International Journal of Digital Information and Wireless Communications (IJDIWC) 4(1): 10-23
The Society of Digital Information and Wireless Communications, 2014 (ISSN: 2225-658X)

Structuring Heterogeneous Big Data for Scalability and Accuracy

Ashraf Gaffar
Dept. of Engineering & Computing Systems
Arizona State University
Mesa, Arizona, USA
agaffar@asu.edu

Eman Monir Darwish
Dept. of Fine Arts
Helwan University
Cairo, Egypt
eman.monir@hu.edu.eg

Abdessamad Tridane
Dept. of Mathematical Sciences
UAE University
Al-Ain, UAE
a-tridane@uaeu.ac.ae

ABSTRACT

Structured data has an inherently great automation value. It renders itself readily for software tools to help store, organize and search effectively. With the growing dependence of data, we face many new problems. While software applications replace each other, and older software has rapidly diminishing value, data has the extreme opposite nature, which we can call "cumulative effect". Unlike software, we see new data as an addition to the old one, so we tend to continuously accumulate date without deleting any thing. Even older data is often archived for it's value, for legal reasons, or just because we can never be sure if we'd need them again. This would not be a problem if we had structured data, as we can automate the storing and retrieval process. However, most of the valuable information lie inside unstructured data, which is extremely difficult to store and retrieve in large scale. Our work shows a new concept of adding structure to highly unstructured, heterogeneous data, which will greatly improve the total process of storing and effectively retrieving them. We use Human-Computer-Interaction (HCI) patterns as a case study to present our proof of concept.

KEYWORDS

HCI patterns, information encapsulation, semi-structured data, information dissemination, information reuse

1 BACKGROUND

The concept of patterns lends itself to human nature. Patterns are an effective approach of disseminating knowledge gained by expert practitioners over the years. Christopher Alexander [1], [2], [3] first introduced them in his work about designing living spaces inside homes and outside home communities. In the pattern community, it is widely accepted that we need to have a look at Alexander's original work to get a real sense of what patterns really are and how they can be used [4]. Several domain experts followed the same pattern approach and started documenting their years of experience in terms of patterns. One major landmark in pattern work is the "Design Pattern" book published by the so-called "Gang-of-Four" [5]. This seminal book made a major breakthrough in disseminating highly sophisticated programming experience from well-known experts to novice programmers by following the pattern approach. This major success motivated several other domain experts to start adopting the pattern approach in wrapping and disseminating their years of experience and delivering it to the novice practitioners in their domain.

The idea of using patterns is fairly simple and straightforward. A pattern provides a solution to a problem in a specific context, allowing pattern users to get directly to a suitable solution to a problem without having to go through extensive theoretical background, analysis, or previous attempts. With their growing success, we ended up with a tremendous amount of patterns, or a plethora. While extremely useful in most cases, as the number of patterns grows exponentially, patterns users started to be overwhelmed as to finding the most suitable patterns and applying them to their work. Unaware of the problem, patterns authors continued to generate patterns as a way of documenting their domain experience, but did not go far enough to consider the environment of how they are being accessed or reused. Very little effort was spend on understanding how to effectively reuse the plethora of existing patterns [6]. To complicate the problem, patterns –by nature- do not live in isolation. They are typically used in combination, so when they are generated by different authors, the concept of combining them together was a growing problem. It was typically left up to the user to decide on that using their own judgments, and following ad-hoc processes [7]. Redundancies and contradictions were common problems in pattern reuse, resulting in less-than-optimal solution or even erroneous designs.

This particular problem of pattern reuse was identified and discussed in details by different domain experts [8], [9] and [10]. Different tools were build to try and present a possible solution [11], [12], [13]. Unfortunately, most work approached the solution "after the fact". They tried to deal with the heterogeneity of patterns as an inherent nature without trying to generate more structured pattern at a deeper information level.

2 SEARCH BY SYNTAX: A RANKED GUESS WORK

To demonstrate our work, we examined multiple pattern collections, and selected Human-Computer-Interaction (HCI) patterns as case study of both unstructured and large collection of data. HCI patterns are typically visually oriented, so they are presented in a heterogeneous and multi-modal format, including text, code snippets, graphics, flow charts, and often screen shots. While few patterns can be easily understood and applied, previous usability tests on UI designed generated using large number of unrelated HCI patterns have shown several problems in applying HCI patterns in generating new designs [14].

In order to identify the problem, and follow an approach of "separation of Concerns", we can generally split the pattern lifecycle into 2 phases as follows:

1) Pattern Generation Phase: Defines how patterns are generated by the domain experts.
2) Pattern Dissemination Phase: Defines how patterns are communicated from the pattern author to the pattern user.
3) Pattern Reuse Phase: Define how patterns are consumed and reused to generate new designs.

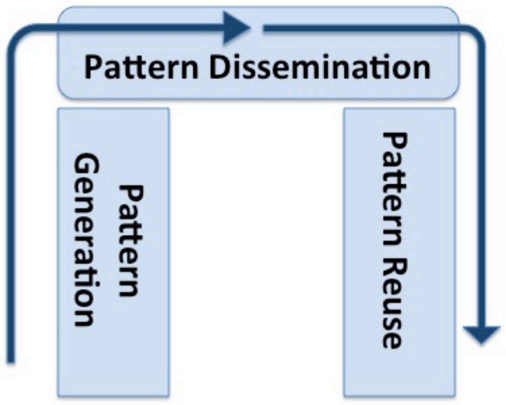

Figure 1: The Pattern Lifecycle

Figure 1 depicts the three phases of the pattern lifecycle.
Defining this lifecycle sheds some light on the main roles associated with patterns. Apparently, the key roles associated with patterns are the "Pattern Author", and the "Pattern User". It is clear that in the overwhelming majority of cases, different people, namely the domain expert, and the novice designer, assume these two roles respectively. Having identified both roles shows the missing link, "who is taking the role of pattern dissemination?" Apparently, the pattern community has relied on distributed responsibility of disseminating patterns, with the majority of the work left to pattern authors. In this regard, pattern authors would present their patterns to the community via conference papers, books, or through their own websites. However, this does not solve the problem. Pattern users still have to "find" the respective patterns, which are often present in multiple disparate sources. This is the neglected task that ends up overwhelming pattern users as they sift through several pattern collections, trying to mix and match them without author's support.
To focus more on the core of the problem, [6] defines ta search behavior, namely the "*Structured Data Continuum*" as depicted in Figure 2.

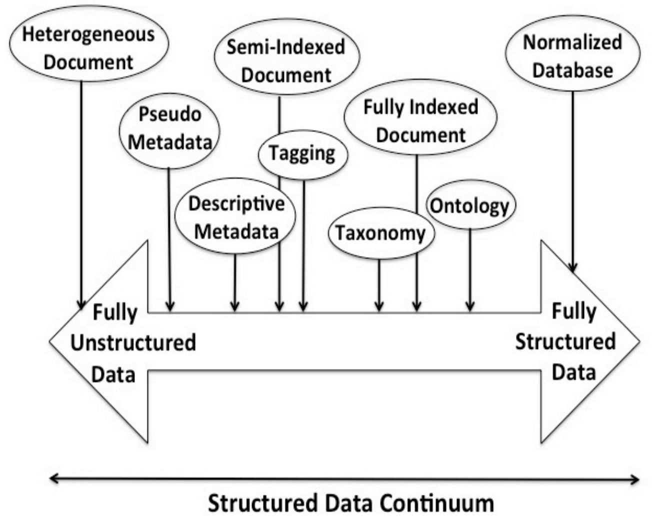

Figure 2: Structured Data

Structured data, presented on the right hand side of the scale can be automatically processed at large scale without any problems. However, structured data are , but they are expensive to create from unstructured data. One common example is database creation. An extensive process of analysis and modeling is required before we are able to generate a well-structured database from daily information.

The other side of the scale the far left hand side) presents the other extreme of data. On this extreme, we typically have rich text documents solely intended for human consumption. They typically have no specific form, and contain multiple modality including text, images, graphs, and even audio and video components. Even when we represent them in an electronic format as in a web page, or an office document (for example MSWord, or MS Power Point), they can be very hard to process at large scale due to the lack of structure. The multi-modal content can be extremely pleasant for a human reader to read and comprehend, regardless of the multiple modality format, or the lack of structure. In fact, we tend to love a heterogeneous document with a coherent context rather than a fully structured excel sheet document for example. One of the main reasons is that a human reader looks for and easily understands the underlying semantics and meaning of a document regardless of how it is presented. Computers, on the other hand, have very hard time extracting the meaning from a text, let alone an image or a video [8].

Software applications use complex algorithms to extract and manipulate the underlying meaning of text sentences. Extensive work is done in the domain of Natural Language Processing (NLP) that is beyond the scope of this paper. To just give a sense of the problem and elaborate more on the semantics challenge, we demonstrate using some examples:

Example 1: If we looked at a common phrase like "*Mr. Brown is wearing a Puma sneakers*". We can easily recognize "*Brown*" as the family name of a person, and we can also recognize "*Puma*" as a common brand of a sports shoe manufacturer. Software applications can have significant challenges trying to understand the actual meaning of those two words, and have to revert to sophisticated algorithms to guess their actual meaning.

Example 2: Searching for a particular picture in a large library of pictures can be a daunting task. If we were matching pixel for pixel of two perfectly identical pictures, it would be easy for a software application to compare them. When it comes to the contents of an image, it is a completely different challenge. Even two identical pictures that have some lighting variations, or are slightly rotated, zoomed in or out, or taken from a slightly different angel can be extremely hard to identify as similar using software. Complex imaging and graphics algorithms, fuzzy logic and pattern matching can help identify the closely similar images [15], [16].

This approach suffers when the images are highly different in pixel contents, but similar in their physical contents. An example would be similar objects laid out differently in two pictures, or two pictures of large-breed dog, and a small-breed dog.

To be able to work our way round such problem, we can identify the "meaning" of the picture contents by adding textual description of the content and linking it to the images. By using standard keyword search of these textual descriptions, we can get sufficiently good search results. The problem with this method is that we need to write a large number of highly descriptive keywords that describe all aspects of the picture, including objects, layout, context, and even associated emotions and feelings. In this case the meta-data writer has to come up with sufficiently descriptive set of keywords to describe what the picture contents are. Equally important, the user (searching for particular pictures) has to come up with sufficiently descriptive set of search terms of what they're looking for in a picture, and hope for good matches with the existing metadata terms, or at least a common overlap between the two sets. This solution fails when metadata is not descriptive enough or absent altogether resulting in low or no correlation. The failure can also result when the user is unable to find descriptive enough keywords or not sure what they are exactly looking for. We can further argue that this is a way of redirecting the core search activity into a correlated search of auxiliary contents rather than the original one. Users are basically searching textual information (the metadata), and not the original data, and the pictures we get are dependent on the quality and expressiveness of the auxiliary text as well as on the search keywords, and not on the original contents.

The cost of writing additional meta data to images can be prohibitively large, especially if we need to create sufficiently descriptive data to a large number of pictures. A good example is if we need to describe all images available on all websites on the Internet. We can see that the two dimensions "*highly descriptive metadata*" and "*large number of images*" makes it an extremely difficult task. Google recently came with an elegant solution to work around this difficulty. Considering the two dimensions we explained above, Google opted for a great reduction of cost by trading off the highly descriptive metadata generation with the "existing accompanying text". In simple words, Google scanned each image on the Web, and linked it to the text that comes with the page, and then used the accompanying text as a pseudo meta data. The assumption made by Google was that if the webpage contained the word "dog" on it, then it is likely that the images will contain a dog in them. Further ranking criteria like the number of occurrences of the word, the location(s) in the document where the word was used (header, body, caption, etc.), can refine the returned images and rank them for higher fidelity, improving the hit rate.

To check for the validity of our assumptions, we executed a simple search on Google search engine using their Google Image search. We repeated the experiment several times and visually inspected the results. In all cases, we had a high hit rate, where most of the returned images were indeed dog images. We identified four categories of accuracy:

1) Highly-accurate search results: A picture of a dog
2) Moderately-accurate search result: A picture that includes one or more dogs as well as other objects
3) Poorly-accurate search result: A picture that does not include a dog, but is dog-related. Examples are dog-clinics, dog-related charts, or dog-food
4) Wrong search result: A picture that does not contain any dogs or dog-related items. Figure 3 shows one search result of the keyword "dog" with three wrong search results.

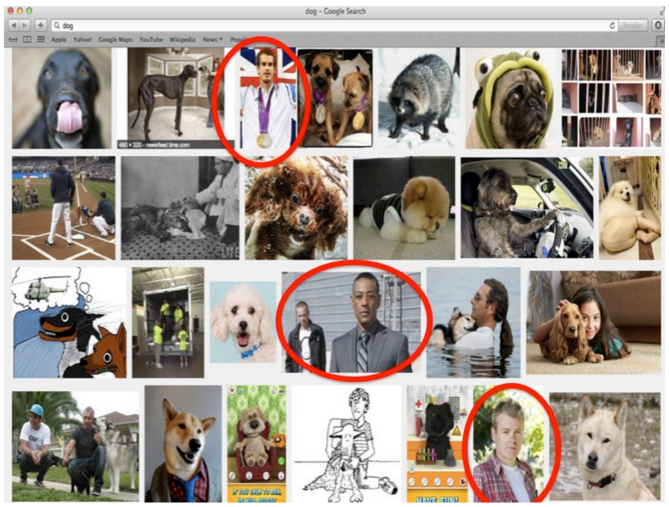

Figure 3: A "dog" keyword search with multiple wrong results

The four categories were observed in our experiment, and we analyzed some of the examples of semantic reasons why the images were identified as dog images.

One of the wrong search results was of a person with the name "Maddog". Nothing else in the whole accompanying text was dog related. Apparently, the method of pseudo meta-text mistakenly identified the contents as dog related. This was a clear case of category 4, "wrong results".

In another case where a person's face was displayed, the title of the accompanying document was " Florida linebacker Antonio Morrison arrested after barking at police dog". Source: New York Daily News, read more: *http://www.nydailynews.com/sports/college/florida-lb-arrested-barking-police-dog-article-1.1404972#ixzz2eWxm5tyB.* This was a clear case of category 3, "poorly-accurate results". For a person looking for dog-related text, this might be relevant. But for a person looking for a dog image, this is not the best result. However, as it is still dog related, we considered it a category 3.

In another example of a person's face image, the document was reporting a criminal incident of a person charged with the attempt to kill a dog: http://www.koin.com/2013/08/05/deputies-man-attached-explosives-to-dog/. This was also a category 3 search result.

In all cases, the search did not produce a highly accurate dog image as desired. The key issue here is that the metadata (the accompanying text in this case) did include related syntax of a "dog" in all cases, and clearly related semantics of "dog"-related issues in the latter two. However, the search still failed to produce a dog image, as would a human search.

Case Study: Smiling Lady painting

In this experiment, we ran a simple search on the keyword "*Smiling Lady Painting*". This is a particularly interesting problem is searching for artwork. As we got back and analyzed the search results, we identified several semantic differences. There were 5 main types of unrelated search results:

Type 1: The "Smiling Lady" was identified as the well-known enigmatic work of the Mona Lisa by Leonardo Da Vinci, and the "Painting" was identified as a noun, for example at: http://www.destination360.com/europe/france/paris/mona-lisa

Type 2: The 'Painting" was still identified as a noun of fine art artifact, but the "Smiling Lady" was taken into a more generic semantics of any painting that has a smiling lay in it. Two examples are:
http://abstract.desktopnexus.com/get/430280/?t=su1g6iveelahljg2avfa68pg2252b21d338f119
and
http://www.ebsqart.com/Artist/Hiroko-Reaney/5939/Art-Portfolio/Gallery/Other-Art/Smiling-Womans-Face/428398/

Type 3: This search result was still in the realm of fine arts, but the word "Painting" was assumed to be a verb, and the search result was of a smiling lady in front of a painting tripod, actually holding a brush and painting something. The example is shown at
http://www.dreamstime.com/stock-photo-smiling-senior-woman-painting-image16998150

Type 4: A more distant semantic search was that of a lady painting a wall. The word "Painting" was seen as a verb, and the actual painting context was that of a wall, not of an artwork, found at:
http://www.123rf.com/photo_5525670_smiling-beautiful-woman-painting-interior-wall-of-home.html

Type 5: The most distant semantics of the search was that of a lady painting her nails.
http://www.dreamstime.com/stock-photos-smiling-woman-painting-her-nails-image1553133

We can see that the same phrase "Smiling Lady Painting" can result in at least 5 completely different semantics. Figure 4 shows all 5 categories as a search result using Google Image search with the keyword "Smiling Lady Painting". As we can see, the use of accompanying text to guess on the contents of an image can work most of the time, but it remains a mechanical workaround, and the success rate goes dramatically down with more complex or less descriptive keywords.

Similarly, as discussed earlier, while the comparison of similar but pixel-imperfect pictures can be manipulated using fuzzy logic and non-traditional database approaches with acceptable results, the problem remains as to fully describe the contents of all contents of a complex picture by a computer in the same intuitive way humans do.

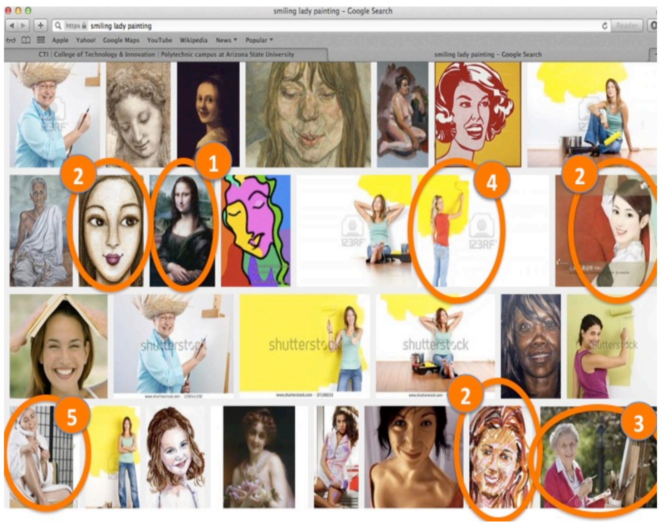

Figure 4: A "Smiling Lady Painting" Search Results

Other approaches to manage unstructured data can be found at Google maps, where they first captured millions of pictures of earth at different heights from ultra high altitudes all the way down to street views, then hired an army of people to manually review, fix, seamlessly connect, and integrate them into one 3-dimensional continuum. In this continuum, you can move in 2-D (north-south and east-west) as well as at the 3rd dimension of zooming in and out. Google further annotated the contents with a gamut of information from street names and directions to related photos and street views. By adding all these integrated structured details, the original pixel-generated images became highly processable by a large number of software application; a perfect example of data reuse.
Amazon.com used a different approach to adding structure to their documents. With the help of millions of users, they annotated the original books contents with additional end-user and system-generated, highly structured contents (e.g. star-ranking, user feedback, related book recommendations, people who bought/looked at this book also bought/looked at those books, etc.).
Further analysis shows an apparent conversion between the two approaches. While Google relied heavily and mainly on hired human force to integrate and annotate data, they later also used the million of users to upload their pictures and used them to further annotate the maps by linking the uploaded pictures to the location where they were take.
Similarly, besides Amazon's heavy reliance on user-created contents on available books, they also included their own compilation of book contents like publisher information, contents scans and chapter examples, and integrated them with the user-created ones.

3 HOW WE SEARCH

We have identified the role of data creator at the "pattern generation" end of Figure 1. Similarly, we can identify the ole of data consumer at the other end, the "pattern reuse" one. We introduce the following data search model to identify three main types of human search behavior.

I) Search to Locate:
This is a common form of search, where the person has already some knowledge about the search object. This pre-existing knowledge can have varying degrees of accuracy related to two main dimensions, the object attributes, and the object location. These two dimensions provide us with four possibilities

- **Clear object and clear location**: The person can have pinpoint accuracy as having seen the object before and has a good idea of where to find it. The search is just trying to locate The object
- **Clear object and vague location**: The person can have accurate information about the object, but has vague idea of where to find it
- **Vague object and clear location**: The person has clear idea about the location of the object. However, she only has abstract idea about the search object itself. The person is vaguely aware of its characteristics or attributes.

The fourth possibility of "vague object and vague location" does not belong to this category as it will have too many unknowns, and the search person's behavior will be greatly different. We believe it

should belong to another category, namely the "search to find", as described below.
The nature of "search to locate" generally has exact keywords, or close enough ones, and will normally end in few iterations. One example is when the user is looking for a particular car model at a particular dealer (clear object and clear location), or when the user is looking for a particular car model at different dealerships (clear object and vague location). The third case of the user looking for all available cars at a particular dealership would resemble the case of vague object and clear location.

II) **Search to Find:**
When we search to find, we do not have a concrete idea of the object we search, or a concrete idea where we might find it (its location). This dual dimensionality of unknowns (unknown object and unknown location) makes the search more complex, and the search domain much larger than the first type. Following the previous example of searching for a new car, this type of search resembles a user looking for a car of any model and make, any year (brand new or used), and at any location (dealership, car resellers, or a private person). We can see the search options in this case are much more than the previous type (search to locate). Moreover, the search behavior of the user can be significantly different. The user will probably look at many different attributes, and will iterate through the search activities several times. The iterations will result in a refinement of the search, and the user will start building an initial idea about both search dimensions: the object, and the location. This type of research will generally take longer than the first type, and may not result in a particular selection; rather it can help build a clear idea of the two search dimensions, or at least one of them, and can eventually evolve into the first type.

III) **Search to Search:**
Unlike the previous two types, in this type of search, the user is not actually searching for any particular object. The user typically has a need, and is not sure how to fulfill it. The user will form an initial idea without having particular object name or attributes. Furthermore, the user will often have no particular search keywords in mind. As the user searches using a wide range of generic keywords, she would get some results that are irrelevant. Those results are typically ignored. Other results would be informative in that it will give the search person an insight into what a search object might be, or even just what some relevant keywords might be. The user behavior in this case would be content-oriented, and not object or location oriented. In content-oriented search behavior, the user will focus on the search result contents and will use them to educate herself about the particular domain, the availability of different objects, the attributes of those objects, the locations, and the like. After several iterations, the user will have identified their needs, and have formed a goal to fulfill those needs. This will result in forming a clear idea about the object to be searched. As we can see, this search will eventually evolve into the previous category of "search to find".

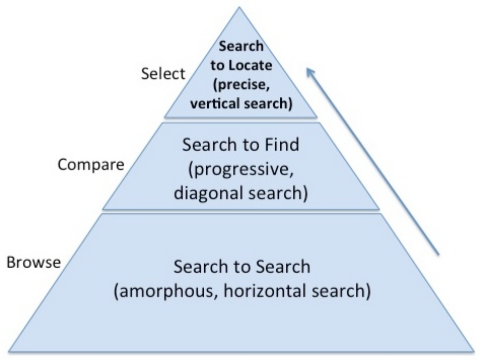
Select
Search to Locate (precise, vertical search)
Compare
Search to Find (progressive, diagonal search)
Browse
Search to Search (amorphous, horizontal search)

Figure 5: Search Activities Model

One case to demonstrate this type of search is when the user is searching for a Christmas present for a family member, but has no specific idea of what that present might be. As we can see, the object is unknown, the location is unknown, but also the search terms and the knowledge of suitable keywords is absent. If the user used "Christmas gift" as a search object, they would be inundated with infinite number of search results with an infinite range of object types, prices, etc. As the user sifts through some of those results, they would start forming an initial idea of what a Christmas present could be. This type of search will mainly help the user get an initial idea about an object, leading to the evolution into the second category, search to find. In Academia, we go through this search to search as we –for example- go through literature review. Initially, we sift through several related and unrelated documents, forming an initial idea without any particular eye on any concrete objects. As we sift through and filter out most of the search results, we refocus on selected papers and start building a concrete idea about search objects, relevant keywords, and specific teams' or person's work.

3.1 Search Lifecycle

While the three search types are stated from more specific to more generic, (categories I through III in the previous section), we can more logically connect them in a progressive lifecycle from the most generic search (browse) to the most specific search (select) as per Figure 5.

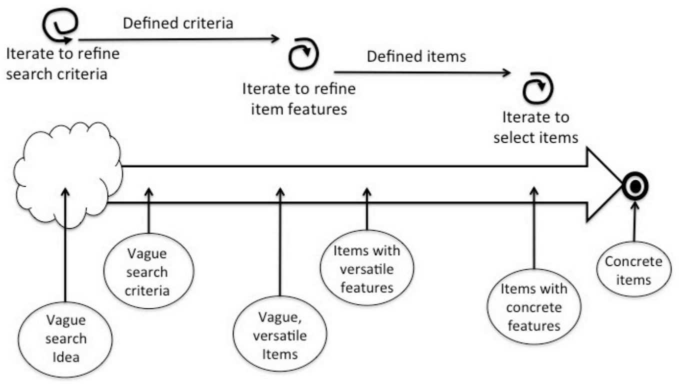

Figure 6: Search Lifecycle

Figure 6 depicts the progression between the three moods of search and some associated artifacts.

4 THE SYSTEM DESIGN

To successfully reuse pattern, we focus on an integrated approach of improving the full lifecycle of data from generation to dissemination to reuse. As we look at the roles associated with the pattern lifecycle (Figure 1), we have identified two main roles: the pattern author, an the pattern user. The fundamental question that arises is: "*Who will be responsible for the dissemination process*".

Pattern Author Role:

From the pattern author perspective, their main role's responsibilities are to formulate their experience and write it own using one of the common patterns format. That will typically include rich, multi-modal document that includes text, images, graphs, screenshots, and possibly a code snippet. This is a significant amount of work, and as we've seen, it is being well-taken care of by pattern authors. However, as we have seen from the literature review above, pattern authors made little effort towards facilitating reuse.

Pattern User Role:

From the user perspective, the biggest challenge for them is to 1) find good patterns, and 2) combine them to generate new design artifacts. Both are challenging tasks for the pattern user. Finding few relevant patterns can be a challenge if we look at the huge number of patterns, mostly represented in reach unstructured, and heterogeneous formats, only suitable for human consumption, but not for machine processing. As discussed earlier, pattern users can only process limited amount of text, so sifting through- and selecting among huge number of patterns become a daunting task. The other challenge is probably equally hard. Combining patterns together in a correct and coherent way would not be easy if it is all presented in isolated patterns or isolated collections of patterns. Research has shown that in several cases, patterns are often applied in a wrong way, or combined in a less-than optimal new design [11] and [16]. Pattern users will have limited capabilities once the patterns are generated in rich document format.

Our approach introduces a new intermediate role of "Pattern Disseminator". This role can be assumed by a human person, or a software agent (Figure 7).

A Human Pattern Disseminator would take the initial pattern artifacts and rewrite them in a structured way that will allow for automatic lookup and integration of patterns. A software agent would interact with the pattern author and –

working at the source- will help pattern authors generate their patterns initially as semi-structured artifacts that render themselves easily to software tools that help the pattern user in finding, filtering, and combining the selected patterns into new design.

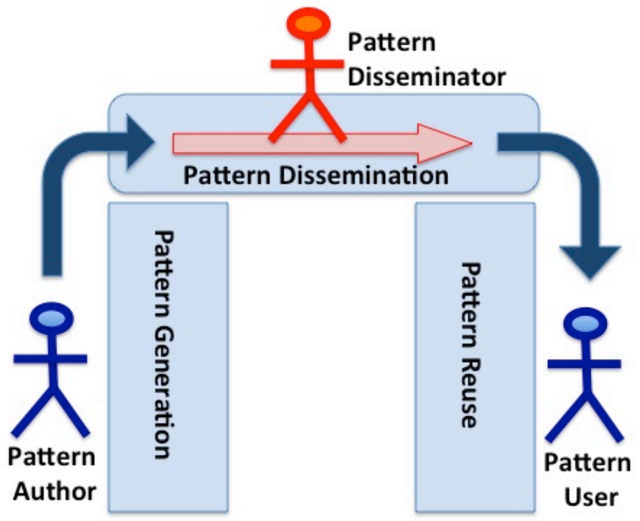

Figure 7: The Role of "Pattern Disseminator"

To validate this approach, we constructed an initial conceptual design with a detailed proof of concept to demonstrate the work. Our system is based on an integrative approach of pattern dissemination that complement the scattered efforts of writing patterns. Following a proof of concept, we built a delivery system in the form of a digital library based on a database of patterns, a transformation logic for any desired algorithms, pattern processing and transformation tools, as well as an interface to offer these functionality to the user. We selected XML as the language to represent patterns for several reasons. XML is becoming the norm for data interchange, and is used extensively in IT systems. Besides the XML language, Several XML-based technologies are emerging to enhance XML-compatible information systems.

The system architecture is shown in Figure 8. The data flow starts at the top right corner in the form of the existing pattern collections. Looking at the data pathways depicted in the figure, we see that the input information is now bypassed from the direct path (from pattern collections directly to user preparations block, marked as A) into the system pre-storage phase, the pattern corpus, and to the rest of the system; marked as B

The User Preparations block, depicted as pathway A, represents the cognitive activity by the user to search for patterns and read through the text, analyze its contents, and figure out how to apply which patterns in the design. This is a fundamental observation in our study. We can evaluate the difference between the two processes, referred to as pathways A and B. If we considered them as representing the dissemination processes between pattern authors and pattern users, we can use the Capability Maturity Model, CMM, of the Software Engineering Institute to briefly evaluate them.
-Process A: This process does not follow a particular approach of dissemination except for relying on users preparations (looking up patterns, understanding them, and applying them in an ad hoc approach). We evaluated this to be a CMM level 1.

-Process B: As suggested by the envisioned system, there is a process in place to help users interact with patterns in a structured way. Moreover, this process relies heavily on feedback, and is constantly changing, as shown in implementing and validating the 7Cs process within the system [17], [18], a CMM level 4

The main modules of the system, as shown in Figure 5 are:

-The Pattern Corpus, a raw collection of patterns before being reformatted and saved in the XMLDB.

-The Data Models, used in the semantics of the XML Rewrites

-The Expert Preparations, used as help in rewriting pattern information to ensure the integrity of the contents and avoid redundancies and inconsistencies.

-The System Process (the 7Cs process), a structured method applied across the system

-The XML Subsystem (XML rewrites, XMLDB and XML semantics/scheme), The core constituent of the system, and the backbone of the three-tiered design.

-The Processing and the Presentation Layers are the middle- and front-tiers respectively.

One of the main aspects of our system is to be able to automatically process the contents of patterns to alleviate the user from this cognitive load. The main purpose is to have a scalable capability to effectively process large number of patterns. Representing patterns in natural language defeats this purpose. XML is much more suitable in this regard due to its machine-readability. It is based on the fundamental concept of automatic processing. Goldfarb [19], the inventor of SGML (the parent, and superset of XML) explains that the vast majority of XML documents will be created by computer programs, and processed by other computer programs, then destroyed. Humans will never see them. The first step is to rewrite patterns in an XML format. A simple XML syntax rewrite (like PLML, Pattern Language Markup Language, [20] can be a small step in this direction, but -by itself- it will not do much good as will be discussed later. To achieve global interoperability, we needed to design the semantics and the behavior modeling behind the XML syntax, according to concrete data models.

Database is a core constituent of the system. An obvious choice for data store is an XMLDB. We implemented part of the system using it. The steps are as follows:

-Phase **1, No-Database System:** By temporarily including the data inside the system (hard-coding the data). We have two major prototypes: In the UPADE project. We developed a systematic approach to glue patterns together, support the integration of patterns at the high design level and automate pattern composition. UPADE [21] generated program elements from an extensible collection of "template" patterns.

In the second prototype, [22], [17], we experimented with prototypes of the interface aspects, and we also hard-coded patterns inside the prototypes. At this stage we were able to test and refine the design and functionality of the system.

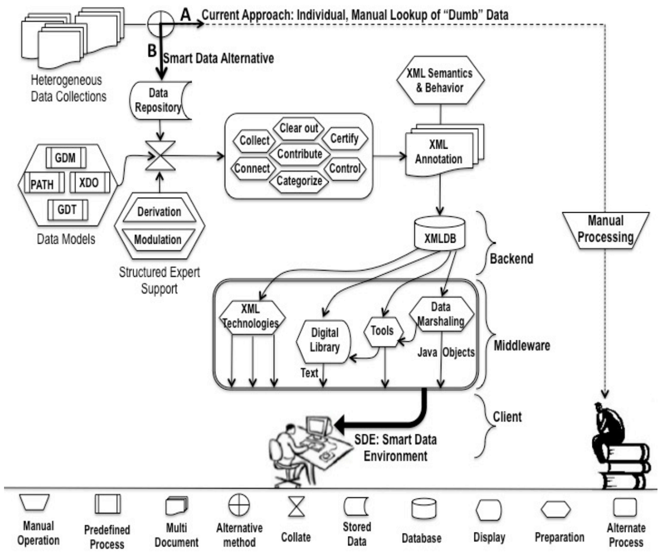

Figure 8: System Conceptual Overview

-Phase 2, Flat-File-Database: To start building an independent database module, external to the system, and to connect it to the rest of the system. We used "XMLSpy" to prototype the XMLDB concept. The XMLSpy allowed us to build an actual, partially functional XMLDB based on flat files as the internal storage medium. The major advantage of this approach is the extreme simplicity of the system, which allowed us to further refine the fundamental concepts of the system without being carried away by the implementation details. The main drawback was its limited scalability and unacceptable performance, as will be explained in step 4.

-**Phase 3, Web-based System:** To implement the functionality of the system as built around external XMLDB on a Web-based environment. We implemented several aspects of the system using the database we have from phase 2. Using 3- tier system architecture (data- logic-, and presentation tiers), DHTML and Web technologies (CGI), we built a web-based distributed system to test our XMLDB. We implemented the system to interact with pattern users by offering a search interface at the presentation tier.

As the user queries the system for some patterns, the system middle tier (at the web server) processes the request by accessing the data tier at the back end, and returning results by dynamically generating a web page with search results.

The 7Cs Process

One of the main aspects of process-oriented approaches is the dependence on a predetermined process. Gaffar [23] describes the "7Cs Process" as a systematic approach to adding structure to rich but ultra heterogeneous data. An established design process instigates quality design by allowing designers to follow structured methods in their activities. In our approach, we emphasized the need for both dissemination and assimilation processes. In this paper, we briefly summarize the 7Cs dissemination process as completely decoupled from any specific assimilation process [8], [11]. This allows it to offer patterns that can be integrated simultaneously in several assimilation processes. "Free patterns" that do not belong to any process at all are hard to integrate in design. Similarly, "proprietary patterns" that are specifically tailored to manually fit one design process using few specific examples defeat the main purpose of pattern generality and abstraction. We see that a pattern can be integrated in several assimilation processes by properly encapsulating its knowledge and presenting its behavior through a well-defined interface. Any assimilation process can then lookup pattern objects and select the appropriate ones using different search criteria. The selected pattern components can then be integrated in new designs or used to generate code fragments. The 7Cs is "a structured process to replace the huge cognitive load of manipulating HCI patterns with a dissemination system of smart patterns" [23]. The 7Cs process identifies both logical and physical aspects of the system. A logical process focuses on *what* actions and activities need to be done. A physical process complements the logical process by specifying the roles associated with the process, and details *who* is going to do what [24], [25], [26]. As part of the pattern reuse problem is associated with missing roles in the dissemination activities (all left to the user), the 7Cs process addresses both how these activities need to be done, and who should be doing each of them. Briefly, the 7Cs process moves gradually from current unplanned generation and use of patterns into building an automated pattern collection. The process comprises seven steps [23]:

Step 1) ***C****ollect: Place Different Research Work on Patterns in One Central Data Repository*

Step 2) ***C****learout: Change from Different Formats/Presentations into One Uniform Style*

Step 3) ***C****ertify: Define Domain Boundaries and Clear Terminology*

Step 4) ***C****ontribute: Receive Input from Pattern Community*

Step 5) ***C****onnect: Establish Semantic Relationships between Patterns Using a Formal Relationship Model*

Step 6) ***C****ategorize: Define Clear Categories for Patterns that Map them into Assimilation Processes*

Step 7) ***C****ontrol – Build Machine Readable Format for Software Tools*

The ultimate goal of the 7Cs process is to allow designers to use the leverage of tools to process large number of patterns efficiently, and to be able to assimilate them effectively into a new design. Patterns that are both human- and machine-readable are the main outcome of the process of pattern *dissemination* and the first step towards *assimilating* them.

DATA MANIPULATION TECHNIQUES

The XML technology is evolving in a rapid pace. Just few years ago, as we started the project, the XMLDB was not supported by Oracle, an industrial norm in databases; or by Apache, an industrial norm in Web servers. The current and future trends are now determined by the new development, as explained in phase 4 of the system development.

-Phase **4** is to use a more advanced, fully-fledged XMLDB to overcome difficulties we discovered in phase 3. XMLSpy internally implemented the XMLDB as flat files. While invisible to the user, this solution is not scalable. It relies on the file structure of the underlying operating system, and is affected by its performance. Besides, inserting to- or deleting from the DB incurs the penalty of rewriting the whole file. The other major obstacle was that some XML technologies we used are still in their early specification stages, and are immature for full implementation. For example, XQuery is the query language supported by XMLDB. We used it to query our XMLDB. While easy to use, it still does not support insertion or deletion, so we had to implement them manually in phase 3.

In phase 4, we are rebuilding the database to be less dependent on immature XML technologies while still offering all XML functionality upfront. The idea is to use newly emerging commercial- or open source systems that are robust enough to support the full functionality of our XMLDB pattern system, while delegating the internal database implementation to the system, be it native- or non-native XML database. Native XMLDB offers an XML interface to the database, and store the database internally in pure XML format. Non-native XMLDB offers an XML interface to the database, similar to the native one, but transforms it internally to relational database, or to a proprietary database format before storing it physically on the hard disk.

Among the hundreds of commercially available XMLDB today, we compared three common alternatives:

1- ORACLE RDBMS with support to XMLDB. An authoritative commercial application that offers a fully functional XMLDB interface, while translating it internally to an RDBMS (a non-native XMLDB)

2- ZOPE, an open source industrial application that implements XMLDB using internal proprietary RDBMS, but provides fully functional XMLDB interface (also a non-native XMLDB).

3- Apache-Xindice XMLDB, a powerful open source native XMLDB technology, supported by Apache (one of the best known and used web servers).

CONCLUSION

Patterns are useful data encapsulation mechanisms that can help UI designers improve the quality of their work by reusing well-known practices of domain experts. For our purpose of building a proof of concept for programmable data contents, they serve as a good example due to their nature of highly unstructured presentation and highly heterogeneous multi-modal contents. They also rely heavily on reuse –at least in principle- allowing us to focus on the amount of reuse as an indicator of success of dissemination. In the pattern community, knowledge dissemination activities focus mostly on generating patterns, with little or on plan on how to deliver them or how to reuse them in new designs. The proliferation of HCI –Human Computer Interaction- patterns is associated with redundancies and inconsistencies that often confuse and overwhelm the user. More traditional approaches like database domain have provided methods to deal with these problems by allowing data to be organized and reused efficiently using normalized database tables and structured queries. We have proposed and built a proof of concept to complement the process of generating patterns by reducing inconsistencies and conflicts, and by providing patterns in a reusable format. We have designed a process that can help in the assimilation and dissemination of patterns. We have implemented a system that -in applying the process- will help novice users receive patterns in an orderly fashion and be able to effectively use them in their design artifacts. The system relies on an underlying XML database and the process will help feed the patterns into the database and rewrite them in a semantically reusable format. The database and the system allow us to add tools to the interface to automatically process patterns in a user-friendly design environment and to connect them to other XML- or UML-based tools.

7 REFERENCES

1. Erickson, T. Lingua Franca for Design: Sacred Places and Pattern Languages, DIS 2000: Proceedings of Designing Interactive Systems, New York, August 17-19, ACM Press, Brooklyn, NY, USA, pp.357-368 (2000).
2. The HCI Patterns. At http://www.hcipatterns.org/patterns.html, accessed on July 8, (2003).
3. Alexander, C., Ishikawa, S., and Silverstein, M. A. Pattern Language: Towns, Buildings, Constructions. New York: Oxford University Press (1977).
4. Stephanidis, C., ed. Human-computer Interaction: Theory and Practice. Part I. Vol. 1. Routledge, (2003).
5. Gamma, E., Helm, R., Johnson, R., and Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley, Boston, Massachusetts, USA. (1995).
6. Gaffar, A., Tridane, A., & Darwish, E. M. (2013). The Failure and Success of Unstructured Data Reuse a Pragmatic Analysis. In The International Conference on Digital Information Processing, E-Business and Cloud Computing (DIPECC2013) (pp. 16-25). The Society of Digital Information and Wireless Communication.
7. Gaffar, A., Tridane, A., & Pribadi, K. (2013). Smart Data: A System for Generating Application-Agnostic Interactive Data. In The Second International Conference on Informatics Engineering & Information Science (ICIEIS2013) (pp. 246-254). The Society of Digital Information and Wireless Communication.
8. Gaffar, A., and Moha, N. "Semantics of a pattern system." STEP 2005 : 211 (2005).
9. Moha, N., Qing,, L., Gaffar, A. and Seffah, A. "Enquête sur les pratiques de tests d'utilisabilité." In Proceedings of the 17th international conference on Francophone sur l'Interaction Homme-Machine, pp. 115-122. ACM, (2005).
10. Gaffar, A., Sinnig, D., Seffah, A. and Forbrig, P. "Modeling patterns for task models. In proceedings of TAMODIA." In 3rd International Workshop on Task Models and DIAgrams for user interface design, pp. 99-104. (2004).
11. Gaffar, A., and Seffah, A. "An XML Multi-Tier Pattern Dissemination System." 740-744 (2005).
12. Gaffar, A., Moha, A. and Seffah, A.. "User-Centered Design Practices Management and Communication." In Proceedings of HCII. (2005).
13. Sinnig, D., Gaffar, A., Seffah, A., and Forbrig, P.. "Patterns, Tools and Models for Interaction Design." MBUI 103 (2004).
14. Moha, N., Gaffar, A., and Michel, G. "Remote usability evaluation of web interfaces." Human Computer Interaction Research in Web Design and Evaluation. P. Zaphiris and S. Kurniawan. Hershey, PA, Idea Group Publishing: 273-289 (2007).
15. Gaffar, A. "Design of a Framework for Database Indexes." PhD diss., Master thesis, Department of Computer Science, Concordia University, (2001).
16. Butler, G., Chen, L., Chen, X., Gaffar, A., Li, J., and Xu, L. "The Know-It-All Project: A case study in framework development and evolution." Domain Oriented Systems Development: Perspectives and Practices: 101-117 (2002).
17. Gaffar, A., Sinnig, D., Javahery, H. & Seffah, A., (2003). MOUDIL: A Comprehensive Framework for Disseminating and Sharing HCI Patterns. Position Paper in ACM CHI 2003 Workshop: Perspectives on HCI Patterns: Concepts and Tools, Ft. Lauderdale, Florida, (2003).
18. Metzker, E., Seffah, A. and Gaffar, A., (2003). Towards a Systematic and Empirical Validation of HCI Knowledge Captured as Patterns. Proceedings of HCI International, the 10th International Conference on Human-Computer Interaction, Crete, Greece, vol. 1, Theory and Practice, LEA, (2003).
19. Goldfarb, C. F., and Prescod, P., (2004). The XML Handbook, 5th edition, Prentice Hall PTR, Upper Saddle River, New Jersey, USA, (2004).
20. Fincher, S., & Finlay, J.,. CHI 2003 Report: Perspectives on HCI Patterns: Concepts and Tools; Introducing PLML. Journal of Interfaces, No. 56, September BCS HCI Group publishers, Canterbury, Kent, UK, pp. 26-28. (2003)
21. UPADE,. User Patterns Automated Design and Engineering at http://hci.cs.concordia.ca/www/hcse/projects/, accessed on Mai 12, 2003 (2000).
22. MOUDIL Project, at http://hci.cs.concordia.ca/moudil/, accessed on June 3, (2003).
23. Gaffar, A. The 7Cs: An Iterative Process for Generating Pattern Components, electronic proceedings of HCI International, The 11th International Conference on Human-Computer Interaction, July 22-27, 2005, Las Vegas, Nevada, USA.

24. Hoffer, J. A., George, J. F., Valacich, J. S., Modern system analysis and design. 3rd edition, Prentice Hall (2002).
25. Whitten, J. L., Bentley, L. D., & Dittman, K., C.. System analysis and design methods,5th edition, McGraw Hill Irwin (2001).
26. Ackerman, Mark S., & Thomas W. Malone.. Answer Garden: A tool for growing organizational memory. Proceedings of the ACM Conference on Office Information Systems: (pp.31-39) (1990).