

# CURA: Complex-system Unified Reference Architecture

## *Position Paper: A Practitioner View*

Ethan Hadar<sup>1</sup> and Irit Hadar<sup>2</sup>

<sup>1</sup>*Communities Informatics, Zefat Academic College, Zefat, Israel*

<sup>2</sup>*Department of Information Systems, University of Haifa, Haifa, Israel*

**Keywords:** Solution Architecture, Reference Architecture, TOGAF, Enterprise IT.

**Abstract:** Constructing enterprise-level solution requires integration of existing, modified, and new modular technologies. A customer specific solution is instantiated from a reference implementation owned by the services organization, as a result of multiple products and their reference design created by the R&D organization. Yet, the disciplines of R&D and enterprise architecture differ in their analysis and design processes, artifacts, and semantics, leading to a mismatch in product design, knowledge and requirements interpretation. The Complex-systems Unified Reference Architecture (CURA) was developed as a common platform for both field and R&D practices. This methodology binds a 4-layered structure and a 4-phased architecture process, controlling the solution architecture lifecycle from reference design to reference implementation and solution instantiation, and fits both agile and DevOps methodologies. The presented version of CURA was tested and implemented with several customers as a lean and minimal blueprinting approach, serving as part of the architectural deliverables. CURA can be adjusted to other visual binding notations such as UML and TOGAF modeling languages, and can scale up to system-of-systems design.

## 1 INTRODUCTION

Architecture-centric evolution (ACE) (Paris et al., 2006) is aimed at supporting requirements that bind product design constructed by R&D architects with solution architecture constructed by field architects, according to customers needs.

R&D architects are responsible for evolving product architecture, and delivering reference implementation to field architects. Field architects combine several reference implementation architectures into a reusable and replicable solution of reference implementation, to be later configured for a specific customer environment. Yet, miscommunicated intention between R&D and field architects can hamper project efficiency or generate customer dissatisfaction. Moreover, such miscommunications can lead to missed or overlooked technical evolution opportunity.

In this position paper, we present CURA, Complex-systems Unified Reference Architecture methodology, as a common methodology that unifies the discipline of architecture for both field and R&D practices.

The methodology depicted in Figure 1 binds a 4-layered architecture structure and a 4-phased architecture process, controlling the solution architecture lifecycle from reference design to reference implementation and solution instantiation. The CURA approach can be employed with any architecture discipline, such as agile and DevOps.

CURA was implemented in several customers' solutions, offering R&D and field architects:

- **Unified Communication** by means of common views across architecture departments.
- **A Single Architecture Discipline** that provides structured analysis of solutions from inception to instantiation.
- **Continuous Deliverables** creation is accelerated and allows for reusable and incremental analysis and design.

This position paper provides motivation for a unified architecture methodology presented in Section 2, and details the methodology' phases and architectural layers and views in Section 3. Finally it highlights the value proposition and differentiating approach in Section 4.

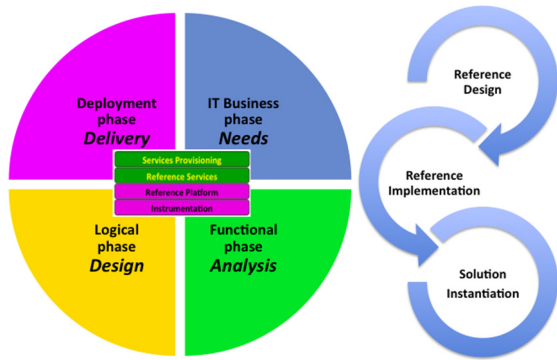


Figure 1: CURA 4x4: a methodology for Complex systems Unified Reference Architecture with 4 architecture layers and 4-phased architecture process.

## 2 MOTIVATION

An IT solution involves combining several products produced by teams with different development styles and software engineering practices (Passos et al., 2010). The products’ architecture varies in design principles and utilizes a myriad of environments and platforms. A field architect is required to add business value as well as improve the quality of the compound non-functional attributes. Architecture quality improvements can include a reduction of the solution’s footprint, removal of duplicated or confusing capabilities, and maximization of cloud services usage. Architecture requirements are created by the development organization, and later on by field architects. As such, a unified semantic language used from architecture inception to solution instantiation is required. The well-known modeling approach of Kruchten’s 4+1 views (Kruchten, 1995), addresses this need by advocating the Use Case view as a binding one, catering for business and product management. However, the approach leads to architecture instructions that are scattered across unrelated visual contexts. To illustrate, consider how DevOps test automation and deployment instructions over a cloud platform using Kruchten 4+1 can be captured with use cases. Specifically, on which blueprint and at what development phase should an architect emphasize the usage of integration and dependency on third party cloud PaaS vendors?

CURA addresses the above problems and offers a unified approach for creating visually connected views alongside a well-defined architecture process,

adjusted to multi-products integration and solution construction.

## 3 CURA BUILDING BLOCKS

This section details the CURA methodology artifacts, binding a 4-layered structure and a 4-phased architecture process, from reference design to reference implementation and solution instantiation.

### 3.1 CURA Architecture Layers

CURA architecture is structured as a 4-Layered High Level template (4LHL). Within the domain of enterprise IT, the four layers are *services provisioning*, *reference services*, *reference platform*, and *instrumentation* (depicted in Figure 2). Each subsequent view uses the same 4LHL layout with different type of CURA entities according to TOGAF (The Open Group Architecture Forum) semantics to bind the architecture perspectives (TOGAF, 2015). CURA accounts for modifications to the 4LHL elements (Amber, 2015), specifically by adjusting the layers and inner sub-layers for a specific solution domain.

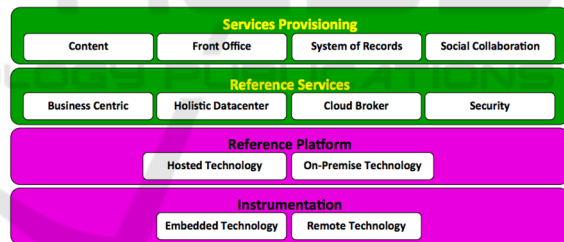


Figure 2: CURA 4-Layered High Level (4LHL) template.

#### 3.1.1 The Services Provisioning Layer

The *Services Provisioning* layer contains the following components:

- **Content** – provides access to content that is created by users of the solution within the production environment, such as readymade processes or configuration scripts; best practices documents; templates; policies; and complex events processing rules.
- **Front Office** – encapsulates presentation elements with adjustable graphical interfaces according to consumer’s devices. Examples are

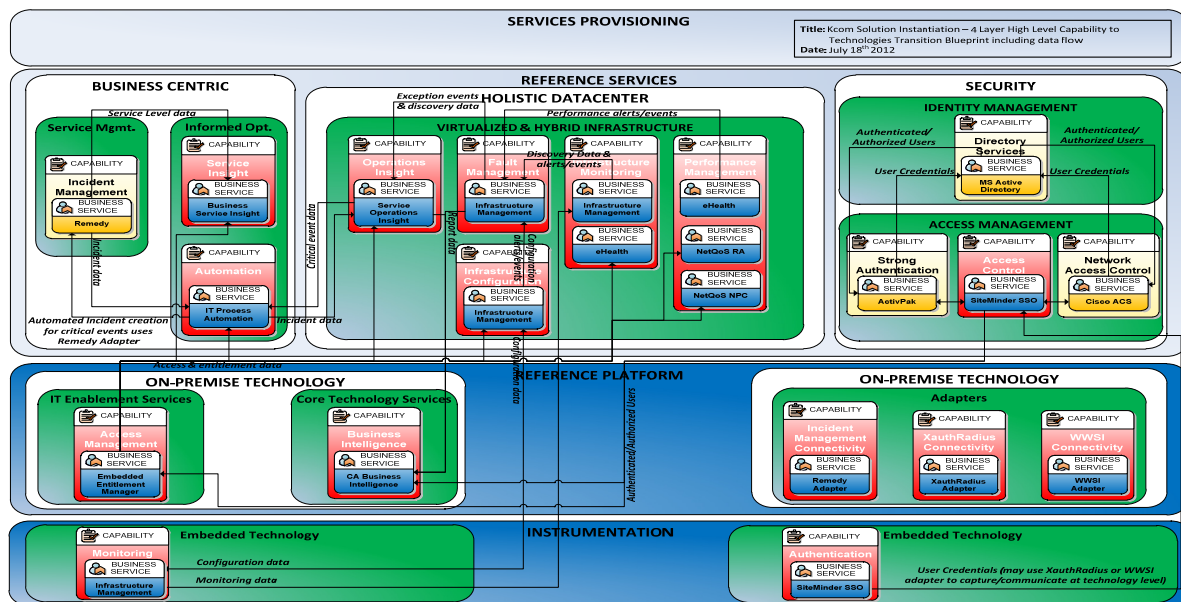


Figure 3: Example: A customer’s solution instantiation 4LHL capabilities to technologies blueprint (Created by Jason Davis for an enterprise level telecommunication company).

- code scripts for responsive web design (RWD); deployable rich client executable; packaged WAR code, HTML5 and CSS; etc.
- **System of Records** – exposes published application programming interfaces (API). Examples are the Web Services Description Language (WSDL) or the Representational State Transfer (REST) API, expressed in the solution’s Domain Specific Language (DSL) semantics.
- **Social Collaboration** – securely provides integration with third party social collaboration tools, adhering to compliance and regulations, such as blocking obscenity text.

### 3.1.2 The Reference Services Layer

The *Reference Services* layer contains the classical business logic and data tiers of a compound solution. Yet common components and remote platform services are excluded and captured within the reference platform and instrumentation layers. In this paper we demonstrate the usage of CURA in the domain of IT management. Consequently, architects would be able to adjust these core components according to their specific modeled domain.

In the IT management domain, the core components are:

- **Business Centric Module** – contains ITIL’s *Project, Portfolio & Financial Management* for controlling and governing contractual service level agreements, alongside financial aspects.

- **Holistic Datacenter** – focuses on IT system, performance and operational management, such as private cloud management and IT system health.
- **Cloud Broker** – focuses on interacting with remote cloud services. The goal of the gateway broker is to encapsulate the aggregation, intermediation, and arbitrage of external cloud services.
- **Heterogeneous Security** – focuses on authentication, authorization, and single-sign-on of an enterprise IT solution, including information fidelity concerns.

### 3.1.3 The Reference Platform Layer

The *Reference Platform* layer accesses IT platform services or readymade IT components. The sub-layer categories are:

- **Hosted Component** – contains commoditized IT services, such as those offered by IBM’s Bluemix PaaS, Softlayer or Amazon. These services can be mobile push notification, NoSQL DB, or even cognitive computing as offered by IBM WATSON. IT readymade packages hosted on a Managed Service Provider (MSP), can be readymade bundles such as the combined Node.js container and CloudAnt database web starter, or Python Flask starter kit.
- **On-Premises Component** – contains shared

components installed at the customer datacenter. It also contains managed connectivity to legacy products within the customer Enterprise Architecture (EA).

### 3.1.4 The Instrumentation and IoT Layer

The *Instrumentation and IoT* layer contains interaction protocols with managed devices. Such devices can be a server, network card, router, switch, or an appliance such as a smart TV, media device, or mobile device. Managed technologies can be agents, probes, sensors (monitors), or actuators (modifiers and controllers). Regardless of the managed target, the embedded and remote technologies are separated by ownership and location of the managed technology.

The **Embedded Technology** provides access to devices within the firewall (physical location) and private virtual networks.

The **Remote Technology** accesses devices via third party and public networks, with or without encryption.

## 3.2 CURA 4 Process Phases

CURA 4-phases process separates mandatory and optional views, structured on top of the 4LHL template as described in Table 1. The blueprints images were blurred to maintain confidentiality. The mandatory blueprints are capability-to-architecture roadmap; capability-to-technology as depicted in Figure 3; super-position logical 4x6 as depicted in Figure 4 (Hadar et al., 2012b); and deployment. The optional views are capability, security, and capability-to-business.

- **The Business Phase** focuses on understanding customer needs, and defines the required high-level capabilities in conjunction with the existing and needed technology. Aimed at detecting dependency on previously delivered capabilities as well as new evolving architecture requirements, the result is a merged capabilities and architecture value-roadmap.
- **The Functional Phase** focuses on systematic decomposition of capabilities into supporting processes and functionalities, by detecting explicit connectivity between technical components or implicit dependencies of business processes. Once the implementation technologies for the *Business-Service-Innovations* steps are detected (Hadar et al., 2012a), the technologies are mapped onto the 4LHL *Capabilities-to-Technologies* blueprint, exemplified in Figure 3.

- **The Logical Phase** binds 4x6 logical bricks (Hadar et al., 2012b) of existing or planned components of each participating product (see Figure 4). The 4x6 blueprint depicts both the underlying products and the solution glue-code. This phase includes eliciting the evolution steps required from both the products and solution teams, as well as the customer's enterprise architecture (EA) configuration needs. This list of evolution steps is merged with the combined capability and architecture roadmap view. The blueprint is based on aggregating a separate 4x6 logical product views, is a 6-tier architecture containing a modified 3-tier architecture pattern extended with three additional integration tiers for the EA.
- **The Deployment Phase** aims at producing instructions that can be automated by DevOps scripts for staging and producing the IT solution. The 4LHL *Deployment* Blueprint depicts the installation dependency of packaged application components on the required IT infrastructure.

## 4 IMPLEMENTATION AND CONCLUSION


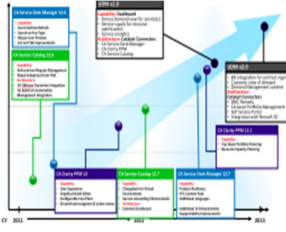

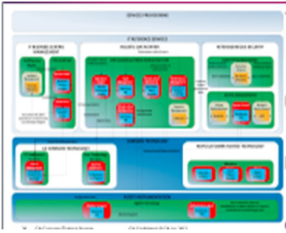
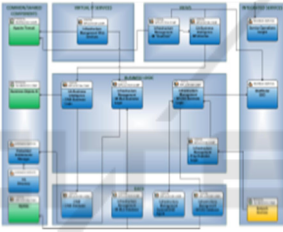

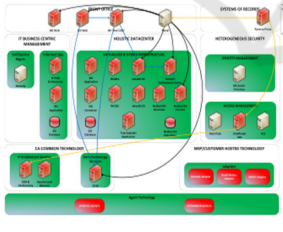
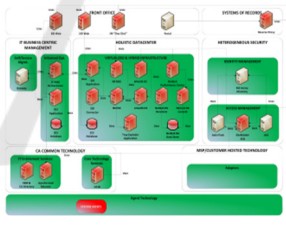
The CURA solution approach was created to handle three types of architecture needs: bundling unrelated technologies, integration, and new processes across products.

- Bundling unrelated technologies and co-locating technologies reduces IT footprint and unifies supporting infrastructure.
- Integrating technologies with a hub-and-spoke pattern or a canonical-data-model pattern ensures separation of concerns, where each product acts on and reacts with the data. The goal is to reduce integration points, standardize web service protocols, and provide common taxonomy for the integrated data.
- Creation of new processes provides a higher business value from a collection of separate products.

CURA targets complex system of systems solution architectures, as well as basic software systems and technologies. Instead of having different views without a binding structural thread, the CURA approach expands on top of a single structural skeleton that bridges multiple architectural aspects.

CURA unifies both R&D and Services architec-

Table 1: CURA views according to the four phases. Images are blurred in order to maintain confidentiality.

<p>Business</p>	 <p>4LHL Capability Blueprint</p>	 <p>Capability and Architecture Roadmap</p>
<p>Functional</p>	 <p>Capabilities to Business-Service-Innovations</p>	 <p>4LHL Capability-to-Technologies and Data Flow</p>
<p>Logical</p>	 <p>4x6 Logical Bricks Blueprints.</p>	 <p>4LHL Solution Logical Super-Position</p>
<p>Deployment</p>	 <p>4LHL Solution Security</p>	 <p>4LHL Solution Deployment</p>

ture discourse in terms of architecture notation, semantics, analysis and design techniques, and aligns the incremental deliverables within a single methodology and framework. CURA definition of DevOps expands from delivering a single cloud automated product, into a compound solution. CURA cultivates knowledge reuse, harvests information, reduces erroneous interpretation, and contributes to the cohesion of the professional community.

Agile or waterfall, incremental or iterative, DevOps or Continuous Delivery, CURA is applicable

for any approach.

## REFERENCES

Amber S, 2015. "Just Barely Good Enough" models and documents: an agile best practice, <http://www.agilemodeling.com/essays/barelyGoodEnough.html> Retrieved 22 December 2015.

Avgeriou P, Zdun U., Borne I, 2006. Architecture-centric evolution: new issues and trends. In *Proceedings of the ECOOP 2006 workshop reader (ECOOP'06)*.

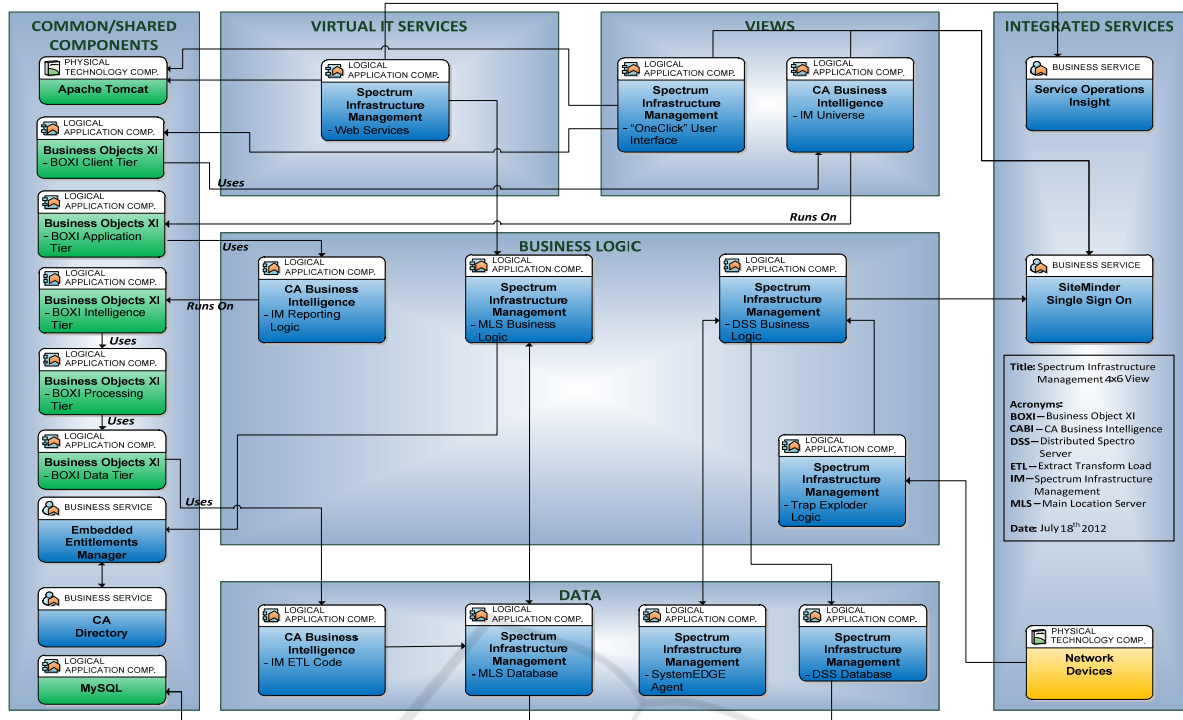


Figure 4: Example of a 4x6 blueprint – CA Technologies, Spectrum IM product (Created by Jason Davis for a enterprise level telecommunication company).

Hadar E., Davis J., Ferguson D. F., 2012a. "Proactive Performance Optimization of IT Services Supply-Chain Utilizing a Business Service Innovation Value Roadmap", *The Third International Conference on Cloud Computing, GRIDs, and Virtualization, Cloud Computing 2012*, July 22-27, 2012 - Nice, France.

Hadar, E., Hadar, I., Silberman G. M. and Harrison, J. M., 2012b. The 4x6 Tiered Architecture Method: An Approach to the Design of Enterprise Solutions, *CAiSE/IWSSA 2012 Workshops*, LNBIP 112, pp. 180-191. Springer, Heidelberg (2012), Gdansk, Poland, 2012.

Kruchten, P., 1995. Architectural Blueprints — The "4+1" View Model of Software Architecture. *IEEE Software* 12 (6), pp. 42-50, (1995).

Passos, L., Terra, R., Valente, M. T., Diniz, R., Mendonça, N., Buschmann, F., 2010. On Architecture Styles and Paradigms, *IEEE Software*, Vol. 27, Issue 4, 2010.

TOGAF, 2015 <http://www.opengroup.org/subjectareas/enterprise/togaf>, retrieved on December 22, 2015.