# Table of Contents for CIO Summary in Part 1

**TABLE OF CONTENTS FOR CIO SUMMARY IN PART 1** **1**
**PART 1. ARCHITECTURE 101 – 'THE LANGUAGE WE SPEAK'** **2**
<<< … >>> 2
CIO SUMMARY 2
**<<< … >>>** **5**

# Part 1. Architecture 101 – 'The Language We Speak'

<<< … >>>

## CIO Summary

Enterprise IT Architecture or Software Architecture (preferred term in this book, for the title at least) is an evolving and ill-defined field, with high level of hype, buzzword noise, and grey areas.
At the same time, modern Enterprise becomes more and more reliant on the Information Technology in achieving the competitive edge, and in staying in the high technology-powered business in the first place.

As with programming and data modelling before that, IT industry makes a forceful effort of applying the rigour to IT Architectures.
We are aiming to define Enterprise IT Architecture in the commonly agreed terms. We strive to define what makes the Enterprise Architecture good, how to build a good Enterprise IT Architecture, and who are the good candidates for the task.

IT Architectures cut across many technologies, business processes or infotainment activities, and involve or affect many different stakeholders. Despite high level of technical and technological content in what comprises the IT Architect's expertise, building Enterprise IT Architecture is a predominantly social activity conducted by people, in collaboration with people, and for people.
After all said and done, and, figuratively speaking, after accountants counted all beans, the perceived success of IT Architecture directly depends on how well all stakeholders aligned behind the idea and the operating model.
There cannot be a 'perfect' technical solution for evolving and diverse business if idea was not 'sold' to all Enterprise stakeholders. Needless to say, IT Architecture proposal will not even take off the ground if Architects failed to subscribe to the idea stakeholders who pay the money and give go-ahead to the proposal.

Growing importance of IT Architectures for all concerned necessitates common understanding of the software concepts in the context of the Enterprise, and common vocabulary facilitating the collaboration of all Enterprise stakeholders in constructing and running efficient, fit-for-purpose IT Architectures.

If we follow the top-down, or 'outside in', view on the Enterprise IT Architecture, core business of the Enterprise provides a background and motivation for IT Architecture.
Business goals and the operating environment of the Enterprise translate into requirements, constraints and criteria for building the IT Architecture. Business context may be as infinitely complex as nature itself, but we consciously abstract from the overwhelming details that deemed to be of insignificant importance to the immediate Enterprise business at hand. We may miss some important details, but we just have to limit the complexity and cardinality of the elements and relationships in the architecture due to our limitations in technology, and our limited ability to comprehend and manage the processes in the real world.
We push those limits of technology and our understanding further and further with constant technological advances, but they (limits) still exist and need to be factored into our IT Architecture designs.

In Chapter 1, we start with philosophical foundations of IT Architecture and its place in the overall scheme of things. We look at IT Architecture as manageable and sufficient model of the real world. From this prospective we explain notions of *abstraction*, *decomposition*, *refinement*, *layering*, *separation of concerns*, *views* – our techniques for dealing with complexity.

Our ability to capture true business requirements and business goals, and, equally as important, to reach the common understanding of and agreement on business requirements with all Enterprise stakeholders – is a defining moment in building the successful Enterprise IT Architecture.
Two extreme examples can illustrate this statement: computationally efficient and tricky program may inspire the awe in your peers-programmers but fails to impress your manager who still did not get the business outcome he or she requested; accounting application may perform its task by keeping the business running without much fuss but scoffed at by proficient programmers who may have to deal with the ugly code 'under the hood' of the application.
We do not advocate here the ugliness in the programming and micro-design, but the efficiency under the hood in the IT Architecture should not compromise overarching business requirements of the Enterprise.

In order to follow the thread of discussion on IT Architecture, and to make a meaningful contribution into such a discussion, the participant must understand some *Key Concepts* that are fundamental in the enterprise computing, regardless of the type of enterprise business or dominant in your enterprise methodologies and technologies.
In Chapter 2, we arbitrarily identified these key concepts that pervade throughout your experiences in building IT Architectures, and prime you for better understanding of particular methodologies, technologies and products and their positioning on the bigger picture. We wholeheartedly subscribe to the opinion that understanding of some fundamental principles and concepts is of primary importance, as opposed to the cluttered knowledge of specific facts about particular technologies and products.

Standards enshrine the common understanding and acceptance of key concepts and technologies. We outline the importance and general procedure of the standardisation process in relation to the Internet standards.

Arguably, 'key concepts' of the enterprise computing could include many complete university courses on System and Software Engineering. We had to draw the line somewhere.
On one hand, there is not much point in repeating all textbook material here. On the other hand, experienced professionals who were in the industry for a long time may have missed some fundamental learning material that recent university graduates take for granted.

We explain the pervasive notion of *transaction*.
Business processes consist of series of *business transactions*. Business transaction translates in the Enterprise IT Architecture into one or many transactions in the Transaction Processing Monitor, Application Server or Database Management System.
Usually, concept of *transaction* is easily understood. That is, the 'sunny day' scenario for the transaction is easily understood. Things get complicated very quickly when something may and will go wrong, for variety of reasons, in different places in the IT infrastructure and at the different times. Hardware may fail, software or human error may disrupt the normal execution, and the external forces or disasters may interrupt the normal transactional flow at any moment.
If we did not build the resilience to transactional failures into our IT Architecture upfront, 'rainy day' scenarios will eventuate sooner or later, and likely with loss or corruption of valuable data.

To preserve integrity of data and processes under any conditions of the executing transaction, IT Architecture must reliably enforce ACID properties of transactions.
Note that complex transactions must possess ACID properties as a whole. I.e. successful *commit* or *rollback* of the part of transaction does not ensure integrity of the overall business transaction if that complex transaction failed to reach the completion or to recover to the previous consistent state.

Failure of standalone transactions may pale into insignificance comparing to prolonged failure of large parts or the whole of the IT Architecture.
Enterprise IT Architecture must devise strategy and possess capacity for the *Business Continuity* in case of disasters or man-made catastrophic events. Show must do on.

Persisting data on the long-term storage devices may be considered a part of the transaction processing in general. However, *database transactions* and Database Management are of particular importance in the IT Architecture, and warrant special attention.
Database Management Systems provide repository for the business data and enforce ACID properties for database transactions.

All products and applications, like any living organism, progress through the stages of the Application Life Cycle from its birth (inception) to the death (decommissioning).
In general, Software Design and Development, as well as Project Management methodologies, aim to streamline and manage the evolution of application through these stages. Methodologies help manage the construction of IT Architecture by balancing 3Rs – Requirements, Resources, Risks.
We did not discuss methodologies in detail in this part – only identified them as key concepts.

Sheer complexity and diversity of inter-related building blocks implies that integration efforts will be a prominent part in building the Enterprise IT Architectures. Hence, Enterprise Application Integration and, more specifically, *Integration Interfaces* are the key concepts in understanding re-use and component-based architectures.
In essence, *Integration Interfaces* in the Enterprise IT Architecture is the concept of the *Interface* in the Object-Oriented Design, graduated into the Enterprise. Often, *Integration Interface* goes by the name *Public Interface* or *Contract*, and implemented as an *Application Programmer Interface* (API).

Integration Interfaces help to pull apart the otherwise monolithic blob of the Enterprise Architecture into manageable, replaceable components and layers. Conversely, Integration Interfaces is a technique for putting together disjointed pieces of the architecture, or disjointed architectures.
Architecture design is the iterative and recursive process of Separation Of Concerns, de-coupling and de-composition of layers and components, layering and partitioning (or segmentation) of pieces in the architecture, refinement of components and interfaces.

Enterprise IT Architecture represents valuable mission-critical part of the modern Enterprise. Business requires easy mass public access by great multitude of customers, partners and personnel to the precious computational and information assets of the Enterprise. At the same time, security and privacy imperatives demand rigorous security checks and procedures, both for managing the access by various legitimate customers, and for preventing or minimising the damage from malicious perpetrators. This mismatch between service availability and required protection sets the scene for the Security and Access Control in the Enterprise.
We explain the concepts of Security and Access Control, Authentication and Authorisation, Security Standards and overall Security Framework, Single Sign-On (SSO) and Federated Identity.

Miscellaneous key concepts include *Client* and *Server*, *Distributed* and *Remote*, *Real-Time* and *Batch*, *Online* and *Off-line*, *Synchronous* and *Asynchronous*.

Concept of *Session* is very important in understanding the state management in distributed transactions over the Internet.

Construction of the new Enterprise solution, or assessment of existing solutions or components for fitness to your Enterprise, requires quantifiable and differentiating *Quality Measures* of Enterprise IT Architectures.
Not surprisingly, discussion of *Quality Measures* in Chapter 3 may look like continuation of discussion of the *Key Concepts* in the Enterprise Architecture.

We discuss most common important Quality Measures that will help you to differentiate and to analyse the suitability of candidate technologies or products. Your evaluation spreadsheet will likely contain such Quality Measures as *Business Continuity*, *Time To Market*, *Total Cost of Ownership* (TCO), *Scalability* and *Availability* etc.
Due to the specific context of your application and your Enterprise, we expect that you revise and expand this list of Quality Measures to suit your task better, and review their priorities or weights.
Either way, do not forget that even with quantifiable and weighted *Quality Measures*, benefits of Enterprise solution may be complex and to the great extent *qualitative*. You may wish to review the final score with some grain of salt, by going through the supplementary results of your discovery and analysis with the whole assessment team (not for the purpose of manipulating the scores and weights to suit the preferred outcome, but to recap and review your justification and completeness of the discovery process).

We devote the separate Chapter 4 to the discussion of various *Application Domains* and types of Enterprise applications where we build the Enterprise IT Architectures. List is by no means exhaustive. You may observe that applications of IT Architectures are vastly different, but concepts and challenges are very similar.

Having introduced *Key Concepts* of the Enterprise IT Architecture or Software Architecture, we can approach the task of defining the *Software Architecture* in Chapter 5.
There is no shortage of definitions for the Software Architecture. They differ significantly in wording and in the use of basic concepts that the definition is constructed from.
We provide our own version for the Software Architecture definition, alongside with expansive collection of 'classical' definitions from multiple sources for your reference. Following intended practical emphasis of this book, we are not engaging in the terminological arguments here, but trying to make sure that you are well armed with supporting information for your own professional debates.

Book provides overview of major Architecture Frameworks and Reference Models, like RM-ODP, Catalysis, TOGAF, Zachman, to name a few.

Next we look for answers to the question: "Who is the Software Architect?" We progress a little bit further than the obvious answer: " Software Architect is the person who does Software Architecture" – technically correct but not very helpful, even after we finally defined what the Software Architecture is.
We find that Software Architect is a highly professional all-rounder, with some metrics that help measuring the competency of the Software Architect. Accordingly, we are able to determine what makes a Software Architect good. In addition to the broad expertise in technology, Software Architect must possess qualities of strategic thinker, leader, consultant and mentor, proficient organisational politician.

Some Software Architects are stronger in some areas, other – in another areas. Software Architect is a very complex role, and a unique persona. There are no two Software Architects with equal combination of skillsets and abilities – some Software Architects are 'more equal' then other, so to speak.
Having accepted the likelihood of different types of Software Architects, we introduce the Taxonomy of Architects. Although based on realistic project situations, we assume that this taxonomy will be viewed in a manner of well-meaning humour, not during the business hours and outside the project documentation.

In Chapter 6, we follow the Software Architect's train of thought in iterative process of architecture design for the non-trivial, but typical Internet-based Enterprise application.
We walk the reader through the iterations of architecture design that progresses towards the better fulfilment of captured business requirements. We make a checkpoint and discuss the challenges and remaining deficiencies on every iteration of the design.
In a sense, we are ahead of ourselves in this chapter, because we did not discuss specific methodologies and technologies as yet. This chapter helps us demonstrate all major architecture concepts and quality measures in action on the real example, and gives us a foretaste of challenges and required technical knowledge in building efficient, fit-for-purpose Software Architectures.

<<< … >>>