Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 65 (2015) 422 – 431

Procedia Computer Science

International Conference on Communication, Management and Information Technology (ICC 2015)

Decision Model for Software Architectural Tactics Selection based on Quality Attributes Requirements

Ahmed E. Sabry*

Department of Computer and IS, Sadat Academy, Cairo, Egypt

Abstract

Due to increasing industrial demands toward software systems with increasing complexity and challenging quality requireme software architecture and implementation mechanisms become an important activity. The decisions made during architecture design have significant implications on quality goals. As addressed, there is a lack of available standard models, architectures frameworks for enabling implementation of quality attributes specially for business intelligence environment and application order to rapidly and efficiently supports decision-making. In addition, a lack of researches related to Quality Attributes (QA) requirements, its implementation tactics, and interrelations or correlations between them. The increasing systems complexity mandates software architects to choose from a growing number of design options (decisions) when searching for an optimal architecture design in a specific domain with respect to a defined (set of) quality attributes and constraints. This results in a design space search that is over human capabilities and makes the architectural design task more complicated.

In this paper, researcher aimed to reveal most of quality attributes implementation tactics affecting applications architectures properties. Several quality attributes of software investigated using applied research methods with mixed quantitative (linear non-linear analysis techniques. It proposes an initiative for finding an easy and systematic way of addressing quality attribute requirements to a set of implementing architectural tactics.

Finally, the findings analyzed and visualized in a way that can support decision stakeholders in addition to a new concept of “safe-tactics” introduced as reduced (pruned) set of tactics that are claimed to be better used in general refactoring cases. In addition, a software tool is developed throughout this research effort as result of gained knowledge and addressing the resear findings.

Peer-review under responsibility of Universal Society for Applied Research

Keywords: Software Architecture; Architecture Styles and Tactics; Quality Attributes (QA); Mining Techniques; Business Intelligence (BI)

* Corresponding author. Tel.:+2-011-120-00747; fax. +2-011-120-00747.

E-mail address: aesmat@ieee.org; aesmat@sadatacademy.edu.eg

Peer-review under responsibility of Universal Society for Applied Research

doi:10.1016/j.procs.2015.09.111

1 Background

Architecture specifications and models are used to structure complex software systems and to provide a ten that is the foundation for other software engineering activities. The decisions made during the activity of archit design have significant implications on software quality goals[1]. In addition, the modern systems and bu intelligence applications implies fundamental knowledge redistribution and requires a careful rethinking c management of information resources and knowledge bases[2]. This mandates more rational decisions shou carefully taken and formal methods and tools should be introduced.

In addition, the growing use of cloud computing, software as a service, as well as open source, increase the from enterprises to implement new applications specifically business intelligence (BI). Related to this, a frame that can help in assessing the readiness of BI implementation for enterprises is increasingly required[3]. Bu intelligence level of readiness can be evaluated while it still considered weak but it can be identified. The c framework tries to further improve the success rate and reduce complexity of BI implementation. The key facto implementing BI architecture specially based on service-oriented architectures have not yet been systemat investigated[4]. Most of the prior studies focus on organizational and managerial perspectives over the tec factors. In this research, the technical factors and tactics that have most affect the implementation of BI architec

Lately, a gap noticed within professional communities and formal researches related to implementation tact software QA requirements. In addition, interrelations and correlations between them and its implication i and/or cost assessment are almost absent in architect decisions. It has been found scattered across many profes and research communities and system domains while similar approaches are proposed in multiple domains w being aware of each other[1].

Quality Attributes (QA) has significant influence on the architecture of enterprise systems. As Archite provides the foundation for achieving quality attributes and adhered to in the implementation. In the p researcher surveyed and defining the main quality attributes. In addition, the interaction and effect of each q attribute with implementation tactics.

1.1 System and Software Architecture Modeling

System Architecture can be defined as the set of principal design decisions taken for a system. Sy architecture is a means for describing the elements and interactions of a complete system including its hardwar software elements[5]. It is concerned with the elements of the system and their contribution toward the system's but not with their substructure.

Software architectures provide high-level abstractions in the form of coarse-grained processing, connecting data elements, their interfaces, and their configurations. This additional level of abstraction, while a comprehension and construction of software systems, and requires an additional effort for its use[1].

The software architecture of a program or computing system is the structure or structures of the system, v comprise the software elements, the externally visible properties of those elements, and the relationships a them (Bass, Clements, & Kazman, Software Architecture in Practice, 2003).

Functionality is the ability of a system to do the work it was intended to do. Functionality often has assoc quality attribute requirements (e.g., an f function is required to have a certain level of availability, reliability performance). Architect can achieve functional requirements and yet fail to meet their associated quality att requirements, as functionality can be achieved using many different architectures.

Enterprise architecture frameworks are aimed at the architecture of the whole organization (sometimes refer as the "application landscape"), rather than the systems within it. However, they share many of the concepts their systems architecture counterparts, and in particular, they all have at their core the notion of views[6].

1.2 Quality Attributes

One of the important concepts in software architecture specification is identifying required levels measurement of software quality attributes (QA) or system qualities such as performance, security, availal reusability and so on. Architects have to choose from a growing number of design options (decisions). Searc space become often beyond human (architect) capabilities for an optimal architecture design and implement tactics within a specific requirements-context or domain with respect to a defined (set of) software qualities constraints.

Quality attributes are properties of work-products or products by which stakeholders judge their qua Examples of quality attributes by which stakeholders may judge the quality of software systems m ay inc availability, usability, interoperability, configurability, performance, security, modifiability, reliability, portab etc. The degree to which a software system meets its quality attribute requirements depends on its architec Thus, architectural decisions are made to promote various quality attributes and a change in architecture to pro one quality attribute often affects other quality attributes. Achieving quality attribute requirements can only be through rationale choice of architectures.

Architecture provides the foundation for achieving quality attributes but is useless if not adhered to ir implementation. That is why the implementation of the architecture should be aligned, controlled and adapted implementation context. It is important to consider the most positive as well as the most negative influence imposing an architectural style[1]. Measurements of software quality attributes, is one of the important concep software architecture evaluation and variety of techniques are used for analyzing specific quality attributes system[1]. Promoting one quality attribute requirement usually has an adverse effect on some other quality attri requirement[1]. Architectural decisions will promote some quality attribute requirements while inhibiting others, resulting in quality-attribute tradeoff decisions. These tradeoffs are best dealt with in the earliest phases of sy development-during the design of the architecture[1].

In general, it is not possible to select an architecture style, which addresses all of quality attribute requiremen specific style is suitable for some special goals and not for all purposes, as provide not all quality attrib simultaneously could be achieved[1]. Therefore, the selection of architecture style must have good trade-off bet required quality attributes in system.

2 Quality Attributes and Tactics

Quality attributes are characteristics that the system has, as opposed to what the system does, such as usab maintainability, performance, and reliability[7]. Quality attributes are not simply met, but rather, satisfaction is alc scale that can be viewed in a scenario (Bass et al., 2003; Bachmann et al., 2005). Considering the fact that qu attributes tend to be system-wide characteristics, system-wide approaches are needed to achieve them. satisfaction should be reached on system architecture level, not on component level[8]. Systems have mul important quality-attributes and decisions made to satisfy a particular quality attribute will affect other qu attributes. For example, decisions to maximize performance will require cost of additional memory nee Therefore, architects must make tradeoff decisions to implement software that optimizes the best set of qu attributes combinations[8].

Because of the importance of quality attributes, it is critical that they be considered during early archite design. It has been addressed that architects commonly consider them simultaneously[8]. However, architects making architectural decisions concerning which tactics to implement and it could be difficult to imple correctly and control.

Tactics are measures taken to improve quality attributes. A tactic may be easily implemented using the structures (and compatible behavior) as a particular architecture pattern. Consequently, a tactic may re significant refactoring to structure and behavior of the pattern, or apply entire new structures or behavior. In case, implementation of the tactic will be more difficult and mandate extensive testing effort. Tactics ca classified as design time tactics related to overall approaches to design and implementation, such as "informa hiding" to improve modifiability, or may be runtime tactics, which are particular aspect of a quality attribute, as "users authentication" to improve security[8].

The implementation tactics should be selected based on quality attribute requirements. For example, security be improved by resisting attacks, detecting attacks, and recovering from attacks. These are categories of tactic security. Tactics for the “resisting attacks” design concern including: Authenticate users, Authorize Users, Mai Data Confidentiality, Maintain Integrity, Limit Exposure, and Limit Access[8].

In some cases, tactics are alternate ways of implementing a design concern. For example, a design concer availability (often called reliability) is “Fault Detection.” Two tactics for fault detection are “Ping/Echo’ “Heartbeat (dead man timer)”. We note that while the model applies to both design time and runtime tactic have focused primarily on runtime tactics for simplicity. The implementation of tactics improves the level quality attribute. However, tactics as addressed will also have side effects on other quality attributes[8]. These e are positive in some cases, and they are negative in many cases, as it will be more elaborated within the next se of the paper.

3 Implementation of Methods

As multi-disciplinary research methods used in this research applied research survey and case study ori methods with mixed quantitative (linear) and non-linear (data mining) analysis techniques.

The main survey has been conducted for the period of twelve months (one complete year) during the period December 2013 to November 2014. Questionnaire used as one of methods used in this study**.** As the resea desires to collect factual information on factors contributing to the area of subject, a “likert” type questionnair developed to collect data for the research questions stated.

The questionnaire was made up of 50 close-ended items for the stakeholders. Close-ended questions are qu compile and straight forward to code. It was distributed personally to the stakeholders of the selected gr companies, or enterprises on appointed and accepted dates. The researcher discussed the questions with mc them and later distributed them to respondents to answer. This was adopted by the researcher because it hel determining values as well as views, attitudes and experiences of the respondents. The purposive sampling us the selection of the respondents and utilized to answer the questionnaires.

In this regard, purposive sampling based on certain criteria laid down by research that the respondents v population have meaning for the data that will be gathered. The respondents of the study will be designer architects from academia and industry who are involved in the software development and engineering processes

As part of reference data analysed, the constructed dataset developed based on evaluating the ten non-funct requirements over fifty-two affecting architecture implementation tactics listed in Table 1, which resulted ir possible effects as search space size. While we have nine scale degrees for measuring effect of a specific tactic each non-function requirement, we will have three-dimensional space to model (quality attributes dimer implementation tactics dimension, and positive/negative effect dimension listed in Table 2). This constructs a table with 4,680 records used in decision tree induction.

Table 1. Surveyed Tactics

Group	Code	Name	Code	Name
Fault Detection	1	Ping/ Echo	2	Heartbeat
	3	Exception		
Recovery Preparation and Repair:	4	Voting	5	Active Redundancy
	6	Passive Redundancy	7	Spare
Recovery Reintroduction:	8	Shadow	9	State Resynchronization
	10	Rollback		
Prevention:	11	Removal from service	12	Transaction
	13	Process Monitor		
Resisting Attacks:	14	Authenticate Users	15	Authorize Users
	16	Maintain Data Confidentiality	17	Maintain Integrity
	18	Limit Exposure	19	Limit Access
Detecting Attack:	20	Intrusion Detection		
Recovering From Attack:	21	Restoration	22	Identification by audit trial
Failure Detection:	23	Timeout	24	Time Strap (stamp)
	25	Sanity Checking		
Failure Containment:	26	Redundancy	27	Replication
	28	Functional Redundancy		

	29	Analytical Redundancy		
Recovery Reintroduction:	30	Fix the Error	31	Rollback
	32	Degradation	33	Reconfiguration
Masking:	34	Voting		
Rank Manage Input/Output:	35	Record /Playback	36	Separate Interfaces from
Implementation	37	Specialized Access Routines/Interfaces		
Localize Changes:	38	Semantic Coherence	39	Anticipated Changes
	40	Generalize Module	41	Limit Possible Options
	42	Abstract Common Services		
Prevention of Ripple Effect:	43	Hide Information	44	Maintain Existing Information
	45	Restrict Communication Paths	46	Use an Intermediary
Internal Monitoring:	47	Built-in Monitors		
Defer Binding Time:	48	Run-time registration	49	Configuration Files
	50	Polymorphism	51	Component Replacement
	52	Adherence to Define Protocols		

Table 2. Tactics Effect Ranking Degrees

Direction	Degree				
Positive	Neutral (0)	Low (1)	Med (2)	High (3)	Very High (4)
Negative		Neg. Low (-1)	Neg. Med. (-2)	Neg. High (-3)	Neg. Very High (-4)

The survey questionnaire was used as one of the supplementary data-gathering methods for this study. questionnaire was divided into four main sections in the Table 3.

Table 3: Survey sections summary.

Section	Description
First Section	about current organization, role, location, and contact information
Second Section	about Experience and main technologies
Third Section	about architectural views used and communicated
Fourth section	about systems and business domains
Fifth Section	about quality attributes main concern
Sixth Section	about systems specific quality attributes, architectural patterns, and size related metrics
Seventh Section	as last and important section about tactics used in software architecture implementation

The questions were structured to provide five choices for every question or statement. The choices represen degree of agreement, usage, knowledge, or utilization each respondent has for the given question. It enable respondent to answer the survey questions easily. In addition, it allows the researcher to carry out quantit approach effectively or statistically analyze data and data interpretation. A sample size of forty-two respon have been gathered and summarized. Table 4 summarizes responses over each surveyed QA as general values and visualized responses results in Figure 1 (a) as general values and in Figure 1 (b) as ranked quality attri usage levels and more details listed in Appendix B.

Table 4: QA's general values within each level.

Response	1.Functionality	2.Portability	3.Maintainability	4.Efficiency	5.Integrity	6.Reliability	7.Usability	8.Confidentiality	9.Availability	10.Safety
High	29	11	20	15	17	24	17	16	25	14
Medium	11	16	17	24	15	13	21	15	14	18
Low	2	12	5	2	8	5	4	8	3	7
Never Used		2						2		2
Don't Know		1		1	2			1		1
	42	**42**	**42**	**42**	**42**	**42**	**42**	**42**	**42**	**42**

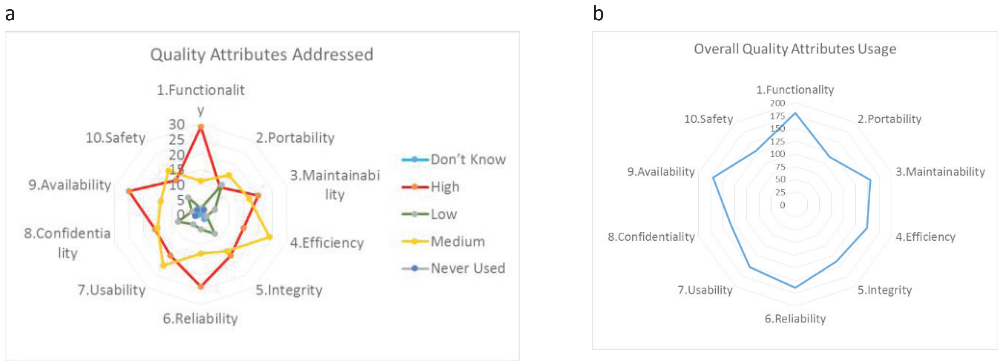

Figure 1: (a) Addressed QA usage levels frequencies; (b) Addressed overall ranked QA usage frequencies.

The responses to questions in the given variables were scaled using the five-pint-scale or Likert scale syster given weight (rated). The normalized rated QA values listed in Table 5 and visualized in Figure 3.

Table 5: Resulting normalized QA values (weighted QA's importance).

QA Normalized Values	0.12	0.08	0.10	0.10	0.09	0.11	0.10	0.09	0.11
BI Normalized Values	0.13	0.07	0.10	0.11	0.11	0.14	0.07	0.07	0.14

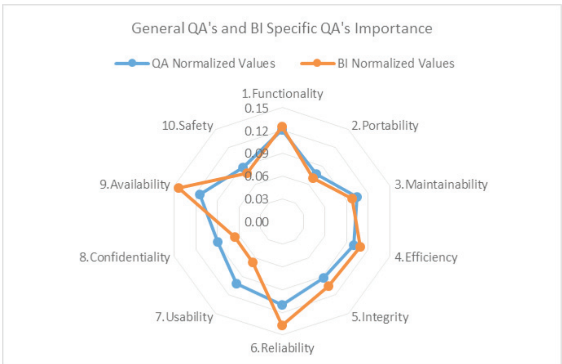

Figure 2: General QA's and BI Specific QA's Importance

Referring to Figure 2 visualized results, it can be concluded visibly that the interest goes more important v BI environments for the following QA's in order:

- Availability
- Reliability
- Integrity

Figure 4 visualizes the pruned decision tree reduced (common) twenty tactics considered as most-affecting t: for trade-offs for implementing the non-functional requirements as deduced from resulting decision tree.

4 Implementation

The implementation effort aimed to develop an easy-to-use and to understand decision model and tool to this process easier, rationale and systematic. Thus supposed to maximize requirements satisfaction and hav optimal set of tactics to implement. In addition, a well prepared set (from the research results) of reduced (comr twenty tactics considered as most-positive-affecting tactics for trade-offs in which we name it "safe-tactic: implement for moving most of the addressed system qualities to better levels without having significant neg: impact on other qualities.

In addition, provide roadmap and guidelines for software architects they can use to rationalize architecturally significant decisions when considering different domains and/or changing quality attrit requirements.

The implemented tool addresses the core part of the proposed decision framework, which is to map, recomr and measure the best implementation design decisions tactics that satisfies specified levels of quality attrit requirements and measure its interrelated effects. Appendix A shows the developed database design corresponding scheme description for the developed tool.

The implemented results including introduced "safe-tactics" are listed in Table 6.

Table 6. The decision tree survived tactics after pruning: named "safe-tactics" list

ID	Tactic Name	Rank	ID	Tactic Name	Rank
51	Component Replacement	1	39	Anticipated Changes	2
40	Generalize Module	3	33	Reconfiguration	4
24	Time Strap (stamp)	5	35	Record /Playback	6
21	Restoration	7	30	Fix the Error	8
23	Timeout	9	18	Limit Exposure	10
11	Removal from service	11	15	Authorize Users	12
6	Passive Redundancy	13	2	Heartbeat	14
14	Authenticate Users	15	26	Redundancy	16
8	Shadow	17	4	Voting	18
37	Specialized Access Routines/Interfaces	19	1	Ping/ Echo	20

5 Conclusions

General framework for implementing architecture with a specific QA's requirements were proposed developed for supporting relevant stakeholders design decisions. It has been observed as validated in the dec tree pruning, that the implementation of many tactics is almost useless and in many cases harmful. This propose reduces (pruned) set of tactics that are better used in general refactoring cases as shown in the pruned decision named "safe- tactics". It has been selected as the largest positive trade-off effects for the evaluated ten qu attributes, while having the least negative effect on the same set of quality attributes.

The stakeholders have been derived through surveys with participants in the field of architec implementation, and service providers. The participants observed that it clearly shows which components mu: refactored or modified. The participants observed that it is useful to use and know how tactics interact with other and their effect on quality attributes required levels.

The case study showed several benefits of implementing suitable tactics in the distributed architecture d specially moving towards more BI environment readiness. In addition, it shows the addition of tactics for performance and security; where an additional authorization component was required affect the compone performance. In addition, it has been observed that the amount of documentation needed is becoming smaller the modelled rules as it clearly addresses the QA-tactics rationale. As documentation of interaction betwee quality-attributes and tactics exists, it can be leveraged to minimize the amount of writing needed.

The interaction of multiple tactics as well as tactics on multiple patterns should have additional study. W considering how the impact of tactics changes when considered in the context of multiple quality attribute l required. It has long been known that architecture patterns and quality attributes are dependent, but have signif interaction with each other. We have found that this relationship is very rich, and involves the implementatic quality attributes through tactics.

Finally, the interaction among quality attributes and tactics falls into several general categories, based on the of changes needed in order to implement the tactic. The amount of work required to implement a tactic i architecture that uses a pattern depends on the type and volume of change needed.

As a limitation, it need more research to know how to generalize proposed and implemented framewor general model for measuring enterprise BI readiness from solutions architecture prospective.

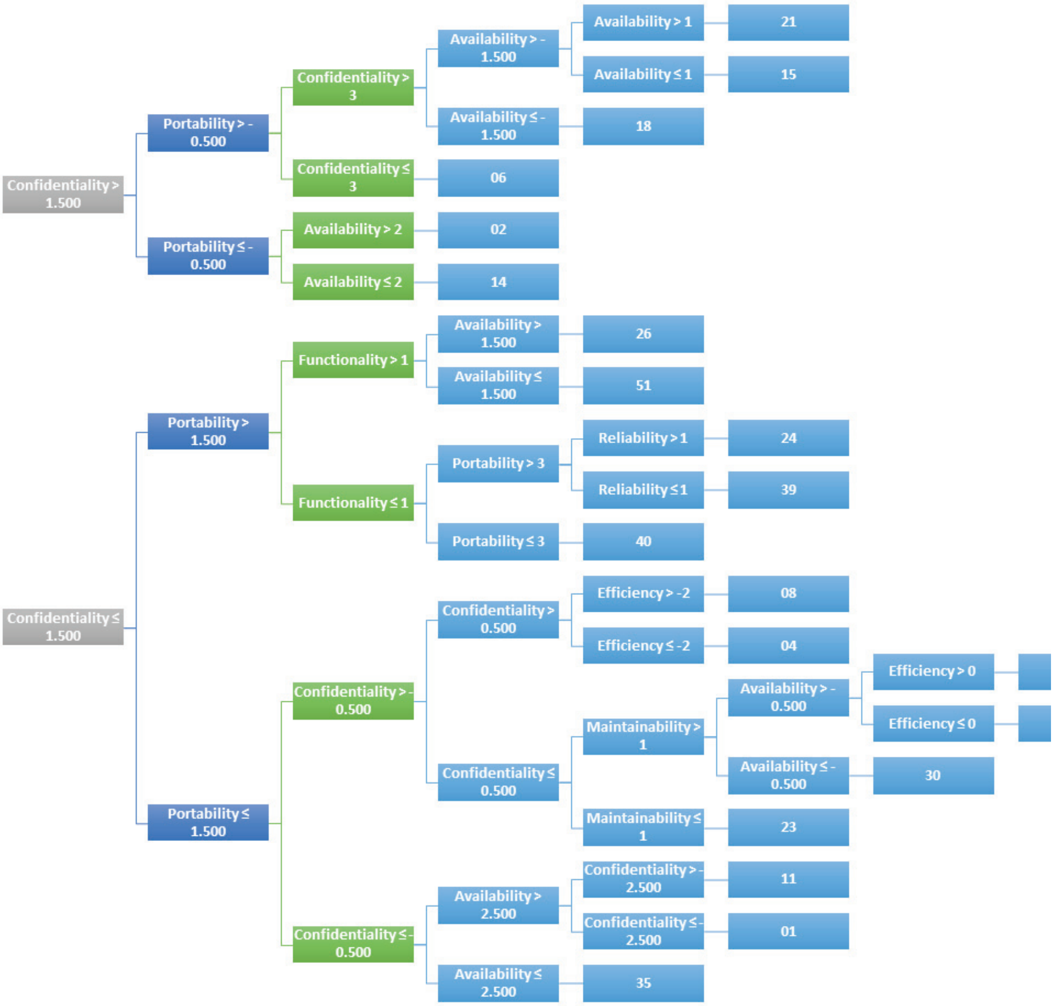

Figure 3: Visualized implemented rules deduced from the Decision Tree for Tactics Selection

Appendix A. Implemented Database Design

The database design with main entities visualized in Figure 4.

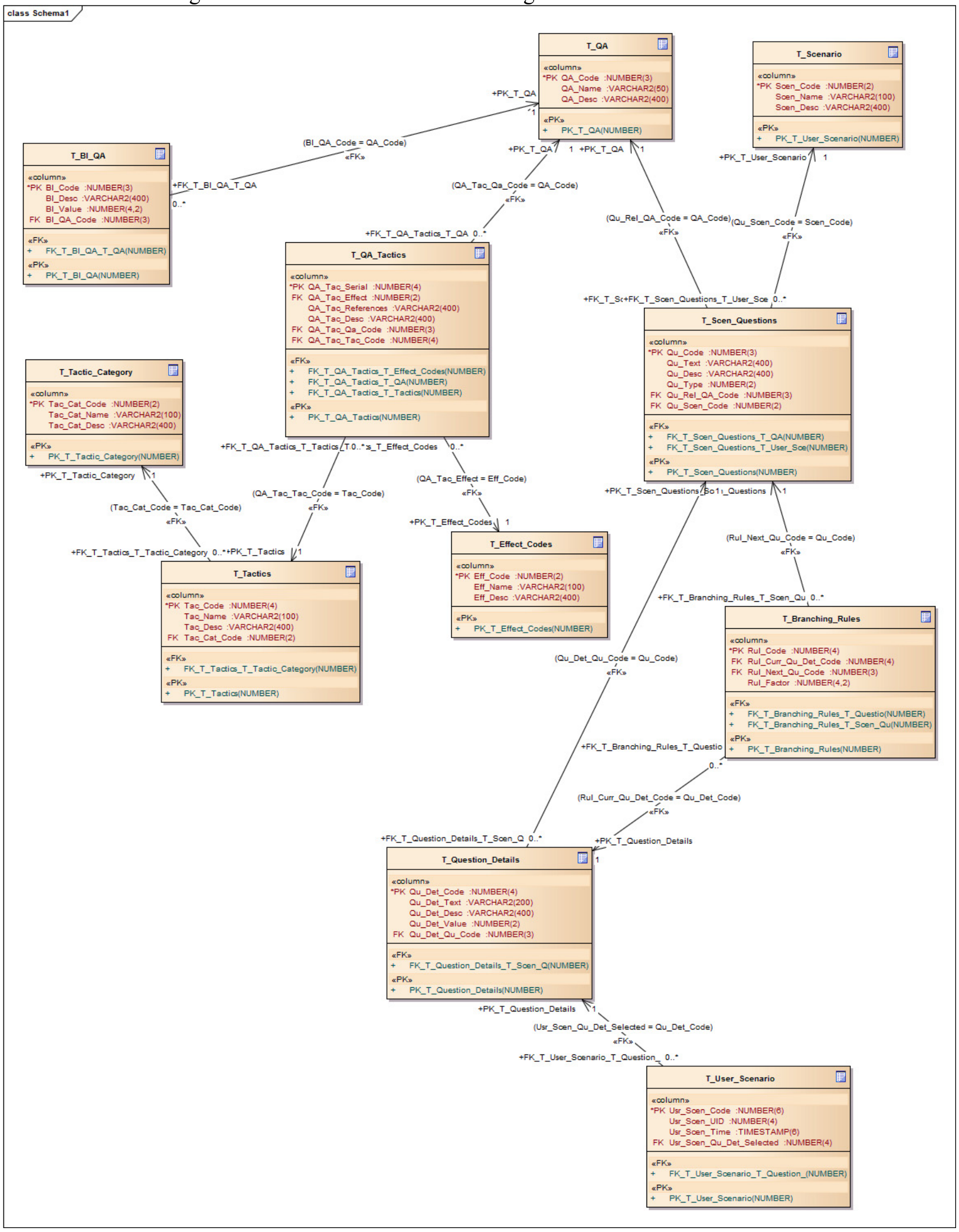

Figure 4: Developed Data Model for Proposed Tool

Appendix B. Survey Data Summary

The following table summarizes selected responses data from conducted survey related to applications tecl domain.

Table 7: Number of Systems per Respondents Country

Country	Data-flow and Production	IS and Enterprise Systems	Web-Based Systems	Scientific Applications	Cloud Computing Applications	Mobile Applications	DSS and BI Apps	Total
Australia		2	5	0	2	4		**13**
Belize	34	32	30	32	28	39	86	**281**
Canada	2	0	1	0	0	2	0	**5**
Colombia	3	5	1	0	2	1	1	**13**
Egypt	111	21	15	0	5	5	5	**162**
France	1	1	1	1	1	1	1	**7**
Germany	1	0	2	0	0	0	0	**3**
India	29	12	33	0	5	4	4	**87**
Japan	5	4	5	4	1	4	4	**27**
Kuwait	0	1	0	0	0	0	0	**1**
Macedonia	3	8	8	0	2	0	3	**24**
Pakistan	2	2	1					**5**
Romania	2	1	1	0	1	0	1	**6**
South Africa	2	0	10	0	2	5	0	**19**
Spain	3	3	6	0	1	1	2	**16**
Sweden	0	2	1	0	1	0	0	**4**
United Arab Emirates	6	6	7	1	3	4	3	**30**
United Kingdom	5	3	27	2	3	1	0	**41**
Uruguay	3	0	4		4		1	**12**
Unknown	23	6	5	0	1	0	3	**38**
Total	**235**	**109**	**163**	**40**	**62**	**71**	**114**	**794**

References

1. N. Eftekhari, M. Poyan Rad and A. Hamid, "Evaluation and Classifying Software Architecture Styles Due to Quality Attributes," Aus Journal of Basic and Applied Sciences, vol. 5, no. 11, pp. 1251-1256, 2011.
2 S. Egaravanda, L. Nugroho, T. Bharata and A. Munawar, "TRANSFORMING G2C/C2G RELATIONS THROUGH VIR COLLABORATION," in ICTS, Bali, 2013.
3. A. Hidayanto, R. Kristianto and M. Shihab, "Business Intelligence Implementation Readiness: A Framework Development," Intern Research Symposium in Service Management (IRSSM), 2012.
4. L.-K. Chan, W. Yeoh, W.-O. Choo and P.-Y. Lau, "Technical Factors for Implementing SOA-Based Business Intelligence Archite Communications of the IBIMA, vol. 2012, p. 10, 2012.
5. L. Bass, P. Clements and R. Kazman, Software Architecture in Practice, Addison Wesley, 2003.
6. N. ROZANSKI and E. WOODS, "APPENDIX," in SOFTWARE SYSTEMS ARCHITECTURE, Addison-Wesley, pp. 627-629.
7. I. Sommerville, Software Engineering, 9th ed., Addison-Wesley, 2010.
8. N. B. Harrison and P. Avgeriou, "How do architecture patterns and tactics interact? A model and annotation," Journal of Syster Software, vol. 83, p. 1735–1758, 2010.