examples include Usenet, e-mail lists and blogs while 'brick and mortar' examples include monographs (books) and periodicals such as newspapers, journals and magazines.

Science

Though there is no political institution organizing the sciences on an international level, a self-organized, global network had formed in the late 20th century. Observed by the rise in co-authorships in published papers, Wagner and Leydesdorff found international collaborations to have doubled from 1990 to 2005. While collaborative authorships within nations has also risen, this has done so at a slower rate and is not cited as frequently.

Medicine

In medicine the physician assistant - physician relationship involves a collaborative plan to be on file with each state board of medicine where the PA works. This plan formally delineates the scope of practice approved by the physician.

Technology

Due to the complexity of today's business environment, collaboration in technology encompasses a broad range of tools that enable groups of people to work together including social networking, instant messaging, team spaces, web sharing, audio conferencing, video, and telephony. Broadly defined, any technology that facilitates linking of two or more humans to work together can be considered a collaborative tool. Blogs, even Twitter are collaborative tools. Many large companies are developing enterprise collaboration strategies and standardizing on a collaboration platform to allow their employees, customers and partners to intelligently connect and interact.

Enterprise collaboration tools are centred around attaining collective intelligence and staff collaboration at the organisation level, or with partners. These include features such as staff networking, expert recommendations, information sharing, expertise location, peer feedback, and real-time collaboration. At the personal level, this enables employees to enhance social awareness and their profiles and interactions Collaboration encompasses both asynchronous and synchronous methods of communication and serves as an umbrella term for a wide variety of software packages. Perhaps the most commonly associated form of synchronous collaboration is web conferencing using tools such as Cisco TelePresence, Cisco WebEx Meetings, HP Halo Telepresence Solutions or Microsoft Live Meeting, but the term can easily be applied to IP telephony, instant messaging, and rich video interaction with telepresence, as well. Examples of asynchronous collaboration software include Cisco WebEx Connect, Microsoft Sharepoint.

The effectiveness of a collabortative effort is driven by three critical factors: - Communication - Content Management - Workflow control

The Internet

> The low cost and nearly instantaneous sharing of ideas, knowledge, and skills has made collaborative work dramatically easier. Not only can a group cheaply communicate and test, but the wide reach of the Internet allows such groups to easily form in the first place, even among niche interests. An example of this is the free software movement in software development which produced GNU and Linux from scratch and has taken over development of Mozilla and OpenOffice.org (formerly known as Netscape Communicator and StarOffice).

Commons-based peer production

> Commons-based peer production is a term coined by Yale's Law professor Yochai Benkler to describe a new model of economic production in which the creative energy of large numbers of people is coordinated (usually with the aid of the

internet) into large, meaningful projects, mostly without traditional hierarchical organization or financial compensation. He compares this to firm production (where a centralized decision process decides what has to be done and by whom) and market-based production (when tagging different prices to different jobs serves as an attractor to anyone interested in doing the job).

Examples of products created by means of commons-based peer production include Linux, a computer operating system; Slashdot, a news and announcements website; Kuro5hin, a discussion site for technology and culture; and Clickworkers, a collaborative scientific work. Another example is Socialtext which is a software that uses tools such as weblogs and helps companies to create a collaborative work environment.

Massively distributed collaboration

The term massively distributed collaboration was coined by Mitchell Kapor, in a presentation at UC Berkeley on 2005-11-09, to describe an emerging activity of electronic mailing lists and blogs and other content-creating virtual communities online.

Chapter 10

Universal Data Element Framework

The **Universal Data Element Framework** (**UDEF**) provides the foundation for building an enterprise-wide controlled vocabulary. It is a standard way of indexing enterprise information that can produce big cost savings. UDEF simplifies information management through consistent classification and assignment of a global standard identifier to the data names and then relating them to similar data element concepts defined by other organizations. Though this approach is a small part of the overall picture, it is potentially a crucial enabler of semantic interoperability.

How UDEF works

UDEF provides semantic links, through assigning an intelligent, derived ID as an attribute of the data element, essentially labeling the element as a specific data element concept. When this UDEF ID exists in both source and target formats, it can then be used as an easy analysis point via a match report, and then as the primary pivot point for transformations between source and target.

UDEF takes a list of high-level root object classes and assigns an integer to each class plus alpha characters to each specialization modifier. It then also assigns integers to property word plus integers to each specialization modifier. These object class alpha-integers are concatenated together with the property integers to form a dewey-decimal like code for each data element concept.

Assuming an application used by a hospital needs to map the data element concepts to the UDEF, the last name and first name (within UDEF you will find Family Name and Given Name under the UDEF property Name) of several people that are likely to appear on a medical record that could include the following example data element concepts –

Patient Person Family Name – find the word “Patient” under the UDEF object “Person” and find the word “Family” under the UDEF property “Name”

Patient Person Given Name – find the word “Patient” under the UDEF object “Person” and find the word “Given” under the UDEF property “Name”

Doctor Person Family Name – find the word “Doctor” under the UDEF object “Person” and find the word “Family” under the UDEF property “Name”

Doctor Person Given Name – find the word “Doctor” under the UDEF object “Person” and find the word “Given” under the UDEF property “Name”

The associated UDEF IDs for the above are derived by walking up each tree respectively and using an underscore to separate the object from the property. For the examples above, the following data element concepts are available within the current UDEF –

“Patient Person Family Name” the UDEF ID is “au.5_11.10”

“Patient Person Given Name” the UDEF ID is “au.5_12.10”

“Doctor Person Family Name” the UDEF ID is “aq.5_11.10”

“Doctor Person Given Name” the UDEF ID is “aq.5_12.10”

Six Basic Steps to Map Enterprise Data to the UDEF

There are six basic steps to follow when mapping data element concepts to the UDEF.

1. Identify the applicable UDEF property word that characterizes the dominant attribute (property) of the data element concept. For example: Name, Identifier, Date, etc.

2. Identify the dominant UDEF object word that the dominant property (selected in step 1) is describing. For example, Person_Name, Product_Identifier, Document_Date, etc.

3. By reviewing the UDEF tree for the selected property identified in step 1, identify applicable qualifiers that are necessary to describe the property word term unambiguously. For example, Family Name.

4. By reviewing the UDEF tree for the selected object identified in step 2, identify applicable qualifiers that are necessary to describe the object word term unambiguously. For example, Customer Person.

5. Concatenate the object term and the property term to create a UDEF naming convention compliant name where it is recognized that the name may seem artificially long. For example, Customer Person_Family Name.

6. Derive a structured ID based on the UDEF taxonomy that carries the UDEF inherited indexing scheme. For example <CustomerPersonFamilyName UDEFID=”as.5_11.10”>.

The Open Group UDEF Project objectives

The UDEF Project aims to establish the Universal Data Element Framework (UDEF) as the universally-used classification system for data element concepts. It focuses on developing and maintaining the UDEF as an open standard, advocating and promoting it, putting in place a technical infrastructure to support it, implementing a Registry for it, and setting up education programs to train information professionals in its use.

Organizations that implement UDEF will likely realize the greatest benefit by defining their controlled vocabulary based on the UDEF. To help an organization manage its UDEF based controlled vocabulary, it should seriously consider a metadata registry that is based on ISO/IEC 11179-5.

History of UDEF

Ron Schuldt, Sr. Enterprise Data Architect, Lockheed Martin, originated the UDEF concept based on ISO/IEC 11179 Metadata standards in the early 1990's. Currently, he is a Senior Partner with UDEF-IT, LLC.

Ownership of UDEF intellectual property

The Open Group assumed from AFEI the right to grant public use licensing of the UDEF.

The Supplier Management Council Electronic Enterprise Working Group of the Aerospace Industry Association (AIA) supports the UDEF as the naming convention solution to XML interoperability between standards that include all functions throughout a product's life-cycle and is working through a well defined process to obtain approval of this position from AIA and its member companies.

Criticism

Classification in UDEF is not sometimes hampered by ad-hoc decisions that might produce problems. Example:

- b.be.5 is "United-Kingdom Citizen Person" and
- c.be.5 is "European Union Citizen Person"

As the United Kingdom is part of the European Union, the classification is not unique.

Some of the concepts in UDEF are not as universal as it is claimed. They show a lot of bias to anglo-american tradition and way of thinking and are not easily transferable to other languages. Example: The following part of the hierarchy shows the concept of an officer.

- j.5 Officer.Person
- a.j.5 Contracting.Officer.Person
- a.a.j.5 Procuring.Contracting.Officer.Person
- a.a.a.j.5 Government.Procuring.Contracting.Officer.Person
- b.a.j.5 Administrative.Contracting.Officer.Person
- b.j.5 Police.Officer.Person
- c.j.5 Military.Officer.Person

In many cultures, the part of the tree below "a.j.5 Contracting Officer Person" would not be placed under j.5 as b.j.5 or c.j.5.

Chapter 11

Enterprise Architecture

An **enterprise architecture (EA)** is a rigorous description of the structure of an enterprise, which comprises enterprise components (business entities), the externally visible properties of those components, and the relationships (e.g. the behavior) between them. EA describes the terminology, the composition of enterprise components, and their relationships with the external environment, and the guiding principles for the requirement (analysis), design, and evolution of an enterprise. This description is comprehensive, including enterprise goals, business process, roles, organizational structures, organizational behaviors, business information, software applications and computer systems.

Practitioners of EA call themselves "enterprise architects." An enterprise architect is a person responsible for developing the enterprise architecture and is often called upon to draw conclusions from it. By producing an enterprise architecture, architects are providing a tool for identifying opportunities to improve the enterprise, in a manner that more effectively and efficiently pursues its purpose.

The scope of an enterprise architecture

The term "enterprise" is used because it is generally applicable in many circumstances, including

- Public or Private Sector organizations
- An entire business or corporation
- A part of a larger enterprise (such as a business unit)
- A conglomerate of several organizations, such as a joint venture or partnership
- A multiply-outsourced business operation

The term enterprise includes the whole complex, socio-technical system, including:

- people
- information
- technology

Defining the boundary or scope of the enterprise to be described is an important first step in creating the enterprise architecture. It should also be noted that the term "enterprise" as used in enterprise architecture generally means more than the information systems employed by an organization.

Methods and frameworks

Enterprise architects use various business methods, analytical techniques and conceptual tools to understand and document the structure and dynamics of an enterprise. In doing so, they produce lists, drawings, documents and models, together called "artifacts". These artifacts describe the logical organization of business functions, business capabilities, business processes, people organization, information resources, business systems, software applications, computing capabilities, information exchange and communications infrastructure within the enterprise.

A collection of these artifacts, sufficiently complete to describe the enterprise in useful ways, is considered by EA practitioners an 'enterprise' level architectural description, or enterprise architecture, for short. The UK National Computing Centre EA best practice guidance states

Normally an EA takes the form of a comprehensive set of cohesive models that describe the structure and functions of an enterprise.

and continues

The individual models in an EA are arranged in a logical manner that provides an ever-increasing level of detail about the enterprise: its objectives and goals; its processes and organisation; its systems and data; the technology used and any other relevant spheres of interest.

This is the definition of enterprise architecture implicit in several EA frameworks including the popular TOGAF architectural framework.

An enterprise architecture framework bundles tools, techniques, artifact descriptions, process models, reference models and guidance used by architects in the production of enterprise-specific architectural description.

In 1992, Steven Spewak described a process for creating an enterprise architecture that is widely used in educational courses.

Areas of practice

Several enterprise architecture frameworks break down the practice of enterprise architecture into a number of practice areas or "domains". In his book on Enterprise Architecture, Spewak divides the practice into two domains at 'level 2': "Business Modelling" and "Current Systems and Technology" and three subordinate domains at 'level 3': "Data Architecture", "Applications Architecture" and "Technology Architecture". The final level of Spewak's EAP is the "Implementation" or "Methods" level, which deals with "how" to migrate the Enterprise to match the new model.

The popular TOGAF framework divides the practice into three domains: "Business Architecture", "Information Systems Architecture" and "Technology Architecture" and then subdivides the information systems architecture into "Information Architecture and "Applications Architecture".

The Strategic Architecture Model allows for a flexible division into up to ten domains covering many aspects of an enterprise from its objectives and goals through its projects and programmes to its software applications and technology.

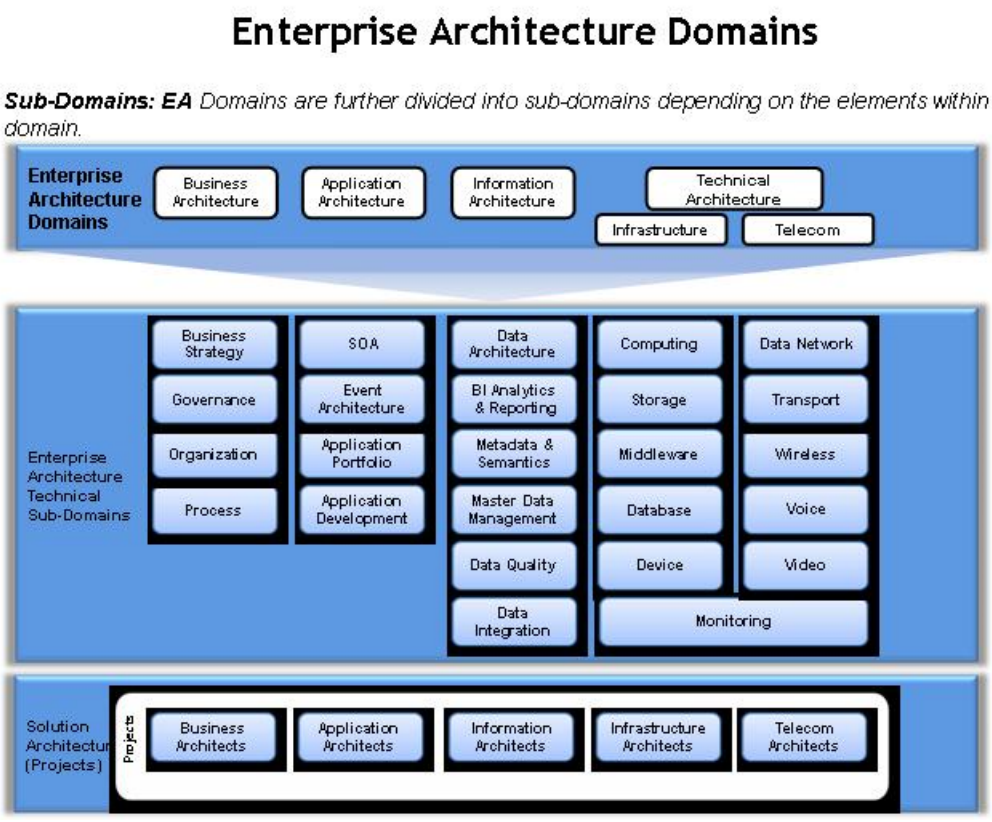
EA Domains: An enterprise architecture’s landscape is usually divided into various domains based on the attributes of the environment and the logical grouping based on Industry EA Frameworks

The dividing of the practice into a number of domains allows enterprise architects to describe an enterprise from a number of important perspectives. This practice also encourages the contributions of many individuals and allows the practice as a whole to make good use of individual domain-specific expertise and knowledge. By taking this approach, enterprise architects can ensure a holistic description is produced.

The popular and most common four domains and their component parts look like this:

1. Business:
    1. Strategy maps, goals, corporate policies, Operating Model
    2. Functional decompositions (e.g. IDEF0, SADT), business capabilities and organizational models expressed as enterprise / line of business architecture
    3. Business processes, Workflow and Rules that articulate the assigned authorities, responsibilities and policies
    4. Organization cycles, periods and timing
    5. Suppliers of hardware, software, and services
2. Information:
    1. Information architecture - a holistic view on the flow of information in an enterprise
    2. Data Architecture- describes the way data will be processed, stored , data flows and used by the projects teams that will use it

3. Master Data Management, is the authoritative, reliable foundation for data used across many applications and business processes with the goal to provide a single view of the truth no matter where the data is located
4. Metadata - data that describes your enterprise data elements
5. Business Intelligence Analytics & Reporting BI (Business Intelligence) is a broad category of applications and technologies for gathering, storing, analyzing, and providing access to data to help the organization users make better business decisions. These include the activities of decision support systems, query and reporting, dashboards , scorecards ,statistical analysis, forecasting, and data mining. This includes Reporting Data Stores (Operational Data Store (ODS), Datamart and DataWarehouses)
6. Data Quality helps identify, analyze, improve, and measure the data quality and data integrity issues and improvement efforts
7. Data models: conceptual expressed as enterprise information architectures, logical, and physical
8. Data Life Cycle Management Processes to govern how to create, classify, update, use, distribute, and archive, and obsolete data and information



Enterprise Architecture Domains Subdomains

3. Applications:

    1. Application software inventories and diagrams, expressed as conceptual / functional or system enterprise / line of business architectures
    2. Interfaces between applications - that is: events, messages
4. Technology:
    1. Inter-application mediating software or 'middleware'.
    2. Application execution environments and operating frameworks including applications server environments and operating systems, authentication and authorisation environments, security systems and operating and monitoring systems.
    3. Hardware, platforms, and hosting: servers, datacentres and computer rooms
    4. Local and wide area networks, Internet connectivity diagrams
    5. Intranet, Extranet, Internet, eCommerce, EDI links with parties within and outside of the organization
    6. Operating System
    7. Infrastructure software: Application servers, DBMS
    8. Programming Languages, etc. expressed as enterprise / line of business technology architecture.

Using an enterprise architecture

Describing the architecture of an enterprise aims primarily to improve the effectiveness or efficiency of the business itself. This includes innovations in the structure of an organization, the centralization or federation of business processes, the quality and timeliness of business information, or ensuring that money spent on information technology (IT) can be justified.

One method of using this information to improve the functioning of a business, as described in the TOGAF architectural framework, involves developing an "architectural vision": a description of the business that represents a "target" or "future state" goal. Once this vision is well understood, a set of intermediate steps are created that illustrate the process of changing from the present situation to the target. These intermediate steps are called "transitional architectures" by TOGAF. Similar methods have been described in other enterprise architecture frameworks.

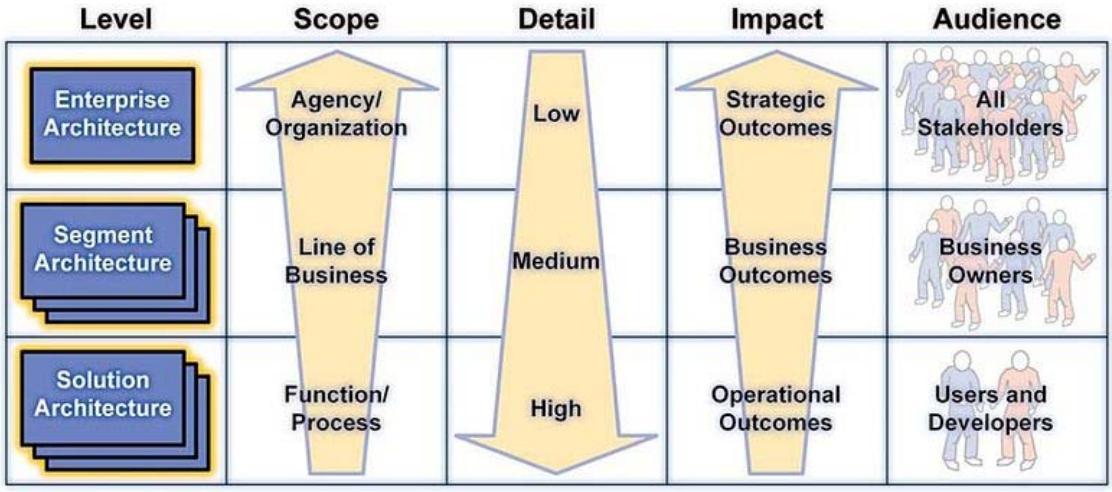
The growing use of enterprise architecture

Documenting the architecture of enterprises is done within the U.S. Federal Government in the context of the Capital Planning and Investment Control (CPIC) process. The Federal Enterprise Architecture (FEA) reference models serve as a framework to guide Federal agencies in the development of their architectures. Companies such as Independence Blue Cross, Intel, Volkswagen AG and InterContinental Hotels Group have also applied enterprise architecture to improve their business architectures as well as to improve business performance and productivity.

Relationship to other disciplines

Enterprise architecture is a key component of the information technology governance process in organizations such Dubai Customs and AGL Energy. Organizations like Dubai Customs and AGL Energy have implemented a formal enterprise architecture process as part of their IT management strategy. While this may imply that enterprise architecture is closely tied to IT, this should be viewed in the broader context of business optimization in that it addresses business architecture, performance management and process architecture as well as more technical subjects. Depending on the organization, enterprise architecture teams may also be responsible for some aspects of performance engineering, IT portfolio management and metadata management. Recently, protagonists like Gartner and Forrester have stressed the important relationship of Enterprise Architecture with emerging holistic design practices such as Design Thinking and User Experience Design.

The following image from the 2006 FEA Practice Guidance of US OMB sheds light on the relationship between enterprise architecture and segment(BPR) or Solution architectures. (This figure demonstrates that software architecture is truly a solution architecture discipline, for example.)



Activities such as software architecture, network architecture, database architecture are partial contributions to a solution architecture.

Published examples of enterprise architecture

It is uncommon for a commercial organization to publish rich detail from their enterprise architecture descriptions. Doing so can provide competitors information on weaknesses and organizational flaws that could hinder the company's market position. However, many government agencies around the world have begun to publish the architectural descriptions that they have developed. Good examples include the US Department of the Interior, and the US Department of Defense business transformation agency.

Chapter 12

Enterprise Engineering

Enterprise engineering, is a subdiscipline of systems engineering in that it applies the knowledge and methods of systems engineering to the design of enterprises. It considers the design of the business, which encompasses the business, information, and organization".

Overview

In theory and practice two types of enterprise engineering have emerged. A more general connected to engineering and the management of enterprises, and a more specific related to system engineering, software engineering, enterprise modeling, and enterprise architecture.

In the field of engineering a more general enterprise engineering emerged, defined as "the application of engineering principles to the management of enterprises". This field "encompasses the application of knowledge, principles, and disciplines related to the analysis, design, implementation and operation of all elements associated with an enterprise. In essence this is an interdisciplinary field which combines systems engineering and strategic management as it seeks to engineer the entire enterprise in terms of the products, processes and business operations. The view is one of continuous improvement and continued adaptation as firms, processes and markets develop along their life cycles. This total systems approach encompasses the traditional areas of research and development, product design, operations and manufacturing as well as information systems and strategic management". This field is related to engineering management, operations management, service management and systems engineering.

In the context of software development a specific field of enterprise engineering has emerged, which "deals with the modelling and integration of various organizational and technical parts of business processes". In the context of information systems development it has been the area of activity in the organization of the systems analysis, and an

extension of the scope of Information Modelling. It can also be viewed as the extension and generalization of the systems analysis and systems design phases of the software development process. Here Enterprise modelling can be part of the early, middle and late information system development life cycle. Explicit representation of the organizational and technical system infrastructure is being created in order to understand the orderly transformations of existing work practices. This field is also called Enterprise architecture, or defined with Enterprise Ontology as being two major parts of Enterprise architecture.

Enterprise engineering methods

Within the area of enterprise engineering formal methodologies, methods and techniques are designed, tested and extensively used in order to offer organizations reusable business process solutions:

- Computer Integrated Manufacturing Open Systems Architecture (CIMOSA) methodology
- Integrated DEFinition (IDEF) methodology
- Petri Nets
- Unified Modeling Language (UML) or Unified Enterprise Modeling Language (UEML)
- Enterprise Function Diagrams (EFD)

These methodologies/techniques and methods are all more or less suited in modeling the enterprise and its underlying processes.

Computer Integrated Manufacturing Open Systems Architecture

CIMOSA provides templates and interconnected modeling constructs to encode business, people and IT aspects of enterprise requirements. This is done from multiple perspectives: Information view, Function view, Resource view and Organization view. These constructs can further be used to structure and facilitate the design and implementation of detailed IT systems.

The division in different views makes it a clarifying reference for enterprise and software engineers. It shows information needs for different enterprise functionalities (activities, processes, operations) and corresponding resources. In this way it can easily be determined which IT-system will fulfill the information needs in a certain activity and process.

Integrated DEFinition

IDEF is a modeling language, which was first developed for the modeling of manufacturing systems. It was already being used by the U.S. Airforce in 1981. Initially it had 4 different notations to model an enterprise from a certain viewpoint. These were IDEF0, IDEF1, IDEF2 and IDEF3 for functional, data, dynamic and process analysis

respectively. In the past decades a number of tools and techniques for the integration of the notations are developed in an incremental way.

IDEF clearly shows how a business process flows through a variety of decomposed business functions with corresponding information inputs, outputs and actors. Like CIMOSA, it also uses different enterprise views. Moreover, IDEF can be easily transformed into UML-diagrams for the further development of IT systems. These positive characteristics make it a powerful method for the development of Functional Software Architectures.

Petri Nets

Petri Nets are known tools to model manufacturing systems. They are highly expressive and provide good formalisms for the modeling of concurrent systems. The most advantageous properties are that of simple representation of states, concurrent system transitions and capabilities to model the duration of transitions.

Petri Nets therefore can be used to model certain business processes with corresponding state and transitions or activities within and outputs. Moreover, Petri Nets can be used to model different software systems and transitions between these systems. In this way programmers use it as a schematic coding reference.

In recent years a number of attempts have shown that Petri Nets can contribute to the development of business process integration. One of these is the Model Blue methodology, which is developed by IBM Chinese Research Laboratory and outlines the importance of model driven business integration as an emerging approach for building integrated platforms. A mapping between their Model Blue business view and an equivalent Petri Net is also shown, which indicates that their research closes the gap between business and IT. However, instead of Petri Nets they rather use their own Model Blue IT view, which can be derived from their business view through a transformation engine.

Unified Modeling Language

UML is a broadly accepted modeling language for the development of software systems and applications. The, so called, “object oriented community” also tries to use UML for enterprise modeling purposes. They emphasize the use of enterprise objects or business objects from which complex enterprise systems are made. A collection of these objects and corresponding interactions between them can represent a complex business system or process. Where Petri Nets focus on the interaction and states of objects, UML focuses more on the business objects themselves. Sometimes these are called the “enterprise building blocks”, which includes resources, processes, goals, rules and metamodels. Despite the fact that UML in this way can be used to model an integrated software system it has been argued that the reality of business can be modeled with a software modeling language. In reaction the object oriented community makes business extensions for UML and adapts the language. UEML is derived from UML and is proposed as a business

modeling language. The question remains if this business transformation is the right thing to do. It was earlier said that UML in combination with other “pure’ business methods can be a better alternative.

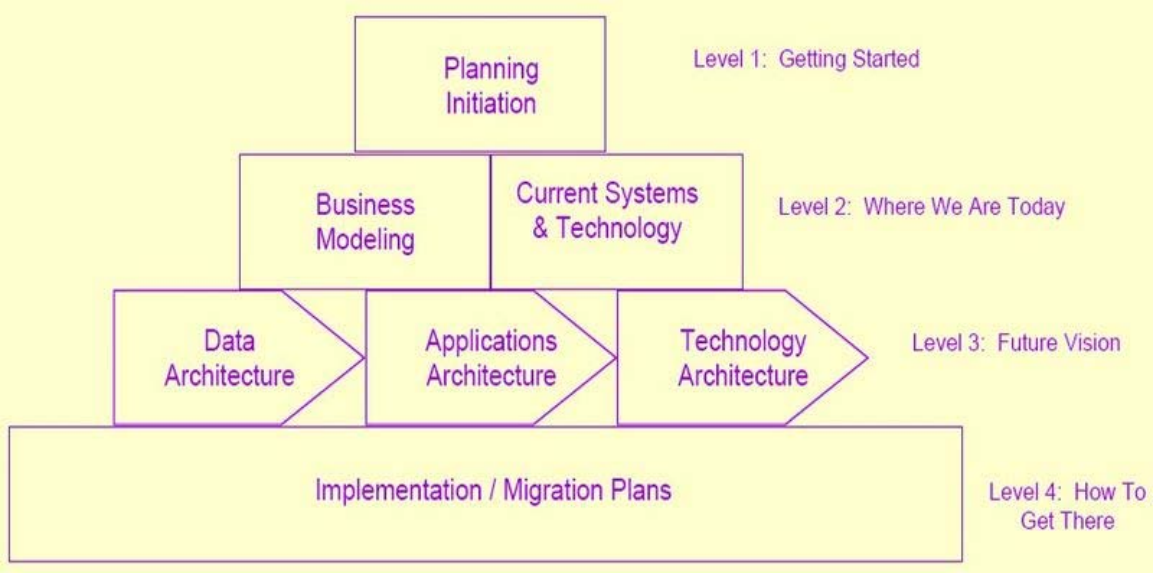
Enterprise Function Diagrams

EFD is a used modeling technique for the representation of enterprise functions and corresponding interactions. Different business processes can be modeled in these representations through the use of “function modules” and triggers. A starting business process delivers different inputs to different functions. A process flowing through all the functions and sub-functions creates multiple outputs. Enterprise Function Diagrams hereby give a very easy-to-use and detailed representation about a business process and corresponding functions, inputs, outputs and triggers. In this way EFD has many similarities with IDEF0 diagrams, which also represent in a hierarchical way business processes as a combination of functions and triggers. Difference is that an EFD places the business functions in an organization hierarchical perspective, which outlines the downstream of certain processes in the organization. On the contrary, IDEF0 diagrams show responsibilities of certain business functions through the use of arrows. Also, IDEF0 has a clear representation of inputs and outputs of every (sub)function.

EFD possibly could be used as a business front-end to a software modeling language like UML. The major resemblance with IDEF as a modeling tool indicates that it can be done. However, more research is needed to improve the EFD technique in such a way that formal mappings to UML can be made. about the complementary use of IDEF and UML has contributed to the acceptance of IDEF as business-front end. A similar study should be done with EFD and UML.

Chapter 13

Enterprise Architecture Planning



Levels of Enterprise Architecture Planning.

Enterprise Architecture Planning (EAP) in Enterprise Architecture is the planning process of defining architectures for the use of information in support of the business and the plan for implementing those architectures.

Overview

One of the earlier professional practitioners in the field of system architecture Steven H. Spewak in 1998 defined Enterprise Architecture Planning (EAP) as "the process of defining architectures for the use of information in support of the business and the plan for implementing those architectures." Spewak's approach to EAP is similar to that taken by DOE in that the business mission is the primary driver. That is followed by the data

required to satisfy the mission, followed by the applications that are built using that data, and finally by the technology to implement the applications.

This hierarchy of activity is represented in the figure above, in which the layers are implemented in order, from top to bottom. Based on the Business Systems Planning (BSP) approach developed by John Zachman, EAP takes a data-centric approach to architecture planning to provide data quality, access to data, adaptability to changing requirements, data interoperability and sharing, and cost containment. This view counters the more traditional view that applications should be defined before data needs are determined or provided for.

EAP topics

Zachman framework

EAP defines the blueprint for subsequent design and implementation and it places the planning/defining stages into a framework. It does not explain how to define the top two rows of the Zachman Framework in detail but for the sake of the planning exercise, abbreviates the analysis. The Zachman Framework provides the broad context for the description of the architecture layers, while EAP focuses on planning and managing the process of establishing the business alignment of the architectures.

EAP is planning that focuses on the development of matrixes for comparing and analyzing data, applications, and technology. Most important, EAP produces an implementation plan. Within the Federal Enterprise Architecture, EAP will be completed segment enterprise by segment enterprise. The results of these efforts may be of Governmentwide value; therefore, as each segment completes EAP, the results will be published on the ArchitecturePlus web site.

EAP components

Enterprise Architecture Planning model consists of four levels:

- Layer 1 - *getting started* : This layer leads to producing an EAP workplan and stresses the necessity of high-level management commitment to support and resource the subsequent six components (or steps) of the process. It consists of Planning Initiation, which covers in general, decisions on which methodology to use, who should be involved, what other support is required, and what toolset will be used.

- Layer 2 - *where we are today* : This layer provides a baseline for defining the eventual architecture and the long-range migration plan. It consists of:

    - Business process modeling, the compilation of a knowledge base about the business functions and the information used in conducting and supporting the various business processes, and

    - Current Systems and Technology, the definition of current application systems and supporting technology platforms.

- Layer 3 - *the vision of where we want to be* : The arrows delineate the basic definition process flow: data architecture, applications architecture, and technology architecture. It consists of:

    - Data Architecture - Definition of the major kinds of data needed to support the business.
    - Applications Architecture - Definition of the major kinds of applications needed to manage that data and support the business functions.
    - Technology Architecture - Definition of the technology platforms needed to support the applications that manage the data and support the business functions.

- Layer 4 - *how we plan to get there* : This consists of the Implementation / Migration Plans - Definition of the sequence for implementing applications, a schedule for implementation, a cost/benefit analysis, and a clear path for migration.

EAP methodology

The Enterprise Architecture Planning (EAP) methodology is beneficial to understanding the further definition of the Federal Enterprise Architecture Framework at level IV. EAP is a how to approach for creating the top two rows of the Zachman Framework, Planner and Owner. The design of systems begins in the third row, outside the scope of EAP.

EAP focuses on defining what data, applications, and technology architectures are appropriate for and support the overall enterprise. Exhibit 6 shows the seven components (or steps) of EAP for defining these architectures and the related migration plan. The seven components are in the shape of a wedding cake, with each layer representing a different focus of each major task (or step).

Applications

Spewak approach to Federal Enterprise Architecture has helped organizations with modeling, business strategy planning, process improvement, data warehousing, and various support systems designs, data administration standards, object-oriented and information engineering methodologies, and project management.