Addressing Quality Requirements in GIS Architectures

Ibrahim HABLI, Lebanon and Tim KELLY, United Kingdom

Key words: GIS, Software Architecture, Requirements, Quality, Software Design

SUMMARY

Several surveys have shown that about 80% of data in local governments and utility companies are geographical and 1.5% to 2% of the annual budget of such sectors is spent on collecting geographic information. Therefore, such reliance on geographic information has necessitated the development of high quality Geographic Information Systems (GIS) with system qualities such as performance, availability and maintainability. However, in practice, attention is paid to GIS functionalities, rather than such qualities, thereby leading to inefficient and unreliable GIS systems that exceed cost and time allocations.

Software architecture design is the first, and most fundamental, stage that addresses the achievement of quality requirements for software-intensive systems such as GIS. Hence, this paper presents and analyzes a systematic framework for the architectural design of a GIS. It is based on the establishment of mappings between quality requirements and architectural decisions by providing the mappings' rationale, benefits, tradeoffs, and risks. Such a framework enables the early discovery of the critical technical decisions involved in achieving GIS system qualities.

Addressing Quality Requirements in GIS Architectures

Ibrahim HABLI, Lebanon and Tim KELLY, United Kingdom

1. INTRODUCTION

An increasing reliance on geographic information demands GIS systems that exhibit qualities such as efficiency, reliability, and security. However, the complexity and large volume of geographic information has made it difficult to develop GIS systems with the aforementioned qualities. In practice, GIS quality requirements are often not systematically captured and documented. Instead, they are usually documented in an ambiguous, general and incomplete way. For, example, it is common to find, in GIS requirements documents, statements such as: "the system shall be portable", "the system shall be highly secured", or "GIS operations shall be efficient". The main problem with such statements is that they are not quantifiable. In other words, such statements are unfalsifiable, in the sense that there are no feasible means to assess whether the system has met its quality requirements or not.

Therefore, the development of GIS systems, like most software-intensive systems, requires a framework that ensures that quality requirements are unambiguously documented. The system can then be systematically designed and evaluated against these requirements. Although no software engineering activity could alone ensure the achievement of quality requirements, software architectural design is recognised as having a substantial impact on the satisfaction of such requirements (Clements 2002). The architectural design decisions are influenced by the necessity of achieving the required qualities. In other words, the software architecture is "shaped" by the quality requirements. Despite such importance, architecture design is insufficient alone if attention is not paid at later stages to the details of the design, implementation and testing.

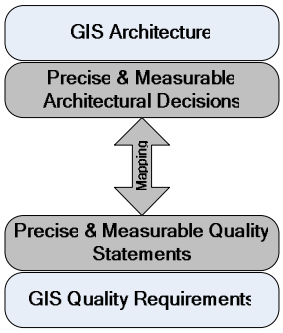

Figure 1: Mapping Between Qualities and Architecture

Not only does the design of software architectures stress the importance of concrete and quantifiable quality requirements, but also precise and concrete architectural design decisions. A mapping between quality requirements and architectural decisions can be established (Figure 1). Each architectural decision can be linked to one or more concrete

quality statements. Such a linkage can be established by providing the rationale, benefits, tradeoffs, and risks that are involved in the satisfaction of the quality statements by the architectural decisions.

2. METHODOLOGY

This paper describes and evaluates a systematic and pragmatic approach to the documentation of quality requirements and the architectural design of a GIS. Specifically, two architectural development approaches are described:

- *Quality Attribute Scenarios:* a documentation technique which articulates the measures by which a quality attribute is gauged (Barbacci 2003)
- *Attribute Driven Design Method (ADD):* a recursive approach to the design of architectures based on the quality attributes that the software has to achieve (Bass 2001)

Quality requirements documentation is discussed in Section 0. ADD is introduced in Section 0. In Section 0, we describe how GIS quality attribute scenarios can be addressed by the application of ADD. The success of this application is evaluated in Section 0. Finally, conclusions are drawn in Section 0.

3. GIS QUALITY REQUIREMENTS DOCUMENTATION

Software requirements can be categorized into two types: functional and quality. Functional requirements define what exactly the system should do, while quality (or non-functional) requirements define how well the system should do it (Kotonya 98). Quality requirements are vital for software-intensive systems such as GIS. For example, even if GIS has met the required functionality, it would be (1) ineffective if its processing misses the acceptable deadline (2) unreliable if it is not available when it should be and (3) unusable if it is difficult to understand. The purpose of this section is to codify GIS quality requirements, especially those needed for enterprise solutions requiring concurrent users working at distributed geographic sites.

The codification of GIS qualities requires 'fitness' criteria in order to become measurable, and consequently testable. Without quantification, it is impossible to verify that the architecture and implementation have met its quality requirements (Robertson 1999).

In this paper, documenting GIS qualities is carried out based on the Software Engineering Institute's (SEI) quality attribute scenario approach. The SEI has based its quality attribute specification process on three observations about quality attributes (Bass 2003):

- Many quality attributes (such as usability and portability) lack an accepted definition
- There is often difficulty in deciding the quality to which a certain characteristic belongs. For example, is portability considered an aspect of modifiability or an independent quality attribute?
- There is a lack of a standard common vocabulary for describing and reasoning about quality attributes

The first two observations are tackled by the use of quality attributes scenarios. These scenarios are documented in the following terms: source of stimulus, stimulus, environment, artefact, response, and response measure. As for the lack of common vocabulary, the SEI has recommended explaining the fundamental concepts behind each attribute to overcome possible ambiguities.

A fundamental aspect of quality attributes is that they usually overlap (Bass 2001). For example, the time needed to carry out a certain task can be considered a feature of both usability and performance. Portability, reusability and integrability might be considered as separate quality attributes or as aspects of modifiability. Therefore, the use of scenarios mitigates the consequences of quality attributes intertwining by codifying the response measure of each scenario. As a result, the focus of the GIS design can be on the achievement of specific scenarios regardless of the quality attribute they fall within. To illustrate the application of the approach, this paper focuses on two of the key quality attributes for GIS: performance and modifiability. (There are, of course, other quality attributes of interest for GIS. However, it is not possible to describe these fully within the confines of this paper.) The following two subsections present a codification of these quality requirements using concrete quality attribute scenarios.

3.1 Performance Quality Requirements

Performance is primarily concerned with the efficiency in which a system processes information within the required constraints. The following scenarios elaborate on two common GIS performance requirements.

3.1.1 PRF_EDITING: Editing a Geographic Feature

Source:	GIS editor/external system/internal source
Stimulus:	A request to change the spatial aspects of a geographic feature
Artefact:	GIS system (data source)
Environment:	Normal operation
Response:	The system processes the edit, updates the data source, and notifies the requester about the change
Response Measure:	Latency (time between the occurrence of the stimulus and the response)

Table 1: PRF_EDITING: Editing a Geographic Feature

Editing in GIS results in an execution overhead, given that a single change of a geographic feature may trigger a series of different computations. This is due to the complexity of the relationships between geographic data themselves and their association with tabular data.

3.1.2 PRF_RETRIEVE: Retrieve Data

Source:	GIS viewer/external system/internal source
Stimulus:	A request to retrieve data from the data source
Artefact:	GIS system (data source)

Environment:	Normal operation
Response:	The system processes the request, and retrieves the requested data
Response Measure:	Latency

Table 2: PRF_RETRIEVE: Retrieve Data

The process of retrieving data in GIS involves spatial and tabular queries. This increases the number of components that need to participate in producing a single query, especially in GIS systems that have controlled geometric topologies or networks.

3.2 Modifiability Quality Requirements

Modifiability is concerned with the ability of a system to be changed after deployment (Bass 2000). It is highly expected for a GIS system to change after deployment due to the frequent modifications in its underlying technologies and operational environment. Moreover, GIS are typically required to adapt to scalability requirements with respect to improving its functionality, increasing number of users, and adding new data sources. The following scenarios define three common GIS modifiability requirements.

3.2.1 MDF_FORMAT: Change GIS Data Format

Source:	GIS administrator
Stimulus:	The administrator requests to convert the data format of a portion of the GIS data
Artefact:	GIS system (Data source)
Environment:	Runtime
Response:	The required format is converted with no data inconsistency
Response Measure:	Number of elements affected/ programming effort/ data loss

Table 3: MDF_FORMAT: Change GIS Data Format

Most GIS systems need to interact with different data formats. GIS users, mainly administrators, are responsible for the data conversion process. Hence, this scenario measures the ability of a GIS system to work with different GIS data formats.

3.2.2 MDF_COMPONENT: Add GIS Component

Source:	GIS programmer
Stimulus:	The programmer wants to add a new component to the system
Artefact:	GIS system
Environment:	Compile Time
Response:	The component is added to the system
Response Measure:	Required time/number of elements affected

Table 4: MDF_COMPONENT: Add GIS Component

GIS systems are usually expected to have a long lifetime. Therefore GIS architectures should provide flexible mechanisms for adding new components. The ability to easily integrate

different components is sometimes referred to as integrability (Bass 1998). Spatial data management, visualization, and analysis components might be developed by different teams, and at different stages, and therefore the architecture should provide a proper framework that easily integrates such components to the system.

3.2.3 MDF_SW: Interface GIS with an External Software System

Source:	GIS programmer/system administrator
Stimulus:	Enable GIS to communicate with an external software system
Artefact:	GIS system
Environment:	Compile Time
Response:	The GIS system is interfaced with an external software system
Response Measure:	Required time/number of elements affected

Table 5: MDF_SW: Interface GIS with an External Software System

GIS systems are required to communicate with various software systems, typically legacy systems, in order to share data and analysis results. Therefore, GIS architectures need to provide the required mechanism for such interface with a minimum integration effort.

In Section 5, the GIS quality scenarios described above are addressed in the architecture design using ADD. However, it is unfeasible for a GIS architecture to achieve all its required quality attributes without tradeoffs being made. Hence, it is important that the tradeoffs be made based on a predefined prioritization mechanism that serves the GIS system objectives.

4. ATTRIBUTE DRIVEN DESIGN METHOD (ADD)

ADD is a recursive approach to software architecture design based on the quality attributes the software needs to achieve (Bass 2001). ADD is based on the interrelation between software architecture and quality attributes. It depends on two main factors:

- *Architectural drivers***:** comprising the architecturally significant requirements, primarily quality attributes. Recognizing such drivers early on during the architecture design centers the design around a limited number of factors that highly affect the quality of the entire system. Moreover, those drivers should be explicitly documented in a way that enables avoiding ambiguity and incompleteness.
- *Relationship between architectural patterns and quality attributes***:** an architectural pattern addresses one or more quality attributes. Nevertheless, it might also have side effects on other quality attributes. Therefore, expressing architectural patterns with respect to their benefits and side effects is essential to the design and assessment of the architecture.

Figure 2: ADD

ADD accepts architectural drivers as inputs and then produces conceptual architecture as an output (Figure 2). Therefore, choosing the requirements that are significant to the architectural design process is a prerequisite for applying ADD. On the other hand, the output of the method is a conceptual model that is used to instantiate concrete architectural models such as the physical and behavioural views.

ADD processes its inputs, i.e. architectural drivers, in a recursive manner. The following steps take place at each iteration:

- Choose one design element: the first design element is the entire system, which is then decomposed into conceptual subsystems. Subsequently, each conceptual subsystem is decomposed into conceptual components.
- Choose the architectural drivers that the decomposition should address.
- Choose architectural patterns that decompose the design element into components that address the selected architectural drivers.
- Assign functionality to each of the design elements (resulting from the decomposition) based on architecturally significant functional requirements (typically modelled as use cases). Function allocation occurs in multiple architectural views.
- Verify that the decomposition has addressed the selected architectural drivers. Then refine the quality scenarios and use cases based on the decisions made during the current decomposition and make them constrain the next stages of decomposition.

Several architectural design methods, such as the Siemens design method (Hofmeister 1999) and functionality-based architectural method (Bosch 2001), provide systematic techniques for the design of software architectures with respect to the explicit definition of the architectural drivers, the design process, and the expected outputs. However, such design methods focus on diverse types of drivers that, unlike quality attributes, might not play a decisive role in shaping the GIS architecture. Hence, as far as GIS architectures are concerned, ADD is more suitable since it is predominantly based on the achievement of quality attributes, which are the key factors in the design of GIS architectures.

5. ACHIEVING GIS QUALITY REQUIREMENTS USING ADD

This section illustrates how the GIS quality attribute scenarios specified in Section 0 can be systematically addressed by carrying out software architecture design according to the steps of ADD. Two decomposition stages are adequate to achieve such quality scenarios. Each design stage is decomposed using several architectural design strategies. Such strategies encompass proven architectural unit operations (Bass 1998), tactics (Bass 2003), styles (Shaw 1996) and design patterns (Gamma 1995).

5.1 First Level of Decomposition

The design element that is handled at this stage is the system as a whole. Figure 3 shows the module view of the first decomposition.

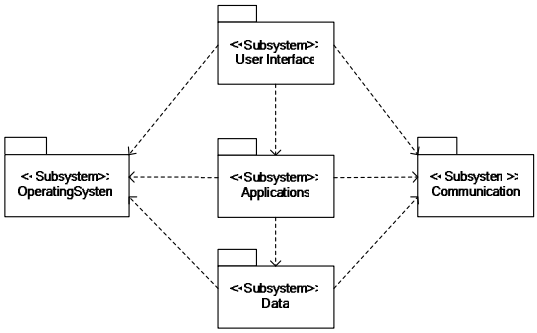

Figure 3: First Architectural Decomposition

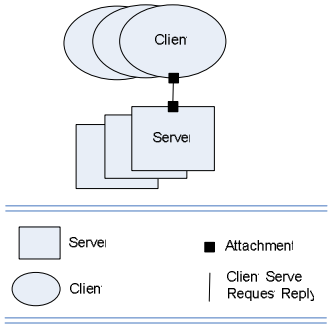

Figure 4: Client-Server Style

This architectural decomposition is based on the following design decisions:

- *Maintain semantic coherence tactic (Bachmann 2003):* applying this strategy enables localizing expected modifications in the OperatingSystem, Communication, User-Interface, Applications, and Data subsystems. Such encapsulation aids each subsystem to provide its services without dependence on external modules. The MDF_COMPONENT scenario is addressed by this tactic.
- *Published interfaces tactic (Sondheim 1999):* The five subsystems shown in Figure 3 should interact through stable interfaces. Therefore, changing the data format in the Data subsystem, for example, does not affect the Applications subsystem as long as the interfaces between the two subsystems are preserved. The MDF_FORMAT is addressed by this tactic.
- *Client-Server style (Shaw 1996):* this style separates the system into two main runtime elements: clients and servers. It improves performance by enabling clients to utilize the high performance processing capability of the servers (Clements 2003). Figure 4 shows a Component-and-Connector (C&C) view of the client-server style. The performance quality scenarios are addressed by this style.
- *Minimize interaction between clients and servers tactic (Tanin 2002):* client applications should only initiate communication with servers in case they do not have enough data stored locally. For example, some clients should be more than simple GIS viewers. They should have the capability to locally execute some operations such as zooming or simple querying without requesting them from servers. This tactic improves the performance of the communication system by reducing the number of requests between clients and servers. The PRF_RETRIEVE and PRF_RT_UPDATE scenarios are addressed by this tactic.

A refinement of the client-server architecture might be required if many users need to work on large spatial data sets for a long period of time (Tanin 2002). In such a case, downloading a large portion of a data set may increase the load on the server, which would in turn lead to a state where clients are idle waiting to be served by a slow server. One solution would be to allow peer-to-peer communication between clients. The system will still have a client-server environment yet the server maintains information about data that are already downloaded to client applications. Hence, when a server is in an overload state, it can start using active clients to operate on its behalf. In other words, clients, that have already downloaded a portion of the GIS data, can temporary act as servers to other clients that require the same

GIS data. This in turn minimizes the demand for resources from the main server. In summary, the server is still the main source of data; yet, the system only switches to the peer-to-peer scheme to balance its load when it is in an overload state.

Even though such an approach (peer-to-peer) improves the overall performance of the GIS system, it complicates the system's modifiability. In other words, a tradeoff between modifiability and performance has to be made. The server in the abovementioned approach is required to keep track of the status of its clients, especially their downloaded data. This in turn increases coupling by exposing the internal structure of the clients to the servers. Such a side effect does not exist in the traditional client-server architecture since servers do not need to be acquainted with the identity or number of clients that access them at run-time (Shaw 1996).

5.2 Second Level of Decomposition

This design stage further decomposes the Data, Applications, and User-Interface subsystems.

5.2.1 Data Subsystem Decomposition

Geographic data is the centre of the GIS technology. Hence, the performance and portability of the GIS data are addressed by the architectural decomposition of the Data subsystem.

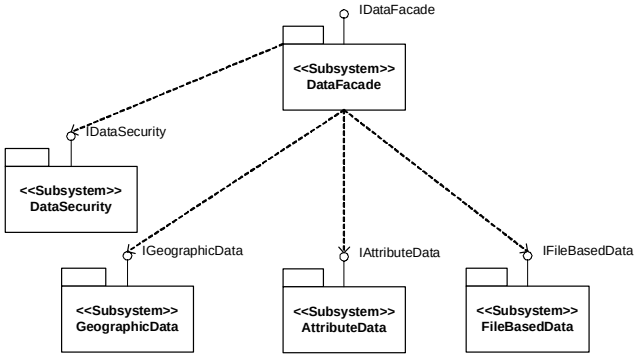

Figure 5: Data Subsystem in Module View

Figure 5 shows the module view of decomposition of the Data subsystem. The design decisions made at this stage are:

- *Data accessor design pattern (Nock 2004):* the physical data access of each category of GIS data (geographic, attribute, and file-based) is encapsulated in a separate subsystem, exposing only the logical operations. Such encapsulation enables changing the data source and format without side effects on the applications that access the GIS data. The MDF_FORMAT scenario is addressed by this pattern.
- *Façade design pattern (Gamma 1995):* the DataFaçade subsystem provides a unified interface to the internal design elements of the Data subsystem. Introducing the façade subsystem encapsulates lower-level interfaces (IGeographicData, IAttributeData, IFile-BasedData, and IDataSecurity) within a cohesive high-level interface (IDataFaçade).

This in turn supports understandability, maintainability and an extra level of portability. The MDF_FORMAT scenario is addressed by this pattern.

- *Concurrency tactic (Bachmann 2003):* concurrency enables the processing of requests in parallel. One of the techniques used for achieving concurrency is multithreading (Chen 2000). Multithreading enables processing different requests on different threads thereby efficiently utilizing the available resources. Not only does distributing information processing on several threads improve the response time, but it also improves the robustness of the system. The PRF_EDITING and PRF_RETRIEVE scenarios are addressed by this tactic.
- *Caching tactic (ESRI 2004):* reusing data and computation by increasing the memory cache leads to significant performance improvement. The PRF_EDITING and PRF_RETRIEVE scenarios are addressed by this tactic.
- *Increase hardware resources tactic (Bachmann 2003):* increasing hardware resources such as adding CPUs, increasing memory, and improving network speed reduces latency. The application of this tactic depends on the budget allocated to hardware resources (tradeoff between performance and cost). The PRF_EDITING and PRF_RETRIEVE scenarios are addressed by this tactic.

Applications that need to access the GIS data are not required to identify the internal structure of the Data subsystem. They only need to recognize the IDataFaçade interface that is responsible of the carrying out of all data editing and retrieval requests. Although this approach improves the portability of the GIS data, an application can still directly access the IDataSecurity, IAttributeData, IGeographyData and IFileBasedData interfaces since the façade design pattern does not place restrictions on such access. This in turn complicates maintainability. Although such direct access improves performance, the performance gain will be insignificant if scheduling policies are properly defined.

5.2.2 Applications Subsystem Decomposition

The drivers considered in this decomposition are similar to those considered in the decomposition of the Data subsystem. There exists interdependence between GIS data and GIS applications and hence the overall quality of the GIS system depends on both of them. For example, if the application server fails, GIS clients will not be served even if data servers are under normal operation and vice versa.

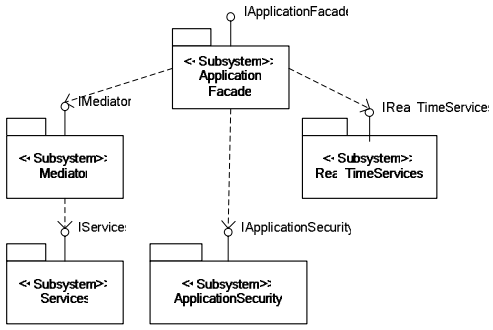

Figure 6: Applications Subsystem (module view)

Figure 6 shows the decomposition of the Applications subsystem. The design decisions made at this stage are:

- *Mediator design pattern (Gamma 1995):* adding a design element (Mediator subsystem) that encapsulates the interaction between clients and GIS services supports modifiability due to the loose coupling such an element promotes. In addition, a mediator enables clients and GIS services to be modified with no side effects on each other. The scenario addressed by this pattern is MDF_SW.
- *Use geographic information standards tactic (Grønmo 2001):* the use of standards reduces the need for data and service conversion and hence improves performance in addition to modifiability. One of such standards is GML (Geography Markup Language) (OpenGIS 2004). GML has been developed by the Open GIS Consortium (OGC), an international industry consortium of 258 companies, in an attempt to standardize GIS data formats. The MDF_FORMAT scenario is addressed by this tactic.
- *Separation unit operation (Kazman 1994):* the separation of real-time from non-real-time services enables direct access to real-time operations without any delay imposed by the Mediator subsystem. In addition, such separation allows the deployment of different technologies since, generally, the technology used to implement real-time services is different from that deployed for non-real-time services. However, this separation has a major drawback on portability since the GIS real-time operations will not be portable (no mediation). A change to the data format or application logic of the real-time operations would have a negative impact on maintainability, especially with respect to the number of modules affected. The PRF_EDITING and PRF_RETRIEVE scenarios are addressed by this tactic.

With respect to the portability of GIS services, the introduction of the Mediator subsystem improves modifiability as it decouples GIS clients from non real-time GIS services. Nevertheless, deploying a mediator may degrade performance (Schmidt 2000) mainly because of the overload that would result from the additional indirection and transformation of user requests. A solution that can lessen the overhead of such additional indirection is the efficient use of geographic information standards. Recent efforts by the OpenGIS Consortium (OpenGIS 2004) have resulted in several specifications for GIS standards. Therefore, the mediator might provide adapters (Gamma 1995) for a number of GIS standards. Then, any GIS client that needs to interact with the server has either to be compatible with such standards or implements its own adapter. This would improve the performance of the GIS server by moving some of the adapters that offer non-standard GIS formats to the client side.

5.2.3 User-Interface Subsystem Decomposition

The user-interface is frequently changed during the lifetime of a GIS. Such changes might arise in the event of an adjustment to the existing services or a need to add new clients.

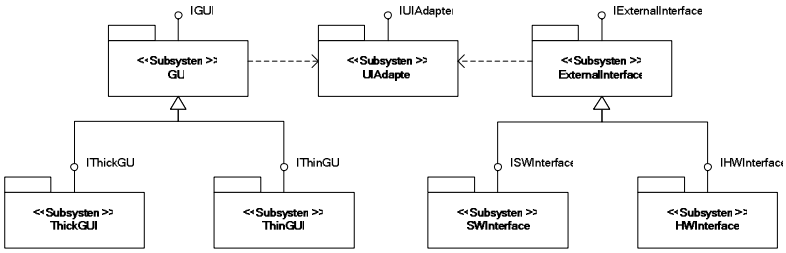

Figure 7: User-Intrface Subsystem

Figure 7 shows the module view of the decomposition of the User-Interface subsystem. The design decisions made at this stage are:

- *Generalization style (Clements 2003):* the GIS user-interface is expected to change after deployment. Nevertheless, in general, there are certain functionalities that are rarely changed such as basic querying and mapping. Those functionalities are the reason behind choosing the generalization style since the parent subsystem should capture the common user-interface functionalities that are shared between many GIS applications. On the other hand, the user-interface variations should be implemented in the children subsystems. The MDF_SW and MDF_COMPONENT scenarios are addressed by this style.
- *Adapter (or wrapper) design pattern (Gamma 1995):* GIS clients interact with the Applications subsystem through predefined interfaces. However, if any client is incompatible with those interfaces, it needs to use an adapter to communicate with the interfaces provided by the Applications subsystem. The MDF_SW scenario is addressed by this style.

The generalization style in this decomposition supports extensibility and evolution. User-interface extensions are introduced by adding, changing or modifying a child subsystem. On the other hand, any change to a parent subsystem is automatically reflected in its children and this, in turn, supports the user-interface evolution.

6. EVALUATION

Addressing quality requirements using a methodological architecture design process, such as ADD, serves as a skeleton for achieving the design purpose. The purpose of this section is to discuss the suitability of such a process for the design of GIS architectures.

6.1 GIS Quality Requirements Analysis

Articulating GIS qualities using quality attribute scenarios (Section 3) provides the following benefits:

- *Understandability***:** documenting quality attributes in scenarios, each comprised of source, stimulus, artefact, environment, response and response measure, unambiguously defines the factors that control the achievement of the required qualities. Therefore,

quality attribute scenarios relieve GIS architects of the burden of worrying about the interrelations between quality attributes. For example, data integrity could be categorized as a security or an availability aspect; yet, such categorization is insignificant since data integrity is specified using a concrete scenario, which precisely defines its characteristics irrespective of the quality attribute data integrity belongs to.

- *Precision***:** explicitly providing the yardsticks, especially response and response measure (against which the achievement of quality attributes can be gauged), offers specific means for assessing GIS architectures.
- *Traceability:* decomposing each quality attribute into scenarios, each describing a specific aspect of a quality attribute, enables traceability of how that attribute has been addressed during the architectural design and evaluation.

6.2 GIS Architecture Design

This subsection analyzes the GIS architecture with respect to the design process and the completeness and consistency of the architectural artefacts.

6.2.1 Attribute Driven Design Method

By defining the architecture recursively, ADD simplifies the architectural design process and forces the systematic consideration of quality attributes alongside traditional functional requirements. It is worth noting that the application of ADD will not necessarily result in a high quality GIS architecture. The quality of the architecture depends on the quality of (1) the requirements documentation, (2) the architectural decisions and (3) the mapping between the quality attribute scenarios and the architectural decisions.

Last but not least, ADD provides a skeleton for the employment of proven architectural decisions (Section 0) and the generation of well organized documentation (Section 0).

6.2.2 Architectural Design Decisions

Software architecture is not only a composition of components and connectors, but also a composition of architectural design decisions (Bosch 2004). Applying ADD illustrates such an assertion by revealing the vital role that architectural design decisions play in the establishment of the GIS architecture. In the end, the architectural views that depict the structure and behaviour of the architecture are nothing more than the results of applying a number of specific architectural design decisions.

6.2.3 Design Documentation

A key problem with software architecture is that information describing the design of the architecture can be lost if not carefully recorded (Bosch 2004). Losing such information increases the risks of carrying out an architectural modification that might violate an architectural assumption or design decision. The application of ADD produces well organized architectural documentation. This documentation comprises GIS quality attribute scenarios, a record of architectural design decisions applied, the resultant architectural views and

underlying design rationale. Inevitably, GIS architectures will change after their initial design. Any change, whether for adding new functionality or quality scenarios or enhancing exist ones, should be carried out in light of the abovementioned architectural information. Properly recording such information aids in the management of the evolution of the GIS architectures.

7. CONCLUSIONS

This paper presents a framework for the documentation of quality attributes and the architectural design of GIS. The documentation of GIS quality attributes using concrete scenarios encourages the specification of such attributes in a precise and assessable way. Moreover, the architectural design process, using ADD, stresses the importance of employing well-known and proven architectural design decisions for the achievement of the desired GIS system qualities. Addressing such qualities, at the architectural stage, enables the early discovery of the critical technical decisions involved in achieving high quality GIS systems.

REFERENCES

(Bachmann 2003) Bachmann, Felix; Bass, Len; Klein, Mark; *Deriving Architectural Tactics: A Step Toward Methodical Architectural Design.* Technical Report (CMU/SEI-2002-TR-025) Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. 2003.

(Barbacci 2003) Barbacci Mario; Ellison, Robert; Lattanze, Anthony; Stafford, Judith; Weinstock, Charles; Wood, William. *Quality Attribute Workshops (QAWs)*, Third Edition. Technical Report (CMU/SEI-2003-TR-016) Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. 2003.

(Bass 1998) Bass, L.; Clements, P.; Kazman, R. *Software Architecture in Practice*. 1st ed. Reading, MA: Addison-Wesley, 1998.

(Bass 2000) Bass, Len; Klein, Mark; Bachmann, Felix. *Quality Attribute Design Primitives*. (CMU/SEI-2000-TN-017) Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. 2000.

(Bass 2001) Bass, Len; Klein, Mark; Bachmann, Felix; *Quality Attribute Design Primitives and the Attribute Driven Design Method*. 4th International Workshop on Product Family Engineering. Bilbao, Spain, 3-5 October 2001.

(Bass 2003) Bass, L.; Clements, P.; Kazman, R. *Software Architecture in Practice*, 2nd ed. Reading, MA: Addison-Wesley, 2003.

(Bosch 2001) Bosch, Jan; Florijn, Gert; Greefhorst, Danny; Kuusela, Juha; Obbink, Henk; Pohl, Klaus. *Variability Issues in Software Product Lines*. 4th International Workshop on Software Product-Family Engineering. 2001.

(Bosch 2004) Jan, Bosch. *Software Architecture: The Next Step*. European Workshop on Software Architecture. St. Andrews, UK. 2004.

(Chen 2000) Chen, Shu-Ching; Wang, Xinran; Rishe, Naphtali; Allen Weiss, Mark. *A High-Performance Web-Based System Design for Spatial Data Accesses*. Proceedings of the eighth ACM international symposium on Advances in geographic information systems. 2000.

(Clements 2002) Clements, Paul; Kazman, Rick; Klein, Mark. *Evaluating Software Architectures: Methods and Case Studies*. Addison Wesley, 2002.

(ESRI 2004) ESRI. *Metadata and GIS*. ESRI White Paper, 2002, < www.esri.com/library/whitepapers/ pdfs/metadata-and-gis.pdf >, Accessed July 24, 2004.

(Gamma 1995) Gamma, E.; Helm, R.; Johnson, R.; & Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison Wesley, 1995.

(Grønmo 2001) Grønmo, Roy. *Supporting GI standards with a model-driven architecture.* Proceedings of the sixth ACM international symposium on Advances in geographic information systems. 2001.

(Hofmeister 1999) Hofmeister, Christine; Nord, Robert; Soni, Dilip; *Applied software architecture.* Addison Wesley, 1999.

(Kazman 1994) Kazman, Rick; Bass, Len. *Toward Deriving Software Architectures From Quality Attributes*. (CMU/SEI-94-TR-10). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University. 1994.

(Kotonya 98) Kotonya, G.; Sommerville, I. *Requirements Engineering: Processes and Techniques*, J. Wiley, c1998.

(Nock 2004) Nock, Clifton. *Data Access Patterns Database Interactions in Object-Oriented Applications*. Addison-Wesley. 2004.

(OpenGIS 2004) Open GIS Consortium, Inc. <http://www.opengis.org > Accessed July 13, 2004.

(Robertson 1999) Robertson, Suzanne; Robertson, James. *Mastering the Requirements Process*. Addison-Wesley, 1999.

(Schmidt 2000) Schmidt, Douglas; Stal, Michael; Rohnert, Hans; Buschmann, Frank. *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*. Volume 2. John Wiley & Sons, Ltd. 2000.

(Shaw 1996) Shaw, Mary; Garlan, David. *Software Architecture Perspectives on an Emerging Disciple*. Prentice Hall, 1996.

(Sondheim 1999) Sondheim, M. Gardels, K; Buehler, K. *GIS Interoperability*. Edited by Longley, Paul; Goodchild, Michael; Maguire, David; Rhind, David *Geographical Information Systems: Principles and Technical Issues*. Volume 1, 2nd edition. Wiley, 1999

(Tanin 2002) Tanin, Egemen; Brabec, František; Samet, Hanan. *Remote access to large spatial databases.* Proceedings of the tenth ACM international symposium on Advances in geographic information systems. 2002.

CONTACTS

Ibrahim Habli
KHATIB & ALAMI (CEC)
P.O.Box: 14-6203 Beirut 1105 2100
LEBANON
Tel. + 961 1 843843
Fax + 961 1 844400
Email: Ihabli@kacec.com
Web site: www.khatibalami.com

Tim Kelly
Department of Computer Science,
University of York
Heslington, York YO1 5DD
UNITED KINGDOM
Tel. +44 (0) 1904 432764
Fax +44 (0) 1904 432708
Email: tim.kelly@cs.york.ac.uk
Web site: http://www-users.cs.york.ac.uk/~tpk/