Association for Information Systems
AIS Electronic Library (AISeL)

December 2007

Improving the Knowledge Base by Integrating Vital Sign Data into Pre-Hospital Patient Care Records

Mark Gaynor
Boston University

Dan Myung

Joseph Restuccia
Boston University

Steven Moulton

Exchanging Electronic Patient Care Records with Web Services

Mark Gaynor, Boston University School of Management, mgaynor@bu.edu
Dan Myung, 10Blade, dan.myung@gmail.com
Joseph Restuccia, Boston University School of Management, jres@bu.edu
Steve Moulton, Boston 10Blade, smoulton@10blade.com

Abstract

*This paper discusses how to share medical information between heterogeneous applications via web services. Our design theory is based on experience building **iRevive**, a working web services enabled pre-hospital documentation application. The tradeoffs between efficiency and flexibility are examined in the context of exchanging information based on emerging standards in the healthcare world. We illustrate the importance of uncertainty in deciding the architecture enabling an application to access medical information from Patient Care Records (PCRs).*

Keywords: Web services, Exchanging medical data, Standards.

Introduction

This paper describes an evolutionary framework that helps designers and architects select appropriate software modules for designing and implementing web service applications that share information from Patient Care Records (PCRs). The tradeoffs between flexibility and efficiency are discussed in the context of how the many fields of a PCR should be offered for consumption by web service based medical applications. Designers should concentrate first on flexibility and then evolve to more focused and efficient solutions as usage patterns emerge. Our design science approach is based on several iterations of the *iRevive* application that exchanges medical information relating to the PCR via web services. This methodology enhances understanding the uncertainties associated with how this medical information will be used and how this influences the architecture of these web services, resulting in web service implementations that are a better fit.

Designers are faced with many choices when designing architecture to encourage exchanging data from PCRs such as:
- a monolithic design providing all fields in the PCR;
- a fined grained approach providing a web service for each field of the PCR;
- a flexible web service that provides a group of fields requested by the client;
- medium grained access to a pre-defined group of fields from the PCR.

These four strategies differ in flexibility and efficiency. The monolithic design is the most efficient for exchanging the entire PCR, but inflexible for providing a more limited set of fields that might be useful for medical research where patient identify must be hidden. Fined grain access to each field in the PCR is flexible, but very inefficient from the viewpoint of both processor usage and network bandwitdth. A flexible web service that dynamically returns any set of fields requested is harder to code in both server and client software because of the more complex Application Program Interface (API). Access to pre-defined groups of fields is efficient but not flexible to evolving requirements. These different architectures meet application demands, which is why we propose a methodology that evolves from a focus on flexibility to one of efficiency that scales as the application becomes more popular.

As an illustrative example, we use the exchange of medical information from a pre-hospital EMS application that creates an Electronic Patient Care Record (PCR) called *iRevive* [Gaynor et al 2007]. The *iRevive* patient care record is complex and consists of many forms of related data including: Demography of the Patient, General Physical Exam, Assorted Vitals, Airway Evaluation, Respiratory Exam, Medication, and many others. Because of the uncertainty about which systems require what information, a flexible architecture must be in place that allows experimentation among the many players, i.e., arriving hospital, payer, and researchers that need access to this medical information.

The *iRevive* application is a working system that has undergone several development cycles including testing with medical professionals such as Emergency Medical Technicians (EMTs) and trauma experts. Our team has input a randomly selected set of paper based PCRs with EMTs from a major air transport Emergency Medical Services (EMS) provider to evaluate the effectiveness of our Graphical User Interface (GUI) and completeness of the database. We have experimented with matching pre-hospital PCR data with in-hospital PCR information as discussed in the section describing *iRevive*. From the beginning, *iRevive* was built around emerging standards for exchanging medical information because our research team believes interoperability is a key factor to improving the effectiveness of Information Technology (IT) in the healthcare space. We also believe that web services are emerging as the technology to exchange information between distributed applications. Combining web services with medical data encoding and messaging standards such as Health Level 7 (HL7) [HL7 2005], Systematized Nomenclature of Medicine (SNOMED) [SNOWMED 2007], and International Statistical Classification of Diseases and Related Health Problems (ICD-9) [ICD-9 2007] will promote semantic interoperability between heterogeneous medical IT applications.

We propose that user uncertainty about what data they require should be a main factor in determining the architectural flexibility for data sharing infrastructure. User uncertainty is the inability to predict what users want (for example what fields are required for research related to trauma). This framework suggests that environments with high uncertainty need a more flexible architecture that promotes experimentation by users to discover which fields should be grouped together for a more scalable information exchange. Low uncertainty implies that the designers can predict which subset fields each group of users will require.

Web services are software components accessible over the Internet [Iyer et al 2003] that have the potential to alter business processes in many organizations because of the easy integration of internal systems and compatibility with other external systems and provide a great example to apply our framework. However, web services have a high overhead because of the Simple Object Access Protocol (SOAP) [SOAP 2003] envelope, and Extensible Markup Language (XML) [XML 2007] processing, which argues for courser-grained modules when performance is important. Our model factors in uncertainty and balances it with performance requirements. High market uncertainty implies that it is acceptable to be flexible and inefficient, because if you become successful, you can then evolve the design into a more efficient composition of modules for data exchange.

Previous Work

Modularity

This problem of designing infrastructure to share PCR data is similar to the problem of module decomposition of any system. The value of modularity in complex systems is understood [Baldwin and Clark 2000], yet deciding how many modules are needed, the composition of each module, and the modules scope of visibility, is still more art than science. Software engineering techniques such as Design Structure Matrix (DSM) help designers break apart complex systems, but does not account for the cost of modularity, the preferences of users, and most importantly, uncertainty in what users want. Modularity creates value, but this value must be maximized within the constraints of budgets and user demands. We argue that user uncertainty should influence module de-composition in complex software systems. When uncertainty is high, architecture that promotes experimentation creates greater value because it enables the discovery of the module decomposition that best fits a particular situation. When uncertainty is low about what fields of a PCR a heterogeneous group of users need access to, then flexibility creates little value [Gaynor 2002; Gaynor and Bradner 2004] compared to efficient architectural solutions.

The modularization of complex software systems is difficult because there are forces pulling design engineers in opposing directions. Arguments about the value of modularity push designers to minimize functionality of each module (i.e. a separate web service for each filed in the PCR) because it promotes experimentation, and efficient parallel development. However, excessive modularity is impractical for several reasons: the cost is high, performance suffers; providing designers too many choices may create an intractable search for the best solution [Fleming and Sorenson 2001]; and users may prefer modules that do more. Invoking a web services for each field in the PCR to exchange an entire PCR has poor performance because it uses a lot of bandwidth and CPU cycles on both the client and server. We believe that creating high value designs in complex environments implies that modules with high uncertainty should be functionally flexible to encourage the experimentation required to find optimal solutions, while modules with low uncertainty can be static and efficient.

Cost/Benefit of modularity

Research on modular structures has identified two key benefits of adopting modular designs. First, modularity allows module designers to engage in parallel innovation efforts while maintaining overall conformance to the interface specifications. The key pay-off here is the reduction of time spent on design, development, testing, and integration [Ethiraj and Levinthal 2004]. The second key benefit of modularity is the increase in the number of possible configurations achievable from a set of inputs, thereby increasing the flexibility of a system [Schilling 2004]. On the down side, when systems migrate toward increased integration they tend to return greater performance. The literature has also cautioned against adopting modularity without considering the trade-offs [Ethiraj and Levinthal 2004] [Fleming and Sorenson 2001]. Fleming and Sorenson [Fleming and Sorenson 2001] show that intermediate levels of modularity tend to produce the most useful inventions. Excessive modularization can blind the designer to potentially important interactions between design choices. In this case, the speed and efficiency gained from modularization can be offset by the increased time spent in testing and integration, where the consequences of ignored dependencies will come back to haunt the designer.

DSM matrix

The Design Structure Matrix is one way to break a complex application down into modules. This technique was originally developed for analyzing design descriptions [Steward 1981]. Recently, it has been extended to analyze development projects modeled at the task level [Eppinger et al 2000] and for software design [Sullivan et al 2001]. In this model, a large project is broken down into tasks and each task is assigned to a row in the matrix. The columns are then named identically. Each cell is then marked if there is a dependency between tasks. As a result of this, scanning a row reveals all of the tasks whose output is required to perform the task corresponding to that row. Traversing down a column reveals which tasks receive information from the task corresponding to that column. If all the tasks are only sequentially dependent according the matrix layout, no marks would appear above a diagonal. Applying the DSM to design the infrastructure for sharing information within a PCR is straight forward. The rows and columns are the list of fields. Each cell is marked if the

fields are related to each other. Examples of related fields include: the Last Name and First Name; Last Name and Home address; and all vital signs.

Table 1 illustrates how a DSM might look for a PCR for different user groups such as medical professionals and administration billing. Heterogeneous user groups will require different DSM matrix representations as illustrated in Table 1. This is a simplified set of fields that include several important types of information such as patient demographics (Gender, Age, Address, Zip Code, and Birth Day), administration data (Date of Admission, Name, and SS #), and medical data (Vital signs, Diagnostic and Treatment Codes, and Medications). The Medical Doctor (MD) requires a complete view of a patient's condition; however, Social Security Numbers (SS#), Address, and Birthday are not relevant to medical treatment and should not be available as Table 1 (A) shows because it might violate patient privacy. Table 1 (B) illustrates the needs of administrative users that need most data except for vital sign of the patient that are unrelated to billing and other administration tasks.

The main weakness of a DSM is the inability to management uncertainty. Groups such as medical researchers that have more undefined and uncertain requirements don't benefit from this type of analytic tool because the users don't know what the relationship between fields will be. This weakness is illustrated in our data matching example where we did not know the optimal data set to anonymously match pre and in hospital PCRs. Different research studies have different dependencies that are not well understood prior to experimentation, which is not well suited to the DSM approach.

Legend

A – Gender	G – Vital signs
B – Age	H - Name
C – Address	I – SS#
D – Date of admission	J – Diagnostic Code
E – Zip code	K – Treatment Code
F - Birth date	L - Medications

A — MD

	A	B	C	D	E	F	G	H	I	J	K	L
A		x	o	x	o	x	x	x	o	x	x	x
B	x		o	x	o	x	x	x	o	x	x	x
C	o	o		o	o	o	o	o	o	o	o	o
D	x	x	o		o	x	x	x	o	x	x	x
E	o	o	o	o		o	o	o	o	o	o	o
F	x	x	o	x	o		x	x	o	x	x	x
G	x	x	o	x	o	x		x	o	x	x	x
H	x	x	o	x	o	x	x		o	x	x	x
I	o	o	o	o	o	o	o	o		o	o	o
J	x	x	o	x	o	x	x	x	o		x	x
K	x	x	o	x	o	x	x	x	o	x		x
L	x	x	o	x	o	x	x	x	o	x	x	

B — Administration

	A	B	C	D	E	F	G	H	I	J	K	L
A		x	x	x	x	x	o	x	x	x	x	x
B	x		x	x	x	x	o	x	x	x	x	x
C	x	x		x	x	x	o	x	x	x	x	x
D	x	x	x		x	x	o	x	x	x	x	x
E	x	x	x	x		x	o	x	x	x	x	x
F	x	x	x	x	x		o	x	x	x	x	x
G	o	o	o	o	o	o		o	o	o	o	o
H	x	x	x	x	x	x	o		x	x	x	x
I	x	x	x	x	x	x	o	x		x	x	x
J	x	x	x	x	x	x	o	x	x		x	x
K	x	x	x	x	x	x	o	x	x	x		x
L	x	x	x	x	x	x	o	x	x	x	x	

C — Research

	A	B	C	D	E	F	G	H	I	J	K	L
A												
B												
C												
D						?						
E												
F												
G												
H												
I												
J												
K												
L												

Table 1

Motivation to Share PCR Information

The benefit of interoperable electronic medical records are huge and multi-faceted with both direct and indirect advantages to heath care providers, vendors of health care goods and services, insurance companies, medical researchers, and most importantly to those receiving medical treatment. In aggregate, the savings from PCRs has been estimated as high as $77 billion per year [Walker et al 2005]. Health care providers, such as hospitals, benefit with more efficient systems that reduce costs while cutting errors, and many clinicians and emergency rooms have made the transition to electronic patient care records, citing improvements in documentation and seamless integration with prescription writing and laboratory reporting systems as overwhelming benefits. Vendors and service providers benefit from a standards based infrastructure to facilitate the exchange of medical information. This will create a rich eco-system of vendors and service providers similar to what developed around Internet and Web standards. Insurance companies will benefit with more accurate billing as the PCR is the most important form used to justify reimbursement for medical services, less redundant tests, and fewer medical errors (which are expensive to troubleshoot). Medical researchers benefit with higher data quality that electronic PCRs enable. Lastly, patients benefit the most for several reasons including: fewer errors; less overlap in testing and querying; and ultimately lower health care costs. The overall advantages of electronic medical records are too significant to ignore in today's environment of rising medical costs.

Interoperability with PCR data will promote the development of better evidence-based treatment protocols, especially in the area of trauma. Traumatic causes of collapse and early death are among the most treatable and are well suited for study, as most victims have no underlying illness. Trauma is also common: it is the leading cause of death during the first three decades of life in the United States and ranks as the fourth leading cause of overall mortality, with over 100,000 traumatic deaths each year [Shires 1985] [Trunkey 1984]. For these patients, it is important to start gathering information with a PCR at the point of care, as the first hour of treatment is often the most critical in determining the patient's eventual outcome. Early collection of accurate and consistent pre-hospital data is a necessary step towards developing more effective emergency medical treatment strategies during this critical phase of patient care.

Prime examples of medical research that will benefit with interoperable PCRs are Traumatic Brain Injury (TBI) and exsanguinations, which are the two most common causes of traumatic death, making the management of head injury, hemorrhage and fluid resuscitation integral parts of early resuscitative care. Nearly 52,000 people die every year in the United States as a result of TBI and another 70,000 to 90,000 develop permanent neurological impairment, resulting in an estimated cost to society of over 40 billion dollars annually [Thurman et al 1999] [US Dept HHS 1989]. Unfortunately, despite the development of evidence based guidelines for the management of severe traumatic brain injury [Anonymous 1996] [Anonymous 2000], a wide spectrum of methods still characterizes most treatment strategies. This is primarily due to the complex nature of these injuries, including the time-sensitive interplay of multiple physiologic factors (temperature, blood pressure, oxygen delivery, intracranial pressure (ICP)), other injuries, and their many treatment variables (fluid management, ICP therapy, hyperventilation, temperature regulation, mannitol, vasopressors, steroids, etc.). The ability to exchange information from the PCR is critical in discovering improved treatment methodologies, and determining which pre-hospital interventions correlate with the lowest eventual morbidity and mortality. This will provide researchers with more accurate and abundant evidence to fine-tune pre-hospital trauma guidelines.

iRevive

The high level view of the *iRevive* system in Figure 1 illustrates the five phases of a typical EMS mission: dispatch; pickup; transport; drop-off; and documentation completion and data transfer. Data capture begins when the call comes into the EMS dispatch center and continues in all phases of a mission. After dispatch, the EMS transport vehicle (helicopter, jet or ground ambulance) arrives at the patient pick up point, which is either a medical facility or injury scene, where *iRevive* is used by EMTs nurses and paramedics to enter data, including the patient's current condition. As transport commences, emergency medical specialists treat the patient, and time permitting, continue the documentation process. The data capture phase ends after patient care is transferred to the care of physicians and nurses at the receiving hospital. At this point, patient data can be transferred into in-hospital medical IT systems. The final phase requires completing all necessary documentation of the mission and transferring this data to the EMT database for billing, storage and future retrieval. Documentation begins at Phase one and ends when the medic completes the documentation for the call.

Figure 1 – High Level View
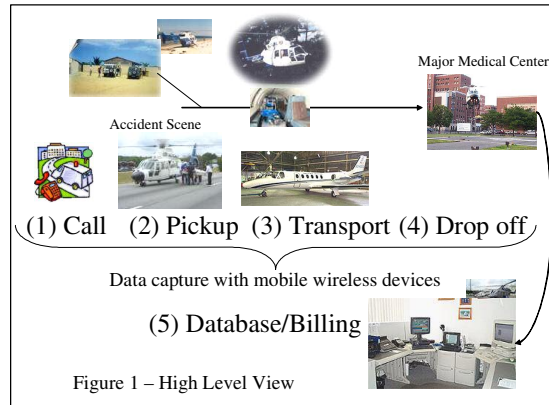
Figure 2 illustrates how the *iRevive* application [Gaynor et al 2007] will be used in the field by a typical EMS service provider. The current version has undergone several iterations of testing by medical professionals with real patient data and is ready for in-field testing[1]. The arriving medic places wireless vital sign sensors on one or more patients. Each medic is equipped with a ruggedized tablet PC that captures and displays the real-time sensor data and allows the documentation of observations and treatments. Data capture is automated for vital sign data, while data on observations and treatments is manually entered by the medics. This data entry is guided by a set of rules that enforce consistent and complete capture of data. Local medics are linked to the transport aircraft via an 802.11 wireless infrastructure that enables situational awareness of the aircraft crew so they can prepare for any additional medical interventions that may be required. Each transport vehicle is equipped with a base station that links to local technicians, command centers, and destination hospitals. This WAN linkage enables global allocation of resources, and increased awareness of the condition of incoming patients at the destination hospital. During patient transport, *iRevive* continues to capture both sensor data and data recorded by medical personnel. The *iRevive* application enables creating and transferring an electronic patient care record that combines automated capture of vital signs with manually entered data on observations and interventions performed.

Figure 2 – iRevive Use Case

---
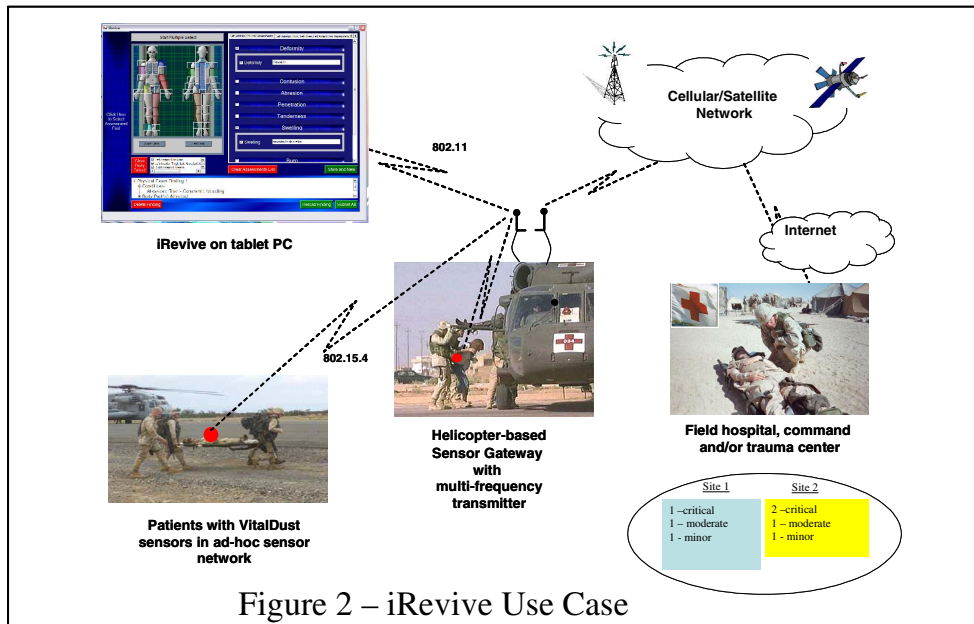
[1] We are working with several organizations towards in-fields testing of *iRevive*.

One of our research goals with iRevive is matching pre-hospital PCRs with in-hospital PCRs to evaluate the effectiveness of in-field treatment. Because of HIPPA regulations and other privacy concerns it is best if this matching is accomplished without identifying the patient. However, it is not clear what database fields are best for this anonymous record matching. We are experimenting with 3 different groupings of fields. All true matches were confirmed by comparing manually recorded medical record numbers from the two year period. We experimented with the following queries:

1. **de-identified data query 1**: gender + initial vital signs from the field (systolic BP, HR, GCS);

2. **de-identified data query 2**: query 1 + patient age + state;

3. **limited data query 3** (query 1 + query 2 + date of admission + pt. zip code + pt. date of birth).

This experiment illustrates the needs for medical researchers to have flexibility in accessing data from the PCR. These results will be published in a follow-up paper.

Evolutionary Strategy to Share Medical Information

One of the design choices faced when designing *iRevive* was the modular decomposition for sharing information from the pre-hospital PCR because it is not clear what fields need to go to which parties. Experimentation is needed to determine the optimal set of fields to exchange with the many data consumers including the EMS provider, the receiving hospital, the payer organization, the billing agency, internal business processes (i.e. QA/AI functions), and medical researchers. By building an infrastructure that promotes easy and inexpensive experimentation, the designer can improve the odds of reaching the goal of effective exchange of medical information in an uncertain environment.

Our approach evolves enabling it to mitigate the uncertainty of pre-determining which groups of fields will be accessed by which group of users. In the initial learning phase users have freedom to experiment with the granularity of access to PCR data. Next, after learning what users want the designers can build efficient web services that return the most frequently used subsets of the PCR. At first uncertainty implies that a flexible solution will out value a more efficient architecture that is more appropriate later on when users have a better understanding of their usage patterns.

There are many choices of architectures which to choose from when providing access to this pre-hospital PCR: from the most modular infrastructure (shown in Figure 3(a)) to the most monolithic design illustrated in (d). These various strategies have different advantages and disadvantages. Fine grained modularity promotes experimentation, but is far from efficient; the single monolithic structure is efficient for exchanging all fields in the PCR, but does not provide any flexibility. There are two intermediate architectures shown such as: (b), a flexible large module that provides fine grained access to the fields in the PCR; and (c), which provides modules of greater functionality than (a), but not the monolithic design in (d).

Figure 3 – Modularity in Web Services
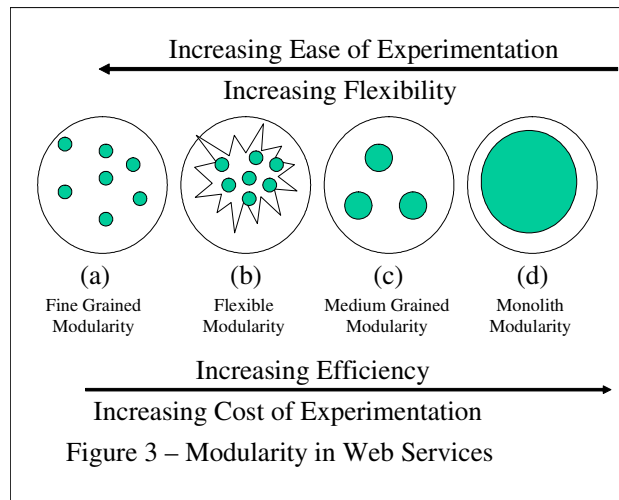
Below we describe each architecture and discuss the advantages and disadvantages:

- **Fine Modularity (a)**: Having a web service for each field in the PCR is very flexible for data consumers because they can pick and choose the fields they want. Experimentation is easy since the user just calls the group of web services required to get the desired sub-set of fields. The main drawback of this approach is the inefficiency from the point of view of processing and network bandwidth when consuming more than a few data fields because each call to a web service incurs significant overhead. Security management is difficult with fine modularity because the application must keep track of which fields users have accessed to verify the users have authorization to view the requested information.
- **Flexible Modularity (b)**: With this structure the designer creates one web service that provides access to any single field, or any set of fields. This is a flexible solution for the data consumer, but a bit harder to experiment with than the single fine grained modules because of the more complex API. It uses less bandwidth than (a) when sending several fields of data. Security management is similar in complexity to (a).
- **Medium Modularity (c)**: This middle approach uses modules of medium granularity. It is not very flexible since the data consumer can only get the fields in pre-determined groups. For example, one module might provide the data the billing agency requires, while another module will provide information to medical researchers. However, if different researchers need different data, then a new module must be created. This has a balance between efficiency and flexibility. This has easy to manage security because each web service can check that the user is authorized for the particular group of fields provided.
- **Monolithic (d)**: With this simple and inflexible approach one module provides all the fields in the PCR to all consumers. This is very efficient when the consumer wants many data fields, but not very efficient if the user only requires a single or few fields. It is not flexible enough to provide different sub-sets of fields to different users. No experimentation occurs since the user has only one choice, all or nothing.

The best structure for a particular situation depends on the uncertainty and resource requirements of the server, client, and network infrastructure. Greater uncertainty argues for the solutions on the left side of Figure 3, while requirements that are well understood call for bigger, static and efficient modules. When there is uncertainty you need experimentation to determine the best modularity structure. This experimentation will discover which fields are accessed as single and which sub-sets of fields are most desired. For example, experimentation is required to discover the fields of most interest to medical researchers. This knowledge then can be used to build a web services application providing this sub-set of information.

We have proposed two very different methodologies that promote experimentation with data consumers of information from the PCR. First, a fined grained structure that allows access to each field of the PCR as a separate web service is easy to build since each web service is simple. The other approach is to provide a complex flexible interface that enables any combination of fields from the PCR to be accessed. This flexibility is harder to maintain because of the increased complexity of a flexible full-featured module. Both these architectures allow experimentation with market selection. As data consumers access this medical data, patterns of usage will emerge to aid designers in determining the most effective modularity.

The tradeoff between flexibility and efficiency is important for the development of distributed medical IT applications. Flexibility can be measured by observing the effort to access different sub-sets of the PCR in the context of program development and maintenance. Efficiency is a combination of network bandwidth and CPU cycles. The flexibility/efficient tradeoff will emerge from analyzing the logs of how users access the PCR. Flexible solutions are critical to discover how to build new applications; and then if they become successful, efficiency becomes more important so the application can scale up for more users.

Figure 4 is a grid that illustrates when to use which infrastructure. The upper left signifies that when you know a particular set of fields that users will need to access, it is best to build efficient web services to produce the requested data. Several examples are: a set of demography data that identifies a patient and provide the information necessary for billing the emergency medical services; information about a patient's condition, treatment and response useful for medical research into areas such as TBI without demographic information that could identify the patient; the complete PCR for use within the EMS organization. The right side of Figure 4 illustrates the regions for experimentation at the beginning, with the variations in how one might expect to access the information and resource requirements. When several fields are required, then the top right quadrant uses less bandwidth, but for a single field the lower right region is the most efficient choice. The points on the right side of the grid enable experimentation to migrate into more efficient solutions on the left hand side of Figure 4.

Figure 4 – Applications and Architecture
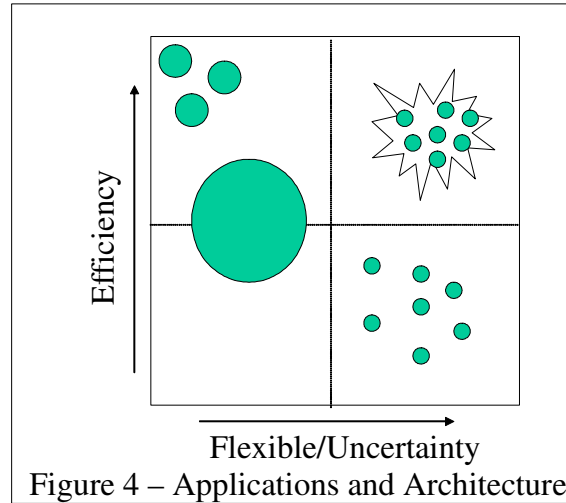
Various applications require different infrastructure as some clients have limited processing power while other clients are constrained by bandwidth. For clients trying to limit processing resources or reduce the required bandwidth required to receive several fields, the upper right side of Figure 4 will provide flexibility to explore because only one web service call is required to return any set of fields the client needs. However, if the set of fields are commonly requested then the upper left side of Figure 4 is a better fit. When only a single field is requested, the lower right side of the gird is the optimal solution. The operating environment drives the choice of infrastructure, uncertainty pushes towards more flexibility, and thin devices push towards more efficient infrastructure to reduce processing on the client, and low speed wireless networks, such as traditional cellular, encourage bandwidth efficient architecture.

We envision an architecture where the PCR resides in a SQL database with the records queried by the web service. For flexible web services in the right upper grid of Figure 4 the client needs to specify which fields to return and the server must parse this list to build the correct SQL request to access the requested fields. In this case, all fields will be returned with a single call to the web service. The three other regions are web services that have a pre-determined set of fields to access. This means the client does not need to specify parameters, and the server does not need to create a dynamic SQL query. Combining a SQL data base with a range of web services provides a flexible infrastructure to exchange PCR information.

The complete picture is illustrated in Figure 5. It shows how uncertainty, task decomposition, cost/benefit of modularity, and user preferences all affect the modular decomposition of a complex web service. The goal is to discover the best mix of fine, medium, and course grained access to data. In uncertain markets, experimentation is required to find this solution. The web services designs that promote this experimentation, such as fine grained access or flexible access to data, will promote the requisite experimentation.

Figure 5 – Applications and Architecture
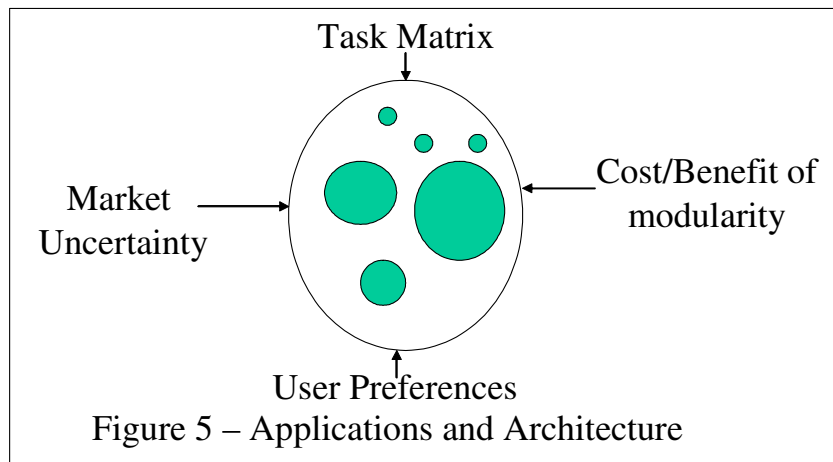
Conclusion

Our evolutionary design theory accounts for user uncertainty and application requirements by encouraging flexibility when uncertainty is great, and efficiency when user requirements are better defined. At first, new web services with high uncertainty in usage should have fine granularity or flexibility. Later as uncertainty is lower, web services should provide groups of fields that users frequently request as efficiently as possible.

References

1   Anonymous. Guidelines for the management of severe head injury. Brain Trauma Foundation, American Association of Neurological Surgeons, Joint Section on Neurotrauma and Critical Care. J Neurotrauma 1996; 13:641-734

2   Anonymous. Guidelines for cerebral perfusion pressure. The Brain Trauma Foundation, American Association of Neurological Surgeons, Joint Section on Neurotrauma and Critical Care. J Neurotrauma 2000; 17:507-11.

3   Baldwin, C and Clark, K, Design Rules: The Power of Modularity. Cambridge, MA: MIT Press, 2000.

4   Eppinger, S,and Whitney, D, and Smith, R, and Gebala D, "A Model-Based Method for Organizing Tasks in Product Development," Research in Engineering Design, 1994.

5   Ethiraj, S and Levinthal,D, "Modularity, Innovation in Complex Systems," Management Science, vol. 50, pp. 159-173, 2004.

6   Fleming, L and Sorenson, O., "Technology as complex adaptive system: Evidence from patent data," Research Policy, vol. 30, pp. 1019-1039, 2001.

7   Gaynor, Network Services Investment Guide: Maximizing ROI in Uncertain Times: Wiley, 2002.

8   Gaynor, S. Bradner, M. Iansiti, and HT. Kung, "The Real Options Approach to Standards for Building Network-based Services," presented at IEEE conference on Standardization and Innovation, Boulder Co, 2001.

9   ICD-9, http://www.cdc.gov/nchs/icd9.htm, 2007.

10 Gaynor, M., Bradner, S. A Real Options Metric to Evaluate Network , Protocol, and Service Architecture, Computer Communication Review(CCR), Oct 2004.

11 Gaynor M, Myung D, Winkler D, Ganesan S, Moulton S. An intelligent pre-hospital patient care system, Accepted to the International Journal of Electronic Healthcare 2007.

12 HL7, Health Level Seven. 1997 - 2005 Health Level Seven, Inc. http://www.hl7.org/

13 Iyer, , Gaynor, and G. Wyner Freedman, "Enabling Dynamic Business Networks using Web Services," Communications of the Association for Information Systems, 2003

14 Schilling, M, "Toward a General Systems Theory and its Application to Interfirm Product Modularity," Academy of Management Review, vol. 25, pp. 312-334, 2000.

15 Shires, GT. Principles and management of hemorrhagic shock. In: Shires GT, ed. Principles of Trauma Care. New York: McGraw-Hill; 1985:3-42.

16 SOAP, The World Wide Web Consortium. SOAP Version 1.2 Part 1: Messaging Framework. W3C Recommendation 24 June 2003. . http://www.w3.org/TR/SOAP

17 Steward, C, Systems Analysis and Management: Structure, Strategy, and Design. New York: Petrocelli Books, 1981.

18 Sullivan, Griswold, Cai, and Hallen, "The Structure and Value of Modularity in Software Design," presented at Joint International Conference on Software Engineering and ACM SIGSOFT Symposium on the Foundations of Software Engineering, Vienna, 2001.

19 SNOMED, http://www.snomed.org/, 2007

20 Trunkey D, D. Trauma. Sci Am. 1983; 249:28-35.

21 U.S. Department of Health and Human Services. Interagency Head Injury Task Force Report. Washington, DC: U.S. Department of Health and Human Services; 1989.
22 Walker, J., Pan, E., Johnston, D., Adler-Milstein, J., Bates, D. W. and Middleton, B. (2005) "The Value Of Health Care Information Exchange And Interoperability." Health Affairs.
23 XML, The World Wide Web Consortium. "Extensible Markup Language" (XML). http://www.w3.org/XML, 2007.