Health-Watcher:
Architecture Description

version 0.h

Coequalizer Corporation

April 2, 2010

Abstract

This document is an *architecture description* (AD) for the HEALTH-WATCHER case study [8]. The AD is intended as a worked example of an architecture description conforming to the requirements of IEEE Std 1471:2000 (also ISO/IEC 42010:2007).

Version history

v0.h Fixed some bibliographic references.

v0.g Cleaned up concerns, added a metamodel. (dated 1 Mar 2010)

v0.f Minor cleanups to concerns, added details to viewpoints: capability metamodel, trust metamodel (dated 17 Feb 2010).

v0.e This version updated to use the terminology of ISO/IEC CD1 42010 (dated 25 Jan 2010) such as model kinds, correspondences. The document still conforms with IEEE Std 1471:2000 (aka ISO/IEC 42010:2007). Viewpoint definitions are elaborated. Views are further refined.

v0.d Goal is to baseline the stakeholders and concerns in this version, and move on. Everything is open to comment!

Open issues

1. OPEN: Re: FR16 (HWRD p17). Is this change currently logged in employee?
2. OPEN: Does system target a single computer or not (HWRD p18–19)?

3. OPEN: Requirements spec doesn't say *why* system should be able to be distributed? Performance? Security? Just for Fun? (HWRD p19)

4. OPEN: Storage medium (HWRD p19)?

Comments on this document

This AD is a work-in-progress intended as a concrete example of an architecture description conforming to the requirements of IEEE 1471 (ISO/IEC 42010). Please submit comments or suggestions for improving this example to:
r.hilliard@computer.org.

1 Introduction

1.1 Purpose

This document describes the architecture of the HEALTH-WATCHER system [8]. Its purpose is to capture key decisions about the system design, its capabilities, technology choices, to support the system's identified use cases and requirements, and communicate those decisions to the relevant system stakeholders.

> Remark: Text in this format are *meta comments about this AD* – not part of the AD itself. Remarks are used to discuss issues under investigation, highlight and discuss conformance and other features of the standard, and so on.

1.2 Scope

The scope of this AD is the HEALTH-WATCHER system as specified by the requirements document [8].

> Remark: The AD also introduces some additional architectural concerns and *change cases* to make the case study more interesting architecturally.

1.3 Context

This AD conforms to IEEE Std 1471:2000, *Recommended Practice on Architectural Description of Software-Intensive Systems* [5] (also ISO/IEC 42010:2007).

Remark: In addition to conforming to the published standards cited above, the example uses the updated terminology of the draft revision, ISO/IEC CD1 42010 (dated 25 Jan 2010).

1.4 Overview

Section 2 identifies the HEALTH-WATCHER's system stakeholders and their system concerns. Section 3 establishes the architecture viewpoints being used in this AD. Sections 5 through 10 are the resulting architecture views.

Acknowledgments. Thanks to ... for comments on earlier versions.

2 System stakeholders and system concerns

In this section, we identify the HEALTH-WATCHER's system stakeholders and their system concerns which drive the HEALTH-WATCHER architecture.

2.1 HEALTH-WATCHER Stakeholders

Citizens use the system to make complaints and obtain health information.

Staff are responsible for system operation and provision of its resources. There are three classes of staff employees that are users of the system: attendants, inspectors and managers.

Client owns the HEALTH-WATCHER requirements as interpreter and final arbiter of those requirements. The Client controls the budget for developing and operating the system.

Local Government approves the system for use; and insures it is in compliance with applicable local laws and regulations.

Implementors have responsibility to build and maintain the system.

Sanitary Surveillance System (SSVS) will exchange data with HEALTH-WATCHER (future).

Remark: In accordance with (IAW) the standard (1471: 5.2/CD1: 5.3), the AD must identify the system stakeholders relevant to the architecture; consideration to include (as applicable): Users, Acquirers, Developers, Maintainers.

2.2 HEALTH-WATCHER system concerns

The following system concerns dominate the HEALTH-WATCHER architecture.

SC 1 *How do I interact with the system?*
HELD BY: *Citizens and Staff*
SOURCE: *NFR:4.1 (Usability)*

SC 2 *What services does this system provide?*
HELD BY: *Citizens, Staff, Implementors*

SC 3 *How is my privacy protected?*
HELD BY: *Citizens*

SC 4 *How does system help me use it?*
HELD BY: *Citizens, Staff*
SOURCE: *NFR:4.1 (Usability)*

SC 5 *How is the system to be constructed?*
What technologies are used where?
HELD BY: *Implementors*
SOURCE: *NFR:4.6 (Hardware and software), NFR:4.7 (Distribution), NFR:4.8 (User Interface), and NFR:4.9 (Storage medium)*

SC 6 *What standards must implementation follow?*
What standards are needed for information exchanges?
HELD BY: *Implementors, Local Government*
SOURCE: *NFR:4.5 (Standards)*

SC 7 *Will system comply with applicable regulations?*
HELD BY: *Local Government*
SOURCE: *NFR:4.5 (Standards)*

SC 8 *What faults and exceptional situations must the system anticipate?*
HELD BY: *Client*
SOURCE: *NFR:4.2 (Availability)*

SC 9 *What elements can fail?*
How do they fail?
HELD BY: *Implementors*
SOURCE: *NFR:4.2 (Availability)*

SC 10 *How are exceptional situations, and faults handled?*
What are mitigations?
HELD BY: *Implementors, Client*
SOURCE: *NFR:4.2 (Availability)*

SC 11 *What resources constrain system operation?*
HELD BY: *Client*
SOURCE: *NFR:4.3 (Performance), NFR:4.6 (Hardware and software)*

SC 12 *What happens when resources reach saturation?*
What are the consequences of each such situation?
HELD BY:
SOURCE: *NFR:4.2 (Availability)*

SC 13 *What threats can compromise the system?*
SOURCE: *NFR:4.4 (Security)*

SC 14 *What system resources must be protected against known threats?*
HELD BY:
SOURCE: *NFR:4.4 (Security)*

SC 15 *What trust mechanisms are to be used?*
HELD BY:
SOURCE: *NFR:4.4 (Security)*

SC 16 *How, and why, is the system distributed?*
HELD BY:
SOURCE: *NFR:4.7 (Distribution)*

SC 17 *What runs where?*
HELD BY: *Implementors*

SC 18 *What communication media and protocols are used?*
HELD BY: *Implementors*

SC 19 *What dependence on commercial networks/service providers? What are the QoS points (SLAs) that must be negotiated?*
HELD BY: *Client*

SC 20 *What "things in the world" does the system need to know about?*
HELD BY: *Implementors, Staff*

SC 21 *How is information managed and kept consistent?*
HELD BY: *Staff*

SC 22 *Can system bear required load (20 simultaneous users)?*
HELD BY: *Client*
SOURCE: *NFR:4.3 (Performance)*

SC 23 *Can system meet required response times (5s for queries)?*
HELD BY: *Client*
SOURCE: *NFR:4.3 (Performance)*

SC 24 *What are the important end-to-end timings?*

SC 25 *Can system meet availability (24x7)?*
HELD BY: *Client, Staff*
SOURCE: *NFR:4.2 (Availability)*

SC 26 *Are system requirements met by this architecture?*
HELD BY: *Client*

SC 27 *What is its cost of ownership?*
HELD BY: *Client*

Remark: In accordance with (IAW) the standard (1471: 5.2/CD1: 5.3), the AD must identify the system concerns relevant to the architecture; consideration to include (as applicable): system purpose, appropriateness of architecture to achieve purpose; feasibility of construction; risks; maintainability, deployability and evolvability.

Remark: The AD must associate each Concern with the Stakeholders hold it (CD1: 5.3). The metamodel in figure 1 also links (some) Concerns to their sources in the Requirements. TBD: *This could be presented in a table of Concern – Stakeholder – Viewpoint – Source*

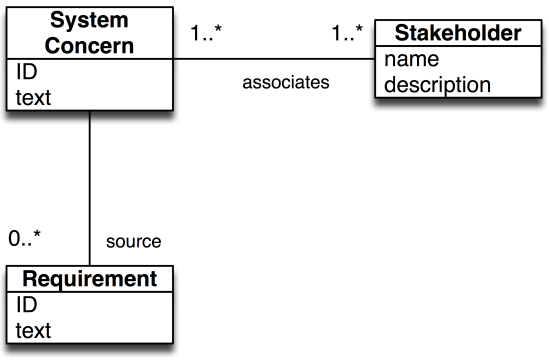

Figure 1: Stakeholders – Concerns – Requirements metamodel

2.3 Open issues

1. OPEN: These concerns are not yet framed by any viewpoint: SC-7, SC-25, SC-26, SC-27
2. OPEN: How is access to Complaint Registration controlled?
3. OPEN: Need to unpack *Usability* (Users and Implementors)
4. OPEN: What are the concerns on latency?
5. OPEN: What overflow capacity is there?

3 HEALTH-WATCHER architecture viewpoints

The HEALTH-WATCHER architecture will be expressed using the architecture viewpoints defined in this section.

> Remark: Viewpoints need not be defined in the AD, but may be included by reference (e.g., to an organization's standards and procedures). This set of viewpoints is probably overkill for the small HEALTH-WATCHER system, but it allows us to illustrate several topics in applying the standard.

TBD: *Update viewpoints format to CD1 requirements, template if needed, these changes will be backward compatible with the IEEE 1471:2000 compliance requirements.*
TBD: *Add rationale for each viewpoint per CD1: 5.8.1.*

3.1 Information viewpoint

3.1.1 Overview

The Information Viewpoint is used to describe the real world things that HEALTHWATCHER needs to know about, how those entities are represented as data within the system, and how that data is managed so that implementors know what to build, users get what they need, and the system is able to exchange data with other systems.

3.1.2 Concerns framed by viewpoint

Data exchange: SC-6; things in the world: SC-20; information management: SC-21.

3.1.3 Viewpoint conventions

Information views consist of 3 interrelated model kinds:

Conceptual model: what are the key kinds of information and their characteristics; ownership, CRUD rules, entity lifetimes.

Data access model: how data is accessed and exchanged with other systems.

Data storage model: how data is persisted and physically managed.

Figure 2 shows how the three model kinds relate. The Data Access and Data Store models *classify* the elements of the Conceptual model—i.e., they deal with classes of elements expressed in that model, based on relevant properties (e.g., all elements that are personally identifying information). Similarly, the Data Access model refers to the Data Store model in classifying how various storage types are accessed.

Figure 3 illustrates the metamodel for the Information viewpoint. In the interests of space, the metamodel does not present the metamodel for the Conceptual Data model portion which is standard stuff: assume that it has Entities, Relations, Attributes, and so on. So the parts of the Conceptual Data model are elided into a cloud in the figure. Elements of the Conceptual Data model will be depicted using UML classes, associations and attributes, respectively.

Data Stores and Data Access (instances) realize conceptual data items (elements, relations, attributes). Instances of type Data Access encapsulate access to Data Stores.

Correspondence rule: Every Data Store and Data Access (instance) is implemented in the Capability view (as a type of Capability).

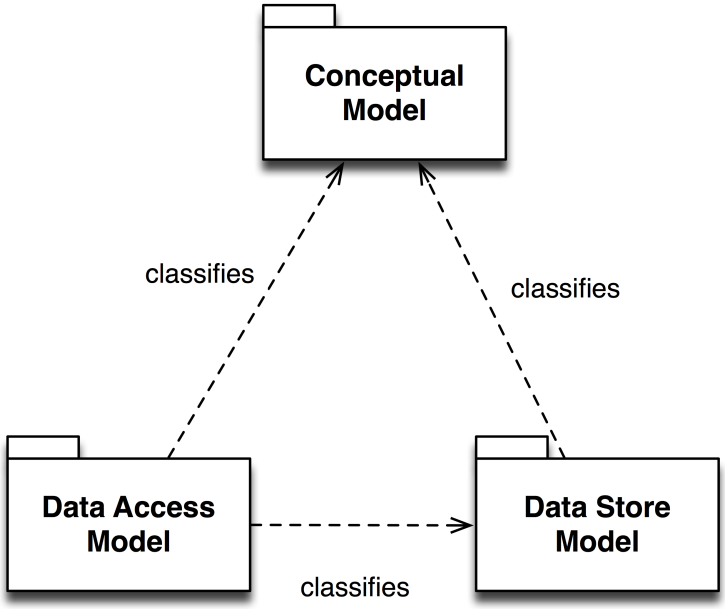

Figure 2: Overview of Information viewpoint model kinds and their relationships

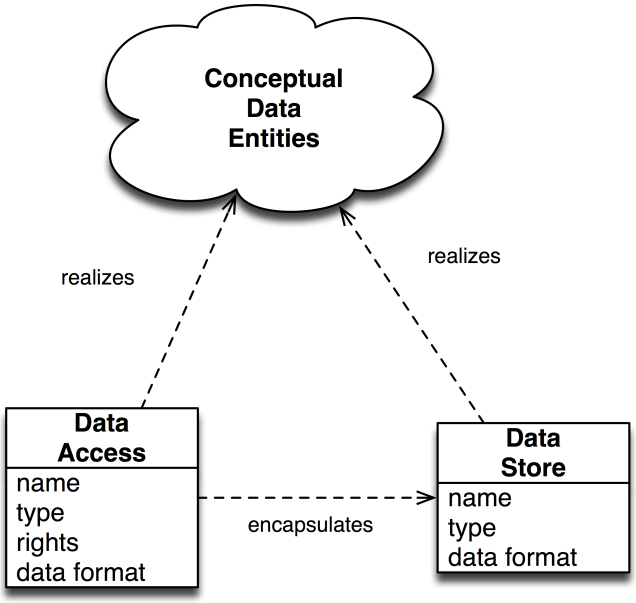

Figure 3: Information viewpoint metamodel

3.1.4 Sources

The Information viewpoint is further described in [3, 9].

3.2 Capability viewpoint

3.2.1 Overview

The Capability viewpoint is used to frame concerns such as: What does the system do? How is the system built? What technologies are used where? The Capability viewpoint is built from the components and connectors viewpoint defined in [1], specialized with additional component (*Capabilities*) and connector types, refined by an additional model kind to used to associate capabilities with user-required functionality derived from the requirements document.

3.2.2 Concerns framed by viewpoint

interaction: SC-1; services: SC-2; help: SC-4; construction and technologies: SC-5; standards: SC-6

3.2.3 Viewpoint conventions

This view is documented using components, their ports and named interfaces and connectors. Components can be classified as: data elements, processing elements, interaction elements (ports), state elements.

Figure 4 shows the Capability metamodel.

Correspondence rule: Capabilities are linked to the user (functional?) requirements they are intended to satisfy.

TBD: *Relations between C2 elements (such as temporal, causal, input, output, include, import/export, inherit...)*

Figure 5 summarizes the symbology used by the Capability viewpoint. Sometimes artistic license is taken to replace entities with pictorial icons.

TBD: *Analyses: chaining of dependencies [Stafford]; data flow and control flow [1, p117].*

3.2.4 Sources

The Capability viewpoint uses components and connectors as primary elements [1]. See also: [2].

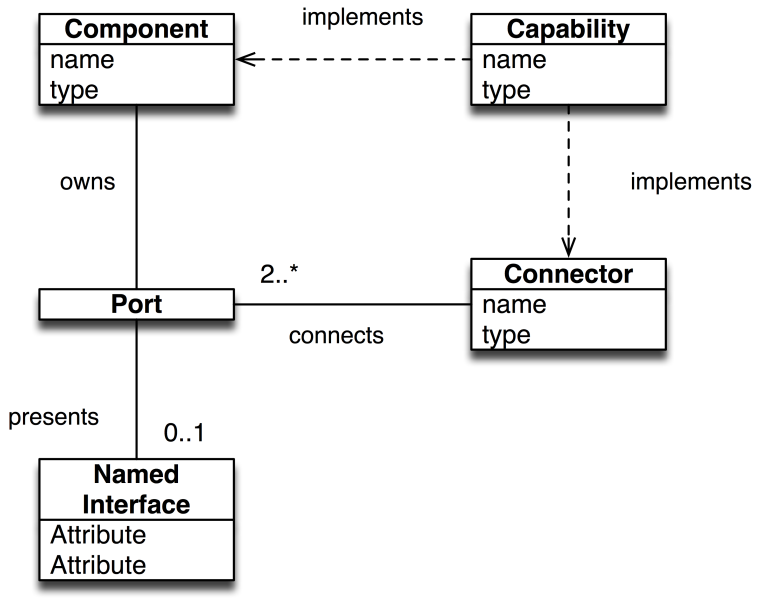

Figure 4: Capability metamodel

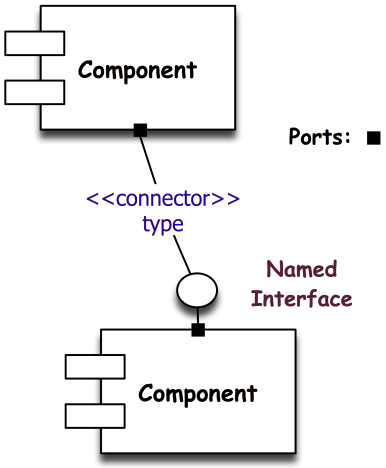

Figure 5: Symbology of Capability viewpoint

3.3 Trust viewpoint

3.3.1 Overview

The Trust viewpoint frames system concerns such as: confidentiality, availability, integrity and privacy; the security and privacy principles to meet trust goals and requirements, and the mechanisms for implementing those principles and mitigate those threats.

3.3.2 Concerns framed by viewpoint

citizens' privacy protection: SC-3; threats, risks and hazards to the system: SC-13; protection of system resources: SC-14; trust mechanisms and mitigations employed: SC-15;
TBD: *Availability SC 25*

3.3.3 Viewpoint conventions

The Trust viewpoint employs two model kinds: the threat model and the trust model. These will be depicted diagrammatically and sometimes in tabular forms.

Figure 6 illustrates the Trust metamodel.
TBD: *Add to metamodel elements of the trust model kind:* principals *(or subjects); resources (or objects);* operations *(ways in which principals interact with resources);* policies *(the rules which specify, for each principal and each resource, what operations that principal is allowed to perform on that resource); and* trust domains *(ensembles of principals and resources which interact at the same level of trust). Domains may have different trust levels, and interact with other domains.*

3.3.4 Sources

See [4] for definition of Trust viewpoint used here. See also [7, 9].

3.4 Quality-of-service viewpoint

3.4.1 Overview

TBD: *Too much bundled into one viewpoint. Need to take Trust viewpoint into account, which already tackles threats and hazards.* Refactorings welcomed!

The QoS Viewpoint frames those concerns which affect the ability of the system to deliver its services, including qualities such as usability, availability, performance and resource utilization.

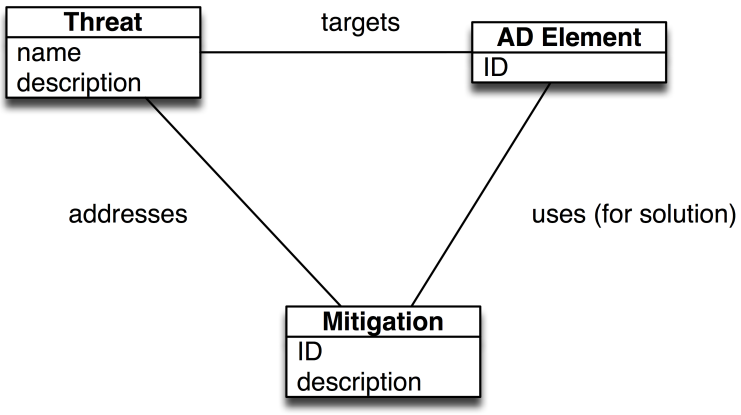

Figure 6: Trust metamodel

3.4.2 Concerns framed by viewpoint

faults and hazards: SC-8, failures: SC-9, fault management: SC-10, resource constraints: SC-11, SC-12, load: SC-22, response time: SC-23, end-to-end timing: SC-24
TBD: *availability: SC 25*

3.4.3 Viewpoint conventions

TBD: *This is a mixed bag of diagrams and tables. It should share threat model with trust view.*

3.4.4 Sources

TBD: *Various*

3.5 Deployment viewpoint

3.5.1 Overview

Deployment addresses issues such as: How is the system physically distributed? What communication media does it use? This viewpoint is also an input to analyses of trustworthiness, reliability and performance.

3.5.2 Concerns framed by viewpoint

how: SC-16, where: SC-17, protocols: SC-18, external service providers: SC-19

3.5.3 Viewpoint conventions

Entities: components, connectors from capability mapped to platforms and other devices.

3.5.4 Sources

Originally defined by Kruchten as the "physical view" in [6], the notation and semantics used here is as defined in UML.

3.6 Scenarios viewpoint

3.6.1 Overview

The Scenarios viewpoint is used to depict the key scenarios which the architecture of the system must anticipate. These include *use cases*, *change cases* and *stakeholder (or quality*[1]*) cases*.
TBD: *Use also for tracing to requirements*

3.6.2 Concerns framed by viewpoint

TBD:

3.6.3 Viewpoint conventions

Use case diagrams are OK for this, with enhanced semantics for changes cases, and to allow other more general interactions, "stakeholder cases".

3.6.4 Sources

Kruchten's original $4 + 1$ paper defines the role of scenarios [6].

3.7 Meta (Big Picture) Viewpoint

3.7.1 Overview

This viewpoint has two purposes: (1) it has as its entities the other viewpoints in this AD. It is intended to show how they are interrelated. This orients the reader of the AD to how to look for answers to questions about the architecture. (2) It is a very simple viewpoint, and thereby shows how a viewpoint declaration and its resulting view is structured and specified. The view language of the meta viewpoint is

[1] See Bass 2003, as cited in E and C, 173ff.

Figure 7: The "Big Picture" of HEALTH-WATCHER AD

very simple. There is one kind of entity: used viewpoint. There is one kind of relationship "uses". It is a directed relationship. Each entity, and each relation instance has a name/label. Figure 7 shows the "big picture" of the HEALTH-WATCHER architectural description.

3.7.2 Concerns framed by viewpoint

TBD: *There may not be any specific stakeholder system concerns for this*
How do the views relate?
What correspondences should/must hold between views?

3.7.3 Viewpoint conventions

3.7.4 Source

Each architectural concern identified in 2 is framed by the viewpoints shown in Table TBD.

4 Big Picture View

TBD: ...

5 Information view

TBD: *Topics to address in Information view: CRUD, Information life cycles: particularly Complaints; persistence; data transmission, serialization? ACID, data integrity*

5.1 Conceptual model

The key information-bearing entities of relevance to the system: Complaints, Licenses, Diseases, Health Units, Specialties, Employees, Certificates.
HEALTH-WATCHER owns Complaints, Birth/Death Certificates.
SSVS owns Sanitary License Applications.

OPEN: Who owns Health Info: Vaccination Info, Disease Prevention, Health Guides? Are these imported from elsewhere? Written by Staff? How do they get into the system?

5.2 Data access model

All access to HEALTH-WATCHER data is implemented as Capabilities (see 6) via data access layer. Data access capabilities determine data access rights, based on user access scenarios in HWRD.

Two classes of data access capabilities: Query and Update, based on rights to query or change system data.

TBD: *Transaction semantics*

Data access layer manages exchanges with other systems (such as SSVS) and handles the transform of relational data to XML where needed (see User Interface, Ajax).

TBD: *Update Capability view with Data Access (and Data Store) capabilities*

5.3 Data storage model

All data is stored relationally.

Applications have no direct access to data. All access through Data access layer using JDBC. Data stores are organized by major abstraction in the Conceptual model: Complaint, Staff, Symptom, Health Unit, Specialty, Disease.

6 Capability view

6.1 Components and connectors

There are two ways users may interact with the system: via web browser or via telephone.

On the web side, the user interface is created with two component types Ajax pages (Asynchronous JavaScript and XML) and Java servlets. See 6.2 for further specification of their details. Standard connections used are: http, RMI, JDBC and (vanilla) local file system access for storing web and servlet data.

Figures 8 and 9 depict the components, their interfaces and connectors of the HEALTH-WATCHER.

TBD: *How to depict Ajax pages. Currently shown where they are executed (by browser/user agent).*

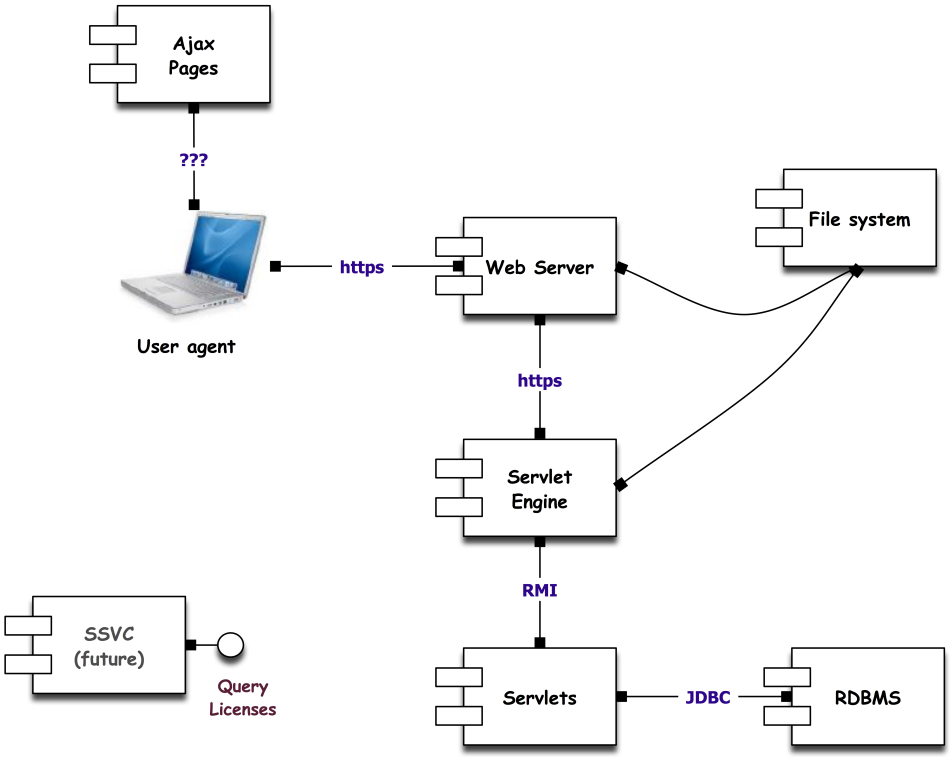

Figure 8: HEALTH-WATCHER Capability: web side

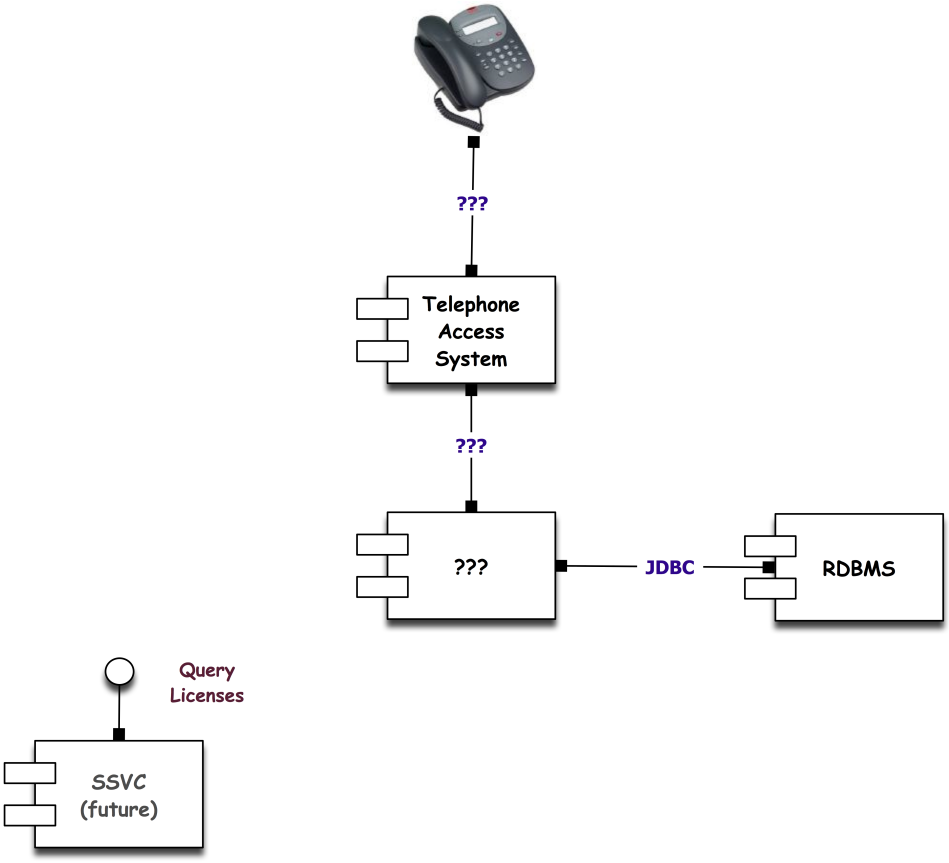

Figure 9: HEALTH-WATCHER Capability: call center side

Remark: The multiple figures (models) demonstrate that the requirement – each view must represent the *whole* system – need not entail that only a single diagram must be used.

6.2 Capabilities

Delineation of capabilities, packaged functionality, tracing back to the HWRD.

Citizen applications:

- Citizens Portal
- Query Health Units by Specialty
- Query Specialties by Health Unit
- Register Complaint
- Query Complaint
- Query Disease Information
- Online Help

Employee applications:

- Login Employee
- Register New Employee
- Update Employee
- Update Information
- Update Complaint
- Change Logged Employee

TBD: *Add above items as type Application (capability), in between Servlets and RDBMS.*

Ajax Pages: at least one will be needed per capability.
Servlets: one per interaction
Online Help, Complaint Control, Health Information,
Error handling: should refer to the Faults identified in Trust view, threat model.

6.3 Open issues

1. OPEN: Who knows about Call Centers (i.e., "Telephone Access Systems")? Possible tradeoff: although the system is not particularly application-rich in functionality, should functionality be moved out of servlets to avoid duplication with telephone side?
2. OPEN: How to represent servlets?
3. TBD: *Each capability/application also ought to have a Manager interface, by which DIEVS staff access applications and data.*
4. TBD: *Interface to SSVC: queryLicenses*
5. FUTURE: Interface to SSVC.ComplaintControl: sanitarySurveillanceComplaints

7 Trust view

7.1 Threats and risks

Table 1 depicts the HEALTH-WATCHER threat model.
TBD: *Develop correspondences between threats and other AD elements in other views to capture the target/vulnerabilities column.*

7.2 Trust model

TBD: *Principals: Resources: Policies: Information Domains: Mitigations and Mechanisms:*

8 Quality-of-service view

TBD: *usability, availability, response time, ...*

8.1 Mitigations

response time, availability, user load

Communications Error: raise error TBD: *to whom?*

Data Retrieval Error: retrieve available information; raise error TBD: *to whom?*

Data Entry Error: raise error to user

Data Consistency Error: Abandon data retrieval (roll back changes); raise error TBD: *to whom?*

Data Storage Errors: Abandon storage action (roll back changes); raise error TBD: *to whom?*

Authentication Error: Abandon current action; raise error TBD: *to whom?*

Citizen Privacy: access controls

Complaint Integrity: access controls

OPEN: Link each item in Mitigation to a element in threat model 1.

9 Deployment view

What's distributable? Where? How? Distribution: remote access, consequence: remote exceptions,

10 Scenarios

This view comprises use cases, changes cases, and (maybe) stakeholder cases. The requirements document defines the following use cases, we will not repeat their details here.

FR01: Query information (Citizen).

FR02: Specify complaint (Citizen).

FR10: Login (Employee).

FR11: Register tables (Employee).

FR12: Update complaint (Employee).

FR13: Register new employee (Employee).

FR14: Update employee (Employee).

FR15: Update health unit (Employee).

FR16: Change logged employee (Employee).

Change Case CC01 (Future): Funding for current system has been approved, and first release of system must be built against the approved requirements document. However, developers want to change user interface to exploit AJAX. Therefore, first release should be designed to minimize impact of this expected change, while meeting current requirements.

Change Case CC02 (Future): Current query of disease information is by disease name (FR01:2.4, p7). What is involved in providing interactive Disease Determination by symptom dialog?

References

[1] Paul C. Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, and Judith Stafford. *Documenting Software Architectures: views and beyond*. Addison Wesley, 2003.

[2] DII–AF Chief Architects' Office. *The Air Force's Command and Control System Target Architecture version 1.0*, 1998.

[3] David E. Emery, Rich Hilliard, and Timothy B. Rice. Experiences applying a practical architectural method. In Alfred Strohmeier, editor, *Reliable Software Technologies – Ada-Europe '96*, number 1088 in Lecture Notes in Computer Science. Springer, 1996.

[4] Rich Hilliard. A trust viewpoint. http://mysite.verizon.net/rfh2/writings/hilliard-TrustVP-r1.pdf, March 2009.

[5] IEEE. *IEEE Std 1471–2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, October 2000.

[6] Philippe B. Kruchten. The "4+1" view model of architecture. *IEEE Software*, 12(6):42–50, November 1995.

[7] Butler W. Lampson. *Computers at Risk*, chapter Requirements and Technology for Computer Security, pages 74–101. National Academy Press, Washington, 1991.

[8] Tiago Massoni, Sérgio Soares, and Paulo Borba. Requirements document health-watcher, version 2.0. http://aosd.di.fct.unl.pt/ea-icse2007/documents/Health_Watcher_Usecase_v2_1.pdf, October 2006.

[9] Nick Rozanski and Eoin Woods. *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perpectives*. Addison Wesley, 2005.

Contents

Threat or Fault	Description	Target / Vulnerabilities	Intent	Priority
COMM	Communications between system elements are interrupted.			
RETR	Error occurs while attempting to retrieve data from a Health-Watcher Data Store			
ENTR	Data attempting to be entered into the Health-Watcher system is invalid, incomplete or redundant			
STOR	System is unable to store data			
AUTH	System cannot validate user			
PRIV	Personal identifying information of a complainant or victim is compromised by unauthorized access.			
COMP	The target of a COMPLAINT attempts to delete or modify the complaint.			

Table 1: HEALTH-WATCHER Threat model