Research Reports in Software Engineering and Management 2010:01 ISSN 1654-4870

Architectural Concerns in Base Station Development

Lars Pareto

CHALMERS | UNIVERSITY OF GOTHENBURG
Department of Computer Science and Engineering

Research Reports in Software Engineering and Management No. 2010:01

Architectural Concerns in Base Station Development

Lars Pareto

Department of Computer Science and Engineering
CHALMERS | University of Gothenburg

Gothenburg, Sweden 2010

Architectural Concerns in Base Station Development

Lars Pareto

Report no 2010:01
ISSN: 1651-4769
Department of Computer Science and Engineering
University of Gothenbgurg and Chalmers University of Technology

Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden

Telephone + 46 (0)31-772 1000

Göteborg, Sweden 2010

Architectural Concerns in Base Station Development

Abstract

This report presents a catalogue of architectural concerns found to be important to stakeholders within Ericsson's Base Station development. The catalogue is based on interviews with software architects, designers, testers, and team leaders within the software development organization of Ericsson's W-CDMA base station development. It reflects concerns held by a representative sample of engineers (~1% of all project members), but is not a complete inventory of all concerns in base station development.

1 Introduction

The concerns catalogue presented in this document is an outcome of a case study carried out within Ericsson's division RBS SW during 2008-2009. Please refer to our past presentation of this study [1] for an introduction to the concerns catalogue and the methods and data sources on which it relies.

1.1 Reader Guidelines

The catalogue consists of a list of *architectural concerns* found in the study, along with definitions of these. Our definition of *concern* is that defined by IEEE1471:

> "*Concerns are those interests which pertain to the system's development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders. Concerns include system considerations such as performance, reliability, security, distribution, and evolvability.*"

Since its inception in 2009 [1], the catalogue has evolved, thus readers comparing this catalogue with our 2009 paper will find inconsistemcies. These are matters of presentation and of concern granularity: the underlying data sources, and the interests the concerns represent have remained the same. An extended version of the 2009 conference paper, consistent with the this document, has been accepted for publication in an in an upcoming issue of the Journal of Software and Systems Modeling.

1.2 Purpose and Scope

The intended audiences for this document are software architects, designers, managers and other stakeholders in software architecture documentation for telecommunication infrastructure components.

Concern names and concern definitions have been formulated to be readable both to Ericsson *outsiders* by the use of common software engineering vocabulary, and to *insiders* by also providing Ericsson internal terminology in brackets. Some telecom domain vocabulary, such as *base station* and *cell,* are inevitable in the concern definitions. Readers not familiar with the telecom domain may want to refer to an introduction to telecommunications systems, such as [2].

The list of concerns below consists of the concern's names (given in the section headings) and the concerns definitions (given in the body text).

1.3 Concern: actors and use cases

Requirements- and system engineers are concerned with the system's use cases, i.e., what actors interact with the system, what their goals are, and the system's black box behaviour in achieving these goals.

1.4 Concern: allocation independent client-server interaction

System engineers want views that emphasize the interaction between computational processes, and that abstract from where in the system (i.e., between which concrete components) the interaction occurs. Ultimate placement of client- and server processes in the system, may not be known at specification time but is done later (in the design phase or at run-time). System engineers want to specify allocation independent protocols, and then have all allocation specific protocol in the design model checked against the allocation independent protocols in the system model.

1.5 Concern: allocation protocols

Interfaces between clients and servers involve a functional protocol for interaction with the actual service and an allocation protocol, for establishing the actual service. These are separate concerns.

1.6 Concern: alternatives, conditions, and iterations in scenarios

Sequence diagrams not only express behaviour for a scenario, but also variations of the behaviour. For instance variation in hardware may call for alternate course of actions, as may backwards compatibility with software components. Engineers need to express these variations. (This concern is covered by the UML2 interaction operators alt, if, etc.)

1.7 Concern: annotating system model with links to project internal artefacts

While working with the system model, system engineers have a need to annotate the system model with links to project internal documents (such as planning documents). These temporary annotations are internal matters, and should neither be visible to other users of the system model, nor become part of the product documentation.

1.8 Concern: application decomposition

The base station considered provides two layered views of its resources that are externally visible for remote applications: the Operation and Maintenance (O&M) view; the Traffic Control view.

1.9 Concern: approval status

Team leaders want better integration of information, in the system model, of what components have been approved to which degree.

1.10 Concern: bandwidth

A base station is a data intensive computing system and the bandwidths on the many buses inside a base station are important for reasoning about performance and for allocating functionality to boards.

1.11 Concern: baseband use

A base station is a digital radio transceiver with algorithms to manage transmission and reception of packets in physical radio channels occupying slices of frequency spaces (basebands), at run-time.

1.12 Concern: baselines

There are many generations and variations of a base station product, thus a multitude of development branches (baselines) are needed. Which such branches exist is a concern for all roles.

1.13 Concern: basic and alternate flows

Use case based requirements engineering distinguishes the most common path through a scenario (basic flow) and optional behaviour outside the normally expected behaviour (alternate flows).

1.14 Concern: board configuration

The architecture recognizes several kinds of boards (main processor cluster, device boards, auxiliary units) holding various kinds of processors (main processors, board processors, device processors, peripheral processors, auxiliary processors, external processors, and non-processing units). These boards must all be configured at startup, and infrastructure for this, known as "device board configuration", exists. Board configuration includes a range of more specialized concerns.

1.15 Concern: board connections (wiring and buses)

Base station hardware consists of several boards connected by various buses and wiring. This perspective is also present in software development, because functions are distributed over boards, and because software is written to handle variations in board connectivity.

1.16 Concern: board interfaces

Hardware interfaces at the level of circuit boards are an essential view of a base station product. It allows reasoning about reuse of components across products.

1.17 Concern: board loading

System Engineers and designers are concerned with how hardware boards are loaded with software modules.

1.18 Concern: boards and their relationships to features

The conception of the system as a set of interconnected boards, some of which contribute to a certain feature is present, e.g. “delay handling for radio synchronization involves board x, y, and z”

1.19 Concern: cable configuration

Software components are shared across many base station products. These have different peripherals, and subsystems. Software needs to know what particular wiring a certain system has, or may have; this information is called a cable configuration.

1.20 Concern: call graph

To reason about what components call each other is important to designers, when trying to understanding the system.

1.21 Concern: capabilities

Capabilities are collections of related services (e.g., *setup cell*) into some area of functionality (e.g., *network-controller-ordered cell configuration*).

1.22 Concern: capability anatomy

A capability anatomy shows dependencies between capability groups (groups of groups of services), or between capabilities within one group.

1.23 Concern: capability decomposition

A capability decomposition is the taxonomy tree consisting of services, capabilities, and capability groups.

1.24 Concern: capability group filtering

Many views contain many components. Designers want projected (or filtered with Ericsson terminology) versions of these views that only contain components relevant to their specific subsystems. Capability groups is one area where filtering is desirable.

1.25 Concern: capability realization anatomy

Capability realization anatomy shows the dependency relationships between capability realizations. It is an abstraction of the dependencies between service realizations.

1.26 Concern: channel deployment (internal logical channels onto physical channels)

Telecom engineering distinguishes logical channels (that simply carry information packets) from physical channels (that carry information packets coded in ways suitable for radio signaling). A telecom node typically contains hundreds of logical channels multiplexed over the physical channels in intricate ways.

1.27 Concern: class and process instantiation (capsule instantiation)

As in any programming paradigm, run-time instantiation of a program is a concern to anyone reasoning about resource usage.

1.28 Concern: clean and precise visualization of requirements and behaviour

System models should ideally be an abstraction of the software (to be) built, yet precise enough to communicate how the system works. This is a difficult balance, and implementation level details (such as coding of messages and handshaking) tend to creep into the system model, thereby polluting the views that the system model is supposed to carry. System engineers are concerned with keeping the system model clean, yet precise.

1.29 Concern: climate

Base stations are engineered to function in extreme environments (e.g., deserts, tundra, mountains) and contain SW functionality to handle various climates.

1.30 Concern: communication paths

Allowed connectivity of subsystems depends on underlying HW. When realizing a new functionality on an existing system, or specifying a new system, system engineers need to reason about available or needed communication channels connecting the system's major parts.

1.31 Concern: communication paths between abstract ports

In some cases it is neither necessary nor practical to graphically show the connections between computational processes, as these would only clutter the diagram. In such cases, an abstract port (also known as unwired port) may be placed within the processes terminating the protocol carried on the port, to obtain an invisible connection. Designers are concerned with these “invisible” communication paths.

1.32 Concern: communication principles

Architecture for telecommunications network components, involve a range of architectural principles for communication, e.g., schemas for handshaking, link establishment, peer-to-peer addressing, and error-indication. These principles are abstract, and may take many forms depending on realization technology, e.g., principles have different incarnations for MDD components, for hardware blocks, and for CORBA-style objects.

1.33 Concern: complementary side perspectives

For understanding the system, it is essential that the system model / architecture documentation provides more than one perspective (cf. Ossher and Tarr's tyranny of the dominating perspective). For instance an engineer interested in how data is handled in the subsystems, should be able to navigate the system from this viewpoint (rather than searching through use case realization sequence diagrams for cues on how data handling is done). Yet side perspectives should preferably relate to the main perspective.

1.34 Concern: component behaviour

Component behaviour is what, in model driven development, is normally specified using state machines or activity diagrams.

1.35 Concern: connection establishment

Many of the communication links inside a base station are dynamic, and must be explicitly managed during runtime. On each dynamic link, the client and the server must negotiate the protocol revision to be used. System engineers are concerned with the infrastructure and conventions for this.

1.36 Concern: coupling and cohesion among service realizations and capability realizations

A service realization is a model of how objects interact to implement a specific service (typically in the form of a sequence diagrams), whereas a capability realization is a grouping of service realizations. Services typically depend on each other, and managing these dependencies with attention to coupling and cohesion among services as well as groups of such is a concern important to system engineers.

1.37 Concern: corporate basic standard for information assets (product structure with Ericsson terminology)

Ericsson has a corporate basic standard for information assets, including product related information. This structure (which recognizes entities such as systems, subsystems, blocks, interface products, and load modules) must be adhered to regardless of what other structures the architecture uses.

1.38 Concern: data - coding of parameters sent in signals

Testers and some designers are concerned with the precise coding of information sent in messages between computational processes.

1.39 Concern: data deployment (relationship between data and their storage)

Data storage may be distributed or centralized, and system design involves deciding where, on which subsystem parts, data is to be stored.

1.40 Concern: data - formal parameters in signals

Designers are concerned with what signal parameters there are, and what parameters mean.

1.41 Concern: data - what's static, dynamic, or persistent.

Base stations contain data bases for storage of persistent data. System engineers are concerned with specifying what data are persistent, volatile, or static.

1.42 Concern: decomposition of functionality into technical subsystems (functional parts with Ericsson terminology)

Any given functionality, defined by a use case, must be decomposed into interacting technical subsystems. These may be hardware entities, software entities, or combinations of both. (At specification time, it is not always decided whether function will be realized in hardware or software.) Technical subsystems are related to, yet distinct from the organizational subsystems, which are packages of function and responsibility used to pinpoint responsibility in the line organization, allocate functionality to design units, and to structure product documentation.

1.43 Concern: deliverables

RBS SW development ultimately results in load modules, i.e., binaries shipped with a product or a product upgrade. Load modules are associated with the RBS subsystems, and have an architecture of their own that associates several dozens of load modules with the many boards and processors of an RBS.

1.44 Concern: differences w.r.t earlier versions

Designers and other roles need to se what, in a diagram, has changed since an earlier version, and what has triggered the change.

1.45 Concern: distribution

A base station is a multi processor system. Distribution is the concern of what processors the various computational objects reside on.

1.46 Concern: division of work

All roles are concerned with who is working on, and who is responsible for, what. Software architecture serves as a backbone for structuring, documenting, and communicating such knowledge.

1.47 Concern: document names (carrying essential concepts)

Base station design documentation involves several hundreds of documents describing various aspects of the system (architecture reference documentation, design rules, subsystem documentation, function descriptions, functional specifications, interface products, and so on). The names carried by these documents turn out to be important to all stakeholders: in retrieving documents, in everyday personal organizations of documents, and in document searches. That documents are given regular and suggestive names (rather than document numbers) turns out to be a navigational aid in architecture work.

1.48 Concern: downtime

Downtime or outage duration is the period that an RBS fails to perform its primary functions. Downtime is a requirements area, and system engineers are concerned with the breakdown of downtime requirements on the system as a whole into requirements on individual system components.

1.49 Concern: error logging and logging (tracing with Ericsson terminology)

Error logging functionality helps localizing faults in an executing system. Detecting a malfunction is easy; the tricky part is the fault localization: to find the part of the system that caused the malfunction, and to understand why it occurred. This is supported by *logging* functionality, which supports generation of trace messages whenever something of interest occurs or to just monitor the behavior of the different parts of the system as they run. Special logging (tracing) infrastructure supports tracing per process or interface, and keeps track of trace groups and whether these are active. Logging is useful during debugging in the field, in lab testing, and in validation, and is of concern to architects, designers, testers, and operators.

1.50 Concern: external interface layering (protocol stack layering as of OSI)

Interfaces in high level models often refer to a certain protocol stack. To serve as specification, any interface must refer to a specific layer of the protocol stack.

1.51 Concern: external triggers

External triggers trigger services. An external trigger is a combination of messages and parameters on an external interface, where that combination has its own semantic meaning. External triggers are thus more abstract entities than messages (which can bee seen as a particular coding of the external triggers). External triggers appear as their own entities in system specifications.

1.52 Concern: fault handling

During startup, operation, or upgrade a number of run-time errors may occur (such as faulty hardware, missing wiring, non-responding connections, and so on). A base station has an extensive architecture that supports detection, localization, isolation, recovery, reporting, correction, verification, restoration, fault filtering, fault coordination, alarm suppression, state propagation, alarm subscription, alarm filtering, alarm distribution, alarm logging, alarm list administration, and alarm manipulation.

1.53 Concern: function onto block deployment (part with Ericsson terminology)

The process of adding a new feature to a base station involves mapping the features onto available system blocks (known as parts within Ericsson).

1.54 Concern: function onto code deployment

Designers are concerned with how a system level functionality (such as upgrade functionality, hardware restart functionality, software loading) maps to specific code modules.

1.55 Concern: function onto platform allocation

The base station design process involves H/W S/W co design with a hardware platform shared across many product lines. Only some of the requirements on a base station are allocated to software components developed by the S/W organization, whereas other are mapped to the platform developed by the H/W organization. This mapping is done by radio network systems engineers (which belong to a third organization within Ericsson). System engineers of the S/W organization are concerned with what functionality is and what functionality is not allocated to the platform of the next generation.

1.56 Concern: functional areas

Functional requirements on base stations are divided into functional areas such as Traffic, Operation and Maintenance, Fault Handling, Site Modifications, Node Preparations, and Factory Procedures.

1.57 Concern: functional deployment (logical- onto physical components)

The relationship between the logical components of the system model and the physical hardware components is of concern to system engineers, architects and designers.

1.58 Concern: hardware architecture

The hardware architecture is of concern to designers for at least four reasons: 1) hardware is the means by which software is executed, and software must be design with respect to hardware capabilities (i.e., processing power, and available communication channels); 2) software shall provide operation and maintenance interface to the hardware, thus software shall provide a view of the base station's hardware; 3) software should support hardware fault handling, and hardware maintenance operations; 4) software should be configurable to various hardware configurations. In all these areas, software engineers are deeply entrenched in details of the hardware architecture.

1.59 Concern: hardware start/restart (equipment restart)

A base station has a requirement area concerned with start/restart of the base station involving timing and synchronization setup, radio equipment startup, infrastructure equipment setup, and setup of signaling to neighbour nodes.

1.60 Concern: hardware variants

Base Stations are shipped in a number of variants, with varying features and platform configurations. Designers and testers need to reason about these variations.

1.61 Concern: how a board works

A base station includes a multitude of special purpose boards for radio signaling, antenna control, power management, signaling to neighbour nodes, and so on. To realized specified functionality (or to correct errors), designers need to understand not only the software interfaces, and general capabilities of the boards, but also how the board operates.

1.62 Concern: how manipulation of managed objects affects functional resource objects and things below.

Managed objects are equipment resource objects (functional resource objects with Ericsson terminology) made visible to telecom network management software through standardized interfaces. The internal relationships between managed objects and their underlying resource objects is an essential concern in the design of a base station.

1.63 Concern: hyperlinking and tagging

Engineers frequently relate diagram elements (such as messages in sequence diagrams) to other sources of information using tagging or hyperlinking.

1.64 Concern: information flow routing between boards

Designers are concerned with how information flows through the base station's many boards, in particular what information flows are interrupted if a specific board fails.

1.65 Concern: integration of descriptive texts in models

A view supports coding of information from a given perspective using a given modeling notation. This is often not good enough: engineers sometimes feel constrained by the view, and want to elaborate on some aspect of the model in prose; engineers also want that what they write will be clearly visible to anyone looking at model—just as illustrative diagrams go hand in hand with the text flow in a traditional sequential document, descriptive text in models should go hand in hand with the model trees and diagrams of model based documentation. With contemporary MDD tools, such complementary "text views" are not as visible as text is in traditional document based engineering: although text appears in generated reports, text is often "hidden away" in the model.

1.66 Concern: interaction between technical subsystems (functional parts with Ericsson terminology)

Interaction between technical subsystems is what is typically defined by a sequence diagram.

1.67 Concern: internal interface layering

Interfaces between system components may be different in the detailed design models than in the system models. What appears as a plain data record in the system model may correspond to streaming a series of bit frames, with retransmission signaling, in the detailed design models. System engineers, designers, and testers are all concerned with the correspondence between signalling at these different levels of abstraction.

1.68 Concern: introduction- and context

Introduction and context is a crosscutting concern that recognizes the role of architecture documentation in learning and understanding the system. In the same way as security aspects are important to some stakeholders and thus should be visible across relevant views, introductory- and contextual information is important to learners, and should be visible (or reachable) across relevant views too.

1.69 Concern: layer-to-layer vs. peer-to-peer signaling

In a layered architecture, there is a distinction between messages sent between components at the same system layer (peer-to-peer signaling) and messages sent between layers (layer-to-layer) signaling. Clear distinction between these two kinds of signaling, is important to keep the architecture description clean and easy to understand.

1.70 Concern: load module structure (program execution handling decomposition)

Load modules are executable software entities (executable machine code for the system's CPU:s, .jar files, database schemas, .html files, fpga machine code, DSP machine code) loaded into the system during startup or upgrade phases. Each subsystem has a load module structure that relates load modules for the systems various software components (equipment handling, sector & cell handling, channel handling, node control, databases, http servers, JVM configuration, device boards, auxiliary units) to specific locations / destinations for these.

1.71 Concern: locality (of reference)

A base station is a multi processor systems situated in a distributed telecommunication system. Storage involves processors boards, device boards, auxiliary units, databases, volatile memories, flash memories, and storage on external nodes. Real time software design in this context is concerned with where in the system data is stored, and what acess/uppdate times are needed and achievable.

1.72 Concern: logical resources

A RBS provides two views: the operation and maintenance view; the traffic control view. Each of these views is layered, in 9 distinct layers. One of these layers is concerned with logical resources at a level of abstraction below the actual services provided by the RBS and above the actual devices used to realize the service.

1.73 Concern: managed object (MAO/FRO/RO) domain

The *managed object model* is the view of the base station seen by telecom operators. Major entities in this view are Managed Adaptation Objects (MAOs), used to raise the abstraction of the system to a level suitable for administration, Facade Resource Object (FROs) which are used for configuration, data administration, parameter validation, storage/retrieval and calculations of usage state, and Resource Objects (ROs) which are device-driver-level components that provide the actual services.

1.74 Concern: major subsystems

The major software subsystem components and their interconnections (at the 1st whitebox level) are of particular importance to project- and line managers, as this view is often a basis for division of work, responsibility, and resource allocation.

1.75 Concern: memory capacity

New base station software oftens needs to be compatible with old hardware, with the implication that designers must assert that software components with added features will run on older boards without running out of space.

1.76 Concern: memory utilization

Dynamic memory is allocated and freed during run-time. Thus the amount dynamic memory in use varies during execution. The total size of the dynamic memory (heap and pool) is defined at build time and is fixed. The peak value of allocated dynamic memory must not exceed the size limit. The peak allocation is dependent on both static and dynamic load. The peak can occur when the node is running at its maximum static capacity or dynamic capacity or at some combination of dynamic and static capacity.

1.77 Concern: naming conventions

Naming conventions are important to make models, as well as artefacts generated from models, readable and automatable. Conventions cover assigned *prefixes* and *number intervals* for services, subsystems, signals, message parameters, system parameters, process names, data base tables, file names, alarms, error messages, events, triggers, and logs.

1.78 Concern: overviews

One important role of design documentation is to provide overviews of the system: the "big pictures" serving as background for understanding the existence, design, precise behaviour, and characteristics of some specific components.

1.79 Concern: performance

Performance is a central area of concern in any real time system: it involves execution times, response times, background loads, static capacity, static load, dynamic capacity, and dynamic load characteristics during start, restart, or operation, and infrastructure for performance management, i.e., measuring performance through counters throughout the system.

1.80 Concern: planning

Project management is concerned with who is realizing/updating/correcting/merging what components when.

1.81 Concern: platform-as-layer vs. platform-as-component

Two views of the role of a platform is exists in the system studied: in the *platform-as-layer view*, a platform is as lower architectural layer used to realize components in a higher layer using to layer-to-layer communication; in the *platform-as-component view* a platform provides certain functions useful across products, but appears as peers to system components at arbitrary levels of abstraction in the system model. (Both these views prevail in telecom standards.) Architects are concerned with how the platform appears in the architectural views, and how it should appear in use case realization sequence diagrams and structural diagrams.

1.82 Concern: power supply and distribution

A RBS has a sophisticated power supply system capable of handling outages and handling of setup/release/supervision of the power and power backup system, through built in-in algorithms as well as through a management interface.

1.83 Concern: process deployment (relationships between Processes, Threads, and CPUs)

MDD tools provide light weight processes that are mapped onto OS-threads of some underlying real-time operating systems, possibly running on different CPUs. This mapping is normally specified using properties in the model, whereas the actual mapping is automated. The mapping is a concern to designers.

1.84 Concern: process interface inheritance (capsule interface inheritance)

Capsule is the term for concurrent processes used by the modeling tool in use. System engineers and designers are concerned with inheriting interfaces of capsules higher up in the process hierarchy, when realizing sub capsules of these.

1.85 Concern: process priorities

Process priorities is a concern in reasoning about real-time system properties, such as capacity, scheduling, performance, and worst case execution time.

1.86 Concern: process/subprocess decomposition (capsule / subcapsule decomposition)

Capsule is the term for concurrent processes used by the modeling tool in use. Capsule / subcapsule decomposition is a structural breakdown of the systems in terms of a hierarchical network of processes communicating using protocols. This is the dominating perspective of the system model.

1.87 Concern: processor load

Designers and testers are concerned with the computational load of the base station's many processors (i.e., CPUs and DSPs) on its many boards.

1.88 Concern: protocol abstraction (several messages at a lower level becoming one message at a higher level)

More details are often desirable in design level protocols than in system level protocols. For instance a system level protocol may include a single transfer of a piece of information, whereas the design level contains messages for resource allocation, handshaking, error indication, etcetera. Engineers want to model at several such levels, and have the protocols formally related, so that inconsistencies may be detected, and so that degree of realization of the system level protocols may be measured.

1.89 Concern: protocol exemption (in behavioural descriptions)

The design guidelines of the studied base station specify generic protocols, i.e., communication patterns that system components must follow. These are instantiated in sequence diagrams (describing use case realizations), adding a lot of redundant detail in system specification. What is far more important to designer is where these generic protocols are not adhered to, but modified. Such deviations are motivated in certain situations, but give a more brittle design. Being able to search out where exemptions to protocols have been made is important to designers.

1.90 Concern: protocols between technical subsystems (functional protocols with Ericsson terminology)

The protocol concept in the project studied originates in the Room methodology. Protocols define what signals may be sent and received on ports (similarly to provided and required interfaces in UML2.) Protocol definition for internal protocols is a major task for system engineers, as well as designers. (External protocols are given by telecom standards.)

1.91 Concern: RBS resource handling

The mechanisms for providing, allocating, and de-allocating resources provided by the hardware layer are called *resource handling.* Several domains of resource handling exist: dynamic device resource handling, equipment resource handling, and baseband resource handling.

1.92 Concern: RBS variables (parameters with Ericsson terminology)

With Ericsson terminology, *parameters* are a changeable/readable attributes of entities in a system. Changing the values of a parameter will affect the characteristics/behaviour of the system. Parameters may be updated/read through operation and maintenance signaling as well as traffic control signaling.

1.93 Concern: redundancy

Base stations are fault tolerant systems that should maintain operation also in the presence of hardware failure. Some components are more critical than others, and six classes of criticality (redundancy types) are identified. System engineers map the base station's boards and units onto these classes.

1.94 Concern: relationship between hardware-interfaces and software interfaces

Interfaces in the base station's architectural view are sometimes hardware interfaces, and sometimes software interfaces. Architects are concerned with which are which, and how certain SW interfaces in one view relate to a HW interfaces in another view.

1.95 Concern: response time

Response time is the time from the reception of a request until the response is sent. It is dependent on CPU load and the speed of external units. Response time is a concern in many contexts, e.g. auxiliary unit response, device board response, O&M message response, and update response.

1.96 Concern: responsibility

Responsibilities in organizations often hinge onto architectural views, e.g. a subsystem in the model is often seen as a packaging of functionality and responsibility; *subsystem responsible* and *interface owner* are established roles.

1.97 Concern: schedulability

For some time critical computations, designers are concerned with whether real time tasks are schedulable within available time frames.

1.98 Concern: service anatomy

A *service anatomy* shows dependencies between services within one capability group.

1.99 Concern: service or use case realization (in terms of functional parts)

A *use case realization* specifies how whitebox objects collaborate to implement to a use case, whereas a *service realizations* specifies how objects collaborate to implement a specific service. Use case- and service realizations are normally specified using sequence diagrams, but other behaviour diagrams are also used. Service and use cases realizations are the same concern appearing at different levels of system function granularity. (Use cases are decomposed into services.)

1.100 Concern: signal grouping and bundling

At the design level, connected components typically communicate through *several* connectors, ports, and protocols. Showing all these entities in the system model easily makes it too cluttered. Rather, system engineers prefer to reason about (and draw) cohesive bundles of connectors---this is sometimes referred to a signal bundling---or single connectors that represent such bundles.

1.101 Concern: startup

Designers, architects, and system engineers are concerned with what is started when in which order during the base stations *startup* phase.

1.102 Concern: synchronization

Synchronization denotes handling of delays and delay variances in the transmission of data packets. Three domains of synchronization, (each with their own detailed problems and solutions): *node synchronization* concerned with transmission of user data to and from antennas and neighbour nodes, *frame synchronization* concerned with data packets sent internally within the RBS, and *network synchronization* concerned with Ethernet packets sent to and from neighbour nodes.

1.103 Concern: system services

A service is a black-box concept that describes a part of a use case. A service can be seen as an operation provided by the system and described by one or more black-box steps in the use case flow. A service is triggered by a signal on an external interface and may send signals on any external interface. Instances of services are the messages found in the UML sequence diagrams that realize the use case scenarios. Generally, there is an n-to-m mapping between use use-cases and services.

1.104 Concern: system traces and system model consistency

System engineers and testers want to be able to reach, view, and compare *system traces* relevant to the interaction specified in the *system model's sequence diagrams* from within the system model itself.

1.105 Concern: system/subsystem decomposition (with Ericsson specific meaning)

The main characteristics of an Ericsson subsystem are: 1) it is used to pinpoint responsibility in the line organization; 2) it is used to allocate derived requirements on; 3) it is a functional product in Ericsson product information management system; 4) it is a grouping of functionality. Subsystem decomposition thus carries several concerns important to engineers as well as managers, and is more than a matter subdividing a system into technical subsystems.

1.106 Concern: test coverage

Testers measure to which degree a component has been tested, and managers are interested in this information.

1.107 Concern: test-system

Testing of base station software and hardware relies on extensive product test infrastructure for integration, verification, and validation of software as well as hardware. Many parts of the system model and the design models play a role in this infrastructure. Accurate use of models relies on these roles, i.e., test system is a concern.

1.108 Concern: tier dimensions and layers

The studied architecture has a multi dimensional layering, that involves the follwing dimensions:1) *requirement models layer* | *system model layer* |*design model layer* | *implementation model layer* | *code layer*; 2) *presentation layer* | *service layer* | *adaptation layer* | *resource layer*; 3) resources related layers; 4) board related layers; 5) *radio related layes*; 6) interntet related layers; 7) processor related layers.

Architects are concerned with what such dimensions are used, how each dimension is segmented into layers; and what the relationships between the components in all layers are.

1.109 Concern: traceability

Traceability, i.e., is which components that realize a specification (*forward traceability*) and which specifications a component originates in (*backwards traceability*) is important in many situations. For instance, designers are concerned with where, in the system model, a particular requirement is being realized. Designers also want backward traceability from the design level components (functional parts) to the places in the system model, in system documents, or in the requirements models, that motivate the existence of these components. This is to make the design easier to relate to the system model and to the requirements.

1.110 Concern: traffic control

Traffic control is a major area of functionality defined by telecom standards, such as *UTRAN Iub interface Node B Application Part (NBAP) signaling*. Functional areas of traffic control are cell configuration management, common transport channel management, system information management, resource event management, configuration alignment, measurements on common resources, radio link management, radio link supervision, compressed mode control , measurements on dedicated resources, downlink power drifting correction , reporting of general error situations, physical shared channel management, downlink power timeslot correction, cell synchronization , information exchange, bearer re-arrangement, multimedia broadcast multicast service notification, user equipment status notification, and exchanging information about the secondary uplink frequency.

Although many concerns related to traffic control are covered by external standards, it is sometimes advantageous to cover them also in the system model.

1.111 Concern: transaction handling

The *transaction* is well known concept in databases and distributed systems research. A transaction is a collection of operations representing a unit of consistency and recovery. A transaction starts by initializing things, then reading and modifying objects, then at the end either committing or saving any changes made or aborting any modifications (leaving the system in the state before initialization). Transactions appear in the management and operation interface; transaction handling is supported by the RBS platform.

1.112 Concern: upgrade

New versions of base station software are released frequently and remote upgrades common. *Upgrades* must be done with as little interference as possible (as downtime is very expensive for operators) and with support for rollback (should the upgrade go wrong). To support this, base stations have extensive upgrade architecture involving *upgrade packages* that bundle new software- firmware- and description-modules for a node with an update script, and an *upgrade engine* that executes upgrade packages and monitors the upgrade process.

1.113 Concern: usage scenarios

Usage scenarios are constituents of a use case specification. One usage scenario describes the main flow, whereas other describe alternate paths through a use case.

1.114 Concern: verifiabilty

System engineers are concerned with *requirement verifiability*, i.e., that it is possible to write test cases for the requirements.

1.115 Concern: virtual hardware

Virtual hardware is an approach to software and documentation reuse. All Ericsson products are associated with a product structure, which is used to organize end user documentation, to order spare parts, to report errors, and for long-time archiving of product information. This product structure is board oriented. When evolving a system, it is highly beneficial to maintain this structure, even though underlying hardware may have changed, e.g., four boards being replaced by one. By the use of virtual boards in the architecture, surrounding software as well as documentation structure may be kept.

1.116 Concern: what is automated

The model considered involves extensive *automation*, e.g., consistency checking, propagation of information, code generation. Engineers are concerned with what automation is built into the modeling infrastructure, and what relationships between modeling artefacts that the automation brings.

2 References

[1] L. Pareto, P. Ericsson, S. Ehnebom, *Concern Visibility in Base Station Development – an Empirical Investigation,* in Model Driven Engineering Languages and Systems, Lecture Notes in Computer Science, 2009, Volume 5795/2009*, 196-210,* Springer Verlag

[2] Heikki Kaaranen, *UMTS networks: architecture, mobility, and services,* John Wiley and Sons, 2005