An Application of the Architecture-Based Design Method to the Electronic House

Felix Bachmann
Len Bass
Mark Klein

September 2000

SPECIAL REPORT
CMU/SEI-2000-SR-009

Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

An Application of the Architecture-Based Design Method to the Electronic House

CMU/SEI-2000-SR-009

Felix Bachmann
Len Bass
Mark Klein

September 2000
Product Line Systems

Unlimited distribution subject to the copyright.

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

Joanne Spriggs

Joanne E. Spriggs
Contracting Office Representative

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Copyright 2000 by Carnegie Mellon University.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use. Requests for permission to reproduce this document or prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

Abstract **ix**

1 Introduction **1**

2 Business Context (Handout 1) **3**
- 2.1 Market Niche/Role 3
- 2.2 Critical Functionality 4
- 2.3 Critical Quality Attributes 4
- 2.4 Constraints 5

3 Abstract and Concrete Functionality (Handout 2) **7**
- 3.1 Goals 7
- 3.2 Abstract Functional Requirements for CNS 7
- 3.3 Abstract Functional Requirements for the Home Integration System 8

4 Abstract and Concrete Quality Attributes (Handout 3) **11**
- 4.1 Goals 11
- 4.2 Abstract Quality Requirements 11

5 Architecture Options (Handout 4) **13**
- 5.1 Goals 13
- 5.2 Options for Architectural Mechanisms 13

6 Architectural Drivers (Handout 5) **15**
- 6.1 Goals 15
- 6.2 Architectural Drivers 15

7 Encapsulate Function (Handout 6) **17**
- 7.1 Goals 17
- 7.2 Requirements 17
- 7.3 Result 18

8 Choose Architecture Style (Handout 7) **21**
- 8.1 Goals 21

8.2 Requirements 21
8.3 Result 22

9 Allocate Functionality to Style (Handout 8) 25
9.1 Goals 25
9.2 Background 25
9.3 Result 25

10 Refine Templates (Handout 9) 31
10.1 Goals 31
10.2 Design Elements 31
10.3 Result 32

11 Verify Functionality (Handout 10) 35
11.1 Goals 35
11.2 Use Cases 35
11.3 Change Scenarios 35
11.4 Result 35
11.4.1 Call to Turn on Air Conditioning 35
11.4.2 Add New Device to CNS 36
11.4.3 Change Communication Mechanism from COM to CORBA 37
11.4.4 Add New Device 37

12 Generate Concurrency View (Handout 11) 39
12.1 Goals 39
12.2 Use Cases 39
12.3 Notation 39
12.4 Result 39
12.4.1 Start-up 39
12.4.2 Two Simultaneous Users 40

13 Generate Deployment View (Handout 12) 43
13.1 Goals 43
13.2 Processor Configuration 43
13.3 Deployment 43
13.4 Two Users Simultaneously Access a Device 45

14 Verify with Quality Scenarios (Handout 13) 47
14.1 Goals 47
14.2 Quality Scenarios 47
14.3 Result 47

References **49**

List of Figures

Figure 1: Basic Architecture Structure 16

Figure 2: Three-Tier Structure 23

Figure 3: Allocation of Encapsulated Function to Style 26

Figure 4: Use-Case Map for Call Arriving to Turn on Air Conditioning 36

Figure 5: Use-Case Map for Adding New Devices to the CNS 37

Figure 6: Use-Case Map for Start-Up 40

Figure 7: Use-Case Map for Two Simultaneous Users 41

Figure 8: Two-Processor Configuration 43

Figure 9: Deployment on the Given Hardware Configuration 44

Figure 10: Use-Case Map for Two Simultaneous Users on Deployed System 46

List of Tables

Table 1: Abstract Functional Requirements for the CNS 7

Table 2: Astract Functional Requirements for the Home Integration System 8

Table 3: Abstract Quality Requirements 11

Table 4: Options for Architectural Mechanisms 13

Table 5: Transformation of Requirements to Responsibilities 18

Table 6: Grouping of Responsibilities 19

Table 7: Prioritizing the Quality Requirements 22

Table 8: Requirement for the Decomposed Design Elements (Allocate Functionality to Style) 26

Table 9: Information Interface for Decomposed Design Elements 30

Table 10: Templates for Decomposed Design Elements 32

Table 11: Definition of Units of Deployment 44

Abstract

The Architecture-Based Design (ABD) Method is a method for designing the software architecture of a product line of systems. It has previously been described in the technical report, *The Architecture Based Design Method* (CMU/SEI-2000-TR-001) [Bachmann 00]. This report elaborates an example of the application of this method to designing the software architecture. The example is the house of the future.

The house of the future is assumed to have a collection of devices within the house that are controlled by a computer network. Entertainment, security, heating/air-conditioning, and utility devices will all interoperate and will be controlled from a central network. The software architecture to support the house must be extendible and flexible, and it must have high security, high performance, and high availability. In this report, we present a first-level decomposition of the software architecture as a demonstration of the ABD Method.

1 Introduction

This report is intended to provide an example of the use of the Architecture-Based Design (ABD) Method. The actual use of ABD has always been on proprietary systems, so to provide an example for public consumption, we have made up this example about the "house of the future."

The ABD Method assumes that both functional and quality requirements have been specified. These requirements, in turn, depend on the business basis for the system being designed. We thus include a description of the business case for BBK (Bachmann, Bass, and Klein). BBK is the company building systems for the house.

The steps of the ABD Method, once requirements have been specified, are as follows:

1. Identify architectural drivers.
2. Divide (encapsulate) functionality.
3. Choose architectural style.
4. Allocate functionality to style.
5. Refine templates.
6. Verify functionality.
7. Generate concurrency view.
8. Generate deployment view.
9. Verify quality scenarios.

In this report, Chapters 2, 3, and 4 are devoted to the business case and requirements, Chapter 5 is devoted to the identification of potential architectural mechanisms, and Chapters 6-13 exemplify the steps above. We have couched each chapter as an exercise to be handed out so that it could be used in conjunction with a course that includes the ABD Method. For each course, the instructor must decide whether each handout is to be an exercise or simply provided to the students. In every case, the first handout is not an exercise but should be provided to the students.

2 Business Context (Handout 1) ABD Example—Home Integration System

Business context includes market niche and market role, critical functionality, critical quality attribute, technical constraints, and business constraints.

2.1 Market Niche/Role

BBK intends to be a major player in the projected multi-billion dollar market for “wired homes.” BBK is well known for systems requiring specialized home wiring such as security systems, and it intends to exploit this reputation and build the electronic/computational infrastructure for the home of the future. This infrastructure is the heart of the future electronic home, allowing devices within the home to communicate and allowing the home to communicate with external entities. While every “wired home” will have different features, all will require this infrastructure, which is called a home central nervous system (CNS).

BBK also builds home integration systems, which use the CNS. BBK will often bundle its own devices with its CNS to encourage customers to buy its devices. BBK will also sell complete home integration systems (HISs), buy devices that it does not make itself, and integrate these devices with its CNS and devices.

BBK builds HISs for installers and is not in the business of installing systems itself. Homebuilders either subcontract to HIS installers or employ their own HIS installers.

BBK has a reputation for high-end, very high-quality systems. Installers of BBK HISs, and in some case homebuilders often market their own products using the phrase “rest assured when BBK is at home.”

HISs are expected to be long lived. A homeowner would typically expect the system to last the lifetime of the home, but wealthy homeowners might replace a system while living in a house.

The initial market will be for new, high-priced houses. An HIS can cost as much as five percent of the cost of the completed house. As HISs become more common, lower priced versions will target less expensive existing homes. Existing homes imply design compromises as they use non-standard devices and employ a sub-optimal hardware infrastructure.

In the future, BBK expects to produce less expensive models and exploit the anticipated very large upgrade market. BKK will market its HISs worldwide.

2.2 Critical Functionality

BBK's CNS is a distributed operating system (OS) for the home, providing device drivers for many of the interfaces to home devices such as telephones, radios, TVs, home security systems, personal computers, personal organizers, etc. BBK caters to their own devices by providing optimized interfaces, which allows for better performance and more functionality than the standard interfaces. However, BBK's CNS will support all the major standard device interfaces.

BBK will build either the bare CNS, will bundle the CNS with its own security and heating, ventilation, and air conditioning (HVAC) devices, or will provide a full functionality system using its devices in addition to other companies' devices.

BBK will also build a product known as "SmartHouse," which provides a rule base and a rule-based editor that describes smart interactions between devices.

An easy-to-use installer's interface must also exist.

2.3 Critical Quality Attributes

In this section, we enumerate the important quality attributes for the CNS and the HIS.

Interoperability: It is critical for BBK's CNS to interoperate with other vendors' devices and device drivers. The standards in this domain are evolving, and conformance with the standards is not sufficient to ensure interoperability.

Modifiability

- The system must be modifiable to migrate to apartment buildings, cheaper homes, and existing homes.
- The system must also be modifiable to allow for upgrades as devices change and as new devices are made.
- The system must either initially support or be easily evolvable to a product-line framework
- Vendor independence is key—BBK wants to avoid being locked into any specific vendor (e.g., BBK systems should not be locked into Wintel, Oracle, etc.). The system must be easily portable to other platforms and be able to exploit the most cost-effective commercial off-the-shelf (COTS) products available.

Usability

- The system should support normal humans operating it. This means that the system must be tolerant of human errors, and simplicity of operation is preferred to efficiency of human operation. That is, fewer keystrokes are not necessarily better.

Security

- The CNS must support authentication for both remote and local users for access to various capabilities.
- The CNS must support confidentiality, as the system might have multiple users with different levels of permission simultaneously using the system.

Reliability

- The CNS must be at least as reliable as the attached devices and must not diminish the reliability associated with using those devices (e.g., the telephone does not become less reliable because it is now accessed via the CNS).
- The failure of any device cannot compromise the rest of the system.
- Devices must still be able to work when the CNS fails.
- The CNS must be fail safe.
- The availability of the CNS varies from 90% to 99.9999%, depending on the product instance.

2.4 Constraints

No constraints for the system designer have been identified as a result of the business case.

3 Abstract and Concrete Functionality (Handout 2)

ABD Example—Home Integration System

3.1 Goals

- Define the abstract functional requirements for the CNS.
- Translate the business context into a collection of abstract functional requirements.
- Make these requirements concrete by turning them into use cases (not a portion of this exercise).
- Break down the requirements into those for the HIS and those for the CNS.

3.2 Abstract Functional Requirements for CNS

Table 1 shows the abstract functional requirements for the CNS. These requirements have been grouped into categories for ease of exposition.

Table 1: Abstract Functional Requirements for the CNS

Remote and local access	• Support local and remote access to devices. • Support multiple access modes (devices) such as wireless (Palm Pilot), direct (PC), telephone (PC), telephone (voice), Web (PC), etc. • Display device status.
Smart integration	• CNS will support customizable smart interaction between devices.
Installer's user interface	• Add devices. • Remove devices. • Support an interface for customization of smart integration rules. • Configure the CNS for various feature sets: – Read floor plan – Network configuration
Home owner's user interface	• Check configuration including the features supported, devices attached, warranty dates for the CNS and each device, etc. • Monitor house state including who is in what room, windows and doors that are open, current mode (and mode definition), permissions, etc.

Table 1: Abstract Functional Requirements for the CNS (cont.)

Diagnosis	• Activate CNS diagnosis. • Activate diagnosis for selected devices.
Modes	• Define modes such as night mode, security mode, and diagnosis mode. • Activate modes manually or automatically.
Basic device support	• Turn on/off. • Receive status. • Reset. • Start self test.

3.3 Abstract Functional Requirements for the Home Integration System

Table 2 shows the abstract functional requirements for the home integration system.

Table 2: Abstract Functional Requirements for the Home Integration System

Oven/stove/microwave management	• Turn on/off. • Set timer. • Query status of food being cooked.
Refrigerator/ food management	• Detect food as it's put into the refrigerator. • Detect food as it's taken out of the refrigerator. • Specify requirements for food in the house. • Place an order (with optional manual intervention) over the Internet when food levels go below a certain level. • Remotely change the temperature setting for freezer or refrigerator. • Access Web-based cookbook and automatically check if ingredients are in the house.
HVAC management	• Turn on/off AC, heater, humidifier, attic fan. • Display current environmental parameters such as temperature, humidity. • Set timers. • Change settings as people come or leave rooms. • Notify home owners regarding maintenance schedules. • Schedule maintenance with device vendors • Open/shut curtains based on how sunny the day is, the house temperature, and the thermostat setting. • Check if there are logs in the fireplace. • Remotely light gas fireplace. • Display utility usage/cost history.

Table 2: Abstract Functional Requirements for the Home Integration System (cont.)

Telephone management	• Display messages in textual form or via voice. • Input important phone numbers and transfer those calls to wherever the person happens to be. • Set up conference calls. • Display image of called parties on monitor of choice.
Household management	• Home office maintenance: Order expendables such as paper and ink cartridges, when low. • Set alarm clocks in kids' rooms. • Speak to anyone anywhere in the house. • Turn on/off water sprinkler. • Car sends a signal to the house when it's leaving work, which triggers events like change house temperature, turn on oven, followed 20 minutes later by microwave. • When everyone goes to bed, turn off the lights, check for locked doors, set house temperature. • Reminders: – Pay bills – with manual invention. – Take out the garbage.
Security	• Door bell rings differently and announces who is at the door. • Monitor who is in the house. • Query who is in the house. • Turn on/off alarm system.
Home entertainment management	• Speaker follows individual through house. • Program the taping of TV. • Order (with manual intervention) advertised products on TV. • Display variety of artwork (changes periodically). • Display digital wallpaper (changes periodically). • Sources of images displayed as artwork or wallpaper can be varied such as from the Web, from photographs, or from elsewhere. • House takes pictures of daughter and baby, which are viewable remotely. • House communicates with another house and displays views from second house.
Remote tracking	• Cars • Children • Borrowed items such as books

4 Abstract and Concrete Quality Attributes (Handout 3)

ABD Example—Home Integration System

4.1 Goals

- Define the abstract quality requirements for the HIS.
- Translate the business context into a collection of abstract quality requirements.
- Make these requirements concrete by turning them into quality scenarios (not a portion of the exercise).

4.2 Abstract Quality Requirements

Table 3 shows the abstract quality requirements for the home integration system.

Table 3: Abstract Quality Requirements

Modifiability	• Device-related modifications – Add/remove known device types. – Add/remove new device types. – Home owner vs. installer modifications – Dynamic vs. static modifications • Portability-related modifications – Move to new OS. – Move to new middleware. – Move to new central processing unit (CPU). • CNS-related modifications – Change from a wire-based to wireless network. – Use power lines for local area network (LAN). – Customize the “smart” interactions.
Security	• Physical intrusion prevention, detection, and reporting – Select room to look at. – Lock/unlock doors remotely. – Remotely position/zoom camera. – Determine who is where in the house. – Identify who is allowed in the house. – Take action for those not allowed in house. – (Re)configure security settings (e.g., when owners of house change). • Electronic intrusion prevention, detection, and reporting – Authentication – Protecting personal data – (Re)configuring security settings (e.g., when owners of house change)

Table 3: Abstract Quality Requirements (cont.)

Safety	• Detection and automatic reporting of safety violations – Fire, carbon monoxide, gasoline fumes, radon, etc. • Authentication for safety-related appliances
Reliability	• Any functions having to do with stove, oven, utilities, fire, and security must be very reliable. • Fail safe
Performance	• User interface performance • Infrastructure performance • Device performance
Interoperability	• BBK devices working with non-BBK devices
Usability	• For installer • For home owner • For children • Acknowledgement for commands • Independent verification • Various input/output (I/O) modes (speech, text)

5 Architecture Options (Handout 4)
ABD Example—Home Integration System

5.1 Goals

- Define potential architectural options to be used to achieve each of the qualities.
- For each quality attribute, enumerate architectural mechanisms or styles that might be used to achieve that mechanism.
- Do not make a commitment at this point.

5.2 Options for Architectural Mechanisms

Table 4 shows, for each quality attribute identified in Table 3, possible architectural mechanisms that might be used to achieve that attribute.

Table 4: Options for Architectural Mechanisms

Modifiability	• Virtual interfaces for various devices types • Layering for portability • Rule-based approach for customizing “smart” interactions • Three-tiered architecture
Security	• Firewall— single point of entry to the CNS • Encryption • Authentication— digital signature
Safety	• Authentication for using safety-related appliances
Reliability	• Analytic redundancy for reliability (simple reliable hardware backups for safety-related systems • Redundancy between the CNS and devices when reliability is key (e.g., the CNS monitors messages to the oven to determine if it turns off within time specified for timer setting).
Performance	• Ability to support single multiprocessor configurations • Support for priority-based communications
Interoperability	• Device interface standards • Message format standards • System-wide quality attribute models
Usability	• Support for Undo, cancellation, etc. • No side effects

6 Architectural Drivers (Handout 5) ABD Example—Home Integration System

6.1 Goals

- Determine architectural drivers
- Examine requirements with goal of determining those that are the most influential in determining the architecture

6.2 Architectural Drivers

There are several architectural drivers for the HIS.

- One fundamental distinction is between the CNS and the associated devices. All of the devices are treated as attachments to the CNS.
- Another fundamental characteristic of the architecture is that other vendors will construct and install applications; hence, there must be a separation between the CNS and the other applications.
- A final fundamental driver of the architecture is the need for security and protection against rogue applications written by organizations other than BBK. This can either be accomplished through a certification mechanism or through a variety of firewall. Certification would have to be done by BBK or some third party and, as yet, there is no such certification agency. BBK does not wish to be responsible for certification, so there should be some protection or isolation mechanism that insulates the CNS from the other applications.

This leads to an architecture that has the basic structure shown in Figure 1. The users interact either with the CNS or with the devices, and the CNS is the path to other applications that are a portion of the HIS, but not a portion of the CNS. Between the CNS and the other applications is a protection or isolation mechanism so that rogue applications cannot harm the CNS. For the remainder of this example we will focus on just the CNS.

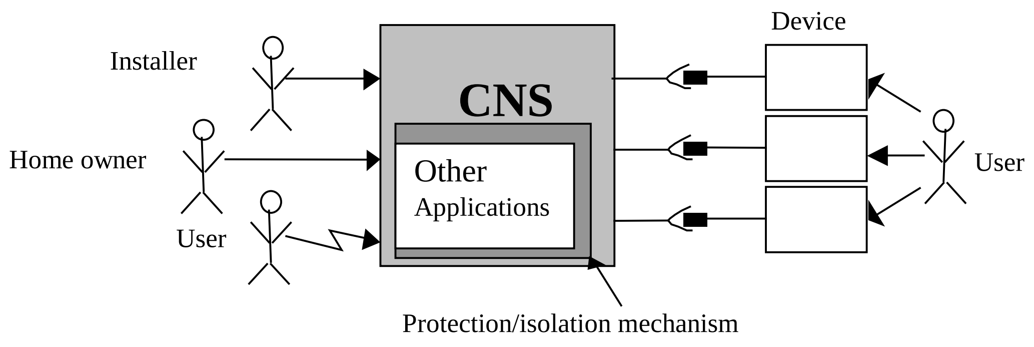


Figure 1: Basic Architecture Structure

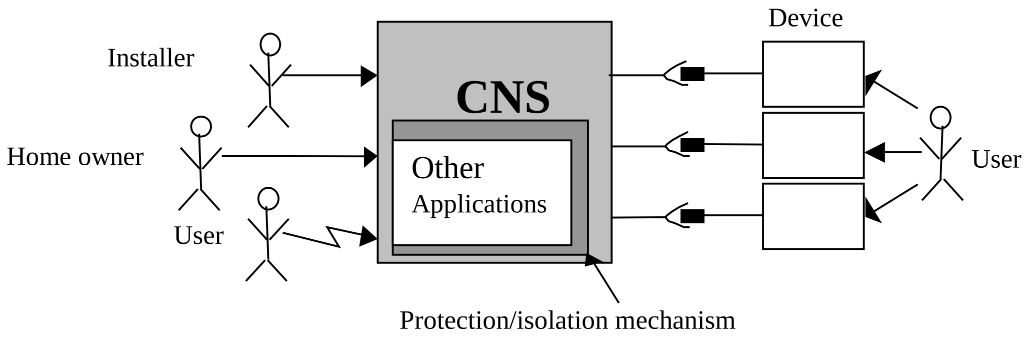
7 Encapsulate Function (Handout 6) ABD Example—Home Integration System

7.1 Goals

- The functional requirements for a design element (what to do) are translated into responsibilities of the design element (how to do it).
- This is done considering the constraints that have been assigned to the design element.
- The responsibilities of a design element are refined and organized into groups that may represent distinct elements within the architecture.
- The goal of this decomposition is to produce groups of functionality of sufficient granularity to
  - represent a decomposition of the design element responsibilities
  - keep the number of groups intellectually manageable (3 to 15)
  - support goals of modifiability

7.2 Requirements

We have repeated Figure 1 (the basic architecture structure) below for context. The design element on which we are focusing for this step is the central nervous system (CNS), which is the part of the home integration system (HIS) that is wired into the house. The CNS provides the computer power and the connectivity needed for the various HIS applications. The CNS also provides a software platform for HIS applications from BBK and other vendors.



Applications that are essential to operate the HIS (such as the configuration, diagnosis, and basic device operation for the most common devices) are part of the CNS. This also includes a user interface for the installer and a basic user interface for the homeowner.

Other applications and/or user interfaces can be installed as HIS applications on the CNS platform, but they are not considered to be part of the CNS. Control software for additional devices can be installed as part of the CNS.

7.3 Result

First we transform the requirements from Table 2 into responsibilities as shown in Table 5.

Table 5: Transformation of Requirements to Responsibilities

Requirements	**Responsibilities**
• Support local and remote access to devices. • Support multiple access modes (devices) such as wireless (Palm pilot), direct (PC), telephone (PC), telephone (voice), Web (PC), etc. • Display device status.	• There exists a communication application program interface (API) that makes available the same functionality to all connected devices (local or remote) as to a local user interface.
• CNS will support customizable smart interaction between devices.	• Devices come with a set of functions that are defined in a way that a rule engine can interpret it. • Rules can be defined any time that describe interactions between devices using the defined set of functions
• Add devices. • Remove devices. • Support an interface for customization of smart integration rules. • Configure the CNS for various feature sets: – Read floor plan – Network configuration	• Plug and play mechanism for devices • There is a concept of rooms, floors, doors, windows, etc. that can be configured • There is a description of the topology of networked computers of the CNS that is configurable.
• Check configuration including features supported, devices attached, warranty dates for the CNS and each device, etc. • Monitor house state including who is in what room, windows and doors that are open, current mode (and mode definition), permissions, etc.	• Display of the current configuration with supported features and available device information like warranty, make, model etc • Dynamic display of the current status of doors, windows, lights • Display the permissions of the users of the system

Table 5: Transformation of Requirements to Responsibilities (cont.)

Requirements	**Responsibilities**
• Activate the CNS diagnosis. • Activate diagnosis for selected devices.	• The system will do diagnosis permanently in off peak times and for parts that are not currently in use • Specific diagnosis can manually be activated at any time
• Define modes such as night mode, security mode, diagnosis mode. • Activate modes manually or automatically.	• There is a concept of modes and functions and user permissions depend on it. Modes can be activated at certain times (day/night), at events (a break in), manually (vacation)
• Turn on/off. • Receive status • Reset • Start self test	• Connected devices have at least the following features: – Turn off/on. – Reset. – Start self test. – Receive status. • Devices that do not support some or all of those features (third party) will have a simulation for those features in the software

Next we group the responsibilities as shown in Table 6 according to logical coherence. Many responsibilities mention devices, which we put together into a *device* group. Other responsibilities talk about user interface, configuration, and diagnosis.

Table 6: Grouping of Responsibilities

Responsibilities	**Group**
• There exists a communication API that makes available the same functionality to all connected devices (local or remote) as to a local user interface • Some user interfaces must support a display with status information that gets updated in a short period of time (one second.) • A user can have multiple user interface instances (of different types, e.g., voice, keyboard) at one time. • Synchronization between the user interface instances of different/same current users needs to be done.	User interface
• There is a concept of rooms, floors, doors, windows, etc., that can be configured. • There is a description of the topology of networked computes of the CNS that is configurable. • Display the current configuration with supported features and available device information like warranty, make, model, etc. • There is a concept of modes and functions, and user permissions depend on it. Modes can be activated at certain times (day/night), at events (a break in), and manually (vacation) • Rules that describe interactions between devices using the defined set of functions can be defined any time. • Display the permissions of the users of the system.	Configuration

Table 6: Grouping of Responsibilities (cont.)

Responsibilities	**Group**
• The system will do diagnosis permanently in off-peak times and for parts that are not currently in use. • Specific diagnosis can be activated manually at any time.	Diagnosis
• Devices come with a set of functions that are defined in a way that a rule engine can interpret them. • Plug-and-play mechanism for devices exists. • Dynamically display the current status of doors, windows, lights, and other connected devices. • Connected devices have at least the following features: – Turn off/on. – Reset. – Start self test. – Receive status. • Devices that do not support some or all of those features (third party) will have a simulation for those features in the software.	Devices

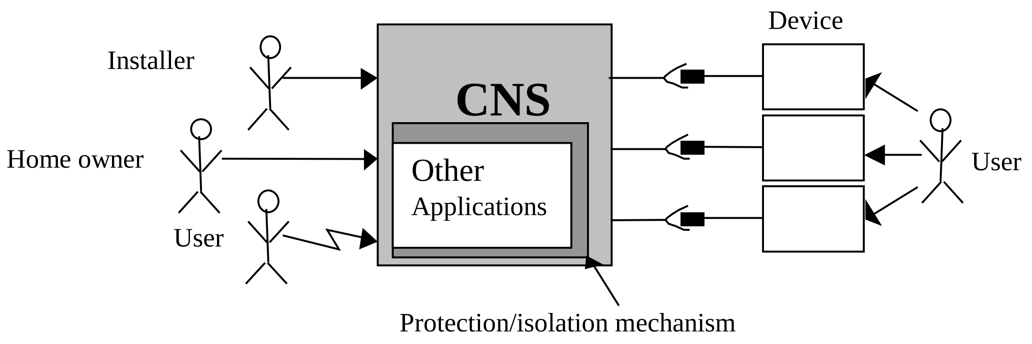
8 Choose Architecture Style (Handout 7) ABD Example—Home Integration System

8.1 Goals

- Determine the dominant architectural style that reflects the important qualities that have to be achieved within the design element. The choice of the dominant architectural style is based on the architectural drivers for this design element.
- Modify the dominant architectural styles to achieve particular goals.
- Document decisions on tradeoffs between the different quality attributes within the chosen style.

8.2 Requirements

Again we have repeated Figure 1 (the basic architecture structure) for context. Recall that the design element for this step is the CNS, which is the part of the home integration system that is wired into the house. The CNS provides the computer power and the connectivity needed for the various HIS applications. The CNS also provides a software platform for HIS applications from BBK and other vendors.



Applications that are essential to operate the HIS, such as the configuration, diagnosis, basic device operation for the most common devices, are part of the CNS. This also includes a user interface for the installer and a basic user interface for the homeowner.

Other applications and/or user interfaces can be installed as HIS applications on the CNS platform, but they are not considered to be part of CNS. Control software for additional devices can be installed as part of the CNS.

8.3 Result

Table 7 shows an analysis of the quality requirements from Table 3 as they apply to the current design element, which is the CNS. The third column gives the priority of each requirement. The symbol ++ indicates that this requirement is highest priority, while the symbol + indicates that this requirement is important but not highest priority. Some of the requirements are not applicable at this level of detail, and these requirements are indicated by the symbol N/A.

Table 7: Prioritizing the Quality Requirements

Quality	**Elaboration of the Quality**	**Priority**
Modifiability	• Device-related modifications – Add/remove known device types. – Add/remove new device types. – Home owner vs. installer modifications – Dynamic vs. static modifications • Portability-related modifications – Move to new OS. – Move to new middleware. – Move to new CPU. • CNS-related modifications – Change from a wire-based to wireless network. – Use power lines for LAN. – Customize the “smart” interactions.	++ This is the major business driver.
Security	• Physical intrusion prevention, detection, and reporting – Breaking through the wall and tampering with the wiring – Attempt to gain access by using electronic devices • Electronic intrusion prevention, detection, and reporting – Authentication – Protecting personal data – (Re)configuring security settings (e.g., when owners of house changes)	N/A Will become relevant on a more detailed level
Safety	• Detection and automatic reporting of safety violations – Fire, carbon monoxide, gasoline fumes, radon, etc. • Authentication for safety related appliances	N/A Relevant on a more detailed level
Reliability	• For some applications (such as stove, oven, utilities, fire and security), the CNS must be very reliable. • Fail safe	++
Performance	• Infrastructure performance	+
Interoperability	• BBK devices working with non-BBK devices	+

Table 7: Prioritizing the Quality Requirements (cont.)

Quality	Elaboration of the Quality	Priority
Usability	• For installer • For home owner • Various I/O modes (speech, text)	N/A

Table 7 shows that modifiability and reliability are the architecture drivers for this element.

The architecture options (shown in Table 4) chosen to support modifiability are layering and virtual interfaces for the devices. For reliability, the analytic redundancy (hardware/software for CNS applications) has been chosen.

In addition, the requirement of having separate user interfaces and the ability to plug in devices (BKK and non-BKK) suggests the use of a three-tier structure as the main architecture style as shown in Figure 2. This would also support the modifiability requirement.

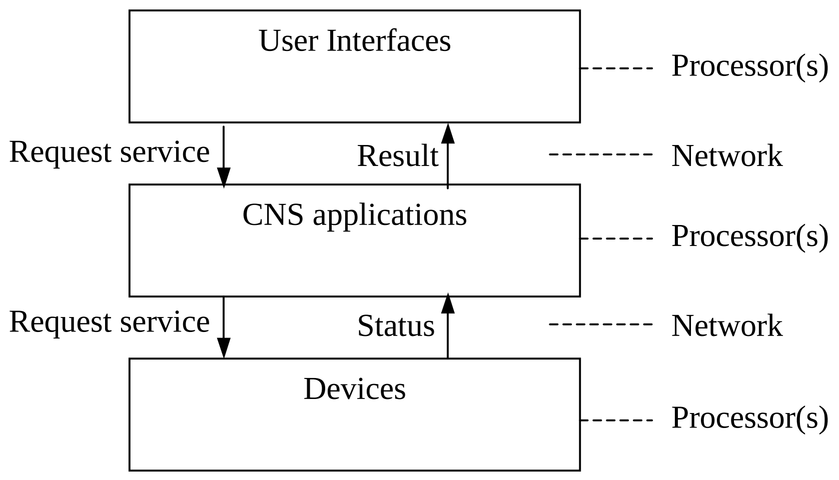


Figure 2: Three-Tier Structure

The three-tier style allows having multiple user interfaces to communicate with multiple CNS applications, which can be distributed onto multiple servers. This supports the modifiability and the reliability requirement.

On the device level, a separate processor with local functionality can ensure a safe mode in case the CNS application processor isn't working anymore.

Tradeoffs

- Performance is an issue because it is very likely that this three-tier structure will be distributed across the CNS network. Thus, it is likely that there is network communication between the layers. The performance of this network has a direct and strong influence on the performance of the overall system. It is assumed at this time that we can install a “fast enough” network. Also, to achieve better performance for some important functions, parts of the CNS applications can be moved to the user interface or the device processors.

- A low-cost version may be built with a single processor. This has a negative influence on the reliability. In this case, the three parts (user interface, CNS applications, and devices) need to run at least in separate processes to support some reliability.

9 Allocate Functionality to Style (Handout 8)

ABD Example—Home Integration System

9.1 Goals

- Allocate the groups of functionality resulting from the decomposition of function to the types of components resulting from the determination of style.
- The functionality is allocated in the form of a set of requirements (functional and non-functional) for the new design element.
- Describe the conceptual interface of the new design elements in terms of new information produced/needed and the data and control flow information needed by each component within the defined architectural style.

9.2 Background

Recall that we have defined the following groups of responsibilities (as shown previously in Table 6)*:*

- user interface
- configuration
- diagnosis
- devices

9.3 Result

We now allocate these groups to the component types defined by the architectural style chosen in Handout 7. This allocation is displayed as Figure 3.

The assignment of the user interface (UI) and device functionality to the style is trivial. The functionality groups for configuration and diagnosis are examples of CNS applications.

The three-tier style defines interactions between elements on a "request" basis (the higher levels request information from lower levels). Because of the multiplicity of the relations between the design elements of different layers and the requirement to have a permanently updated status display, a subscription/notification mechanism is used to make information available to higher levels.

We also introduce a *framework* component, to build a basis for further CNS applications. Parts of the framework deal with the protection against rogue applications.

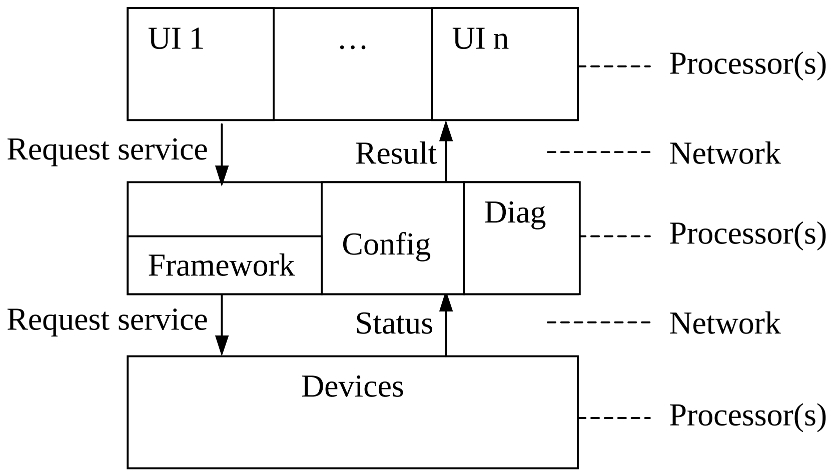


Figure 3: Allocation of Encapsulated Function to Style

Table 8 shows how the **requirements** for the decomposed design elements map to the allocation of groups of function to style.

Table 8: Requirement for the Decomposed Design Elements (Allocate Functionality to Style)

Design Element		**Requirements**
User Interface	Functional requirements	• Some user interfaces must support a display with status information that gets updated in a short period of time (1 second). • A notification mechanism for updated status information is needed. • Multiple user interface instances (of different types, e.g., voice, keyboard) should be available at one time. • Synchronization between the user interface instances of different/same current users is needed. • The user interface needs to react on changing configuration (add/remove devices).

Table 8: Requirements for the Decomposed Design Elements (Allocate Functionality to Style) (cont.)

Design Element		**Requirements**
	Non-functional requirements	Usability • For installer • For home owner • Various I/O modes (speech, text) Modifiability • Portability-related modifications – Move to new OS. – Move to new middleware. – Move to new CPU. • CNS-related modifications – Change from a wire-based to wireless network. – Use power lines for LAN. • Customize the "smart" interactions.
Diagnosis	Functional requirements	• Receive hardware/software error messages and keep a log. • Initiate self-test of connected devices during off peak times. • Collect and combine error data to diagnose faulty components. • Query software components for the current state. • Specific diagnosis can manually be activated at any time. • A subscription/notification mechanism for updated status information is needed.
	Non-functional requirements	Modifiability • Device-related modifications – Add/remove known device types. – Add/remove new device types. • Portability-related modifications – Move to new OS. – Move to new middleware. – Move to new CPU. • CNS-related modifications – Change from a wire-based to wireless network. – Use power lines for LAN. – Customize the "smart" interactions.

Table 8: Requirements for the Decomposed Design Elements (Allocate Functionality to Style) (cont.)

Design Element		**Requirement**
Configuration	Functional requirements	• Configures and provides information about rooms, floors, doors, windows, etc. • Configures and provides information about the topology of net-worked computers of the CNS • Display of the current configuration with supported features and available device information like warranty, make, model, etc. • Configures and provides information about permissions of the users of the system • A subscription/notification mechanism for updated status informa-tion is needed.
	Non-functional requirements	Modifiability • Device-related modifications – Add/remove known device types – Add/remove new device types – Home owner vs. installer modifications – Dynamic vs. static modifications • Portability-related modifications – Move to new OS – Move to new middleware – Move to new CPU • CNS related modifications – Change from a wire-based to wireless network – Use power lines for LAN – Customize the "smart" interactions Security • Electronic intrusion prevention, detection and reporting • (Re)configuring security settings (e.g., when owners of house change)
Application Framework	Functional requirements	• There is a concept of modes, functions, and user permissions de-pend on it. Modes can be activated at certain times (day/night), at events (a break in), manually (vacation) • A rule mechanism for defining and executing rules must be pro-vided.

Table 8: Requirements for the Decomposed Design Elements (Allocate Functionality to Style) (cont.)

Design Element		**Requirement**
	Non-functional requirements	Security • Physical intrusion prevention, detection, and reporting – Breaking through the wall and tempering with the wiring – Attempting to gain access by using electronic devices • Electronic intrusion prevention, detection, and reporting – Authentication – Protecting personal data – (Re)configuring security settings (e.g., when owners of house change) Safety • Authentication for safety-related appliances • Provide a “safe” environment for CNS applications, especially third-party applications.
Devices	Functional requirements	• Provides a device interface that can be used by a rule engine. • Provides a plug-and-play mechanism for devices • Provides the current status of doors, windows, lights, and other connected devices • Provides for connected devices, including at least the following features: – Turn off/on. – Reset. – Start self test. – Receive status. • Devices that do not support some or all of those features (third party) will have a simulation for those features in the software. • A subscription/notification mechanism for updated status information is needed.

Table 8: Requirements for the Decomposed Design Elements (Allocate Functionality to Style) (cont.)

Design Element		**Requirement**
	Non-functional requirements	Modifiability • Device-related modifications – Add/remove known device types. – Add/remove new device types. • Portability-related modifications – Move to new OS. – Move to new middleware. – Move to new CPU. • CNS-related modifications – Change from a wire-based to wireless network. – Use power lines for LAN. Safety • Detection and automatic reporting of safety violations – Fire, carbon monoxide, gasoline fumes, radon, … Reliability • For some applications (such as stove, oven, utilities, fire and security), the CNS must be very reliable. • Fail safe Interoperability • BBK devices working with non-BBK devices

Table 9 shows the **information interface** for the decomposed design elements.

Table 9: Information Interface for Decomposed Design Elements

Design Element	**Information Produced**	**Information Consumed**
User Interface	• User commands	• Floor plans • Status information – Diagnosis data – Device status • User information
Diagnosis	• Diagnosis data	• Device status • User commands
Configuration	• Floor plans • User information	• User commands • Device maintenance
Application Framework		
Devices	• Device status • Device maintenance	• Device commands

10 Refine Templates (Handout 9) ABD Example—Home Integration System

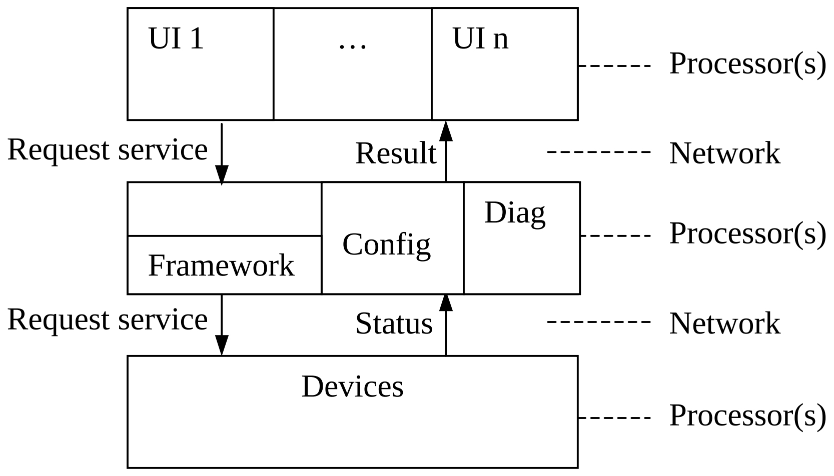
10.1 Goals

Commonalities that apply to a set of design elements will be captured in one or more templates. They are inherited from their parents up the hierarchy.

- A template contains commonalities such as
  - common services
  - access pattern to common services and/or infrastructure
  - policies and/or patterns of how to implement a design element (e.g., how to implement error handling)
- A template may also contain variables (e.g., type of OS used or implemented as a parameter) that can be used to generate specialized sets of templates.

10.2 Design Elements

We have repeated Figure 3 (allocation of encapsulated function to style) here for context. Templates are associated with the allocation we have made, which is displayed in Figure 3. The parent design element (the system, in this case) doesn't have a template associated with it. The process of determining templates is to provide an example of the responsibilities associated with the various design elements and to determine commonalities. These responsibilities are given in Table 8.



10.3 Result

By examining the list of requirements of the design elements, the following commonalities can be found:

- Publish/subscribe mechanism:
  Elements need the ability to subscribe to services provided by other elements and have to provide a subscription mechanism for other elements.
- Configurable elements:
  Elements can depend on a specific configuration. The way to adjust to changes in the configuration should be handled in the same way.
- Diagnosable elements:
  Error conditions detected by an element must be reported to a diagnosis service. This should be done in the same way as for configurable elements. Also, elements need to re-act to status requests from the diagnosis service.
- Communication mechanisms:
  Communication between elements involves inter-process and network communication. These types of communication must be handled the same way.
- OS services:
  Access to the main operation system services, such as thread and event handling, should be localized to support portability.
- Basic device interface:
  All devices must have a basic set of features using a common interface.

Table 10 shows the **templates** for the decomposed design elements.

Table 10: Templates for Decomposed Design Elements

Design Element	**Templates that Pertain to It**
User Interface	• Publish/subscribe mechanism • Configurable elements • Diagnosable elements • Communication mechanisms • OS services
Diagnosis	• Publish/Subscribe mechanism • Configurable elements • Communication mechanisms • OS services
Configuration	• Publish/subscribe mechanism • Diagnosable elements • Communication mechanisms • OS services

Table 10: Templates for Decomposed Design Elements (cont.)

Design Element	**Templates that Pertain to It**
Application Frame-work	• Publish/Subscribe mechanism • Configurable elements • Diagnosable elements • Communication mechanisms • OS services
Devices	• Publish/Subscribe mechanism • Configurable elements • Diagnosable elements • Communication mechanisms • OS services • Basic device interface

11 Verify Functionality (Handout 10) ABD Example—Home Integration System

11.1 Goals

- Exercise use cases to verify that they can be achieved through the proposed structure.
- Determine and document additional responsibilities for the child design elements.
- Exercise change scenarios to determine the difficulty of performing a change.

11.2 Use Cases

- Homeowner calls from within her car via cellular phone to activate the air conditioning of the house.
- A new, so far unknown, BBK device is connected to CNS. The BBK device brings the necessary CNS software with it (as part of the device, not on media).
- Display the floor plan with the current status of the connected devices, such as doors, windows, temperature, oven, etc.

11.3 Change Scenarios

- Change communication mechanism from the Component Object Model (COM) to the Common Object Request Broker Architecture (CORBA).
- A new non-BBK device that does not adhere to the CNS plug-and-play interface needs to be supported.

11.4 Result

11.4.1 Call to Turn on Air Conditioning

Homeowner calls from within her car via cellular phone to activate the air conditioning of the house. The assumption is that the CNS system has a modem connected to it that has its own phone number.

The sequence of events that occurs and the design element affected by the events are as follows:

–	Outside	The call arrives at the modem.
–	Devices	recognize an incoming call and notify the application framework.
–	Framework	requests authentication from the user via the modem device.
–	Outside	enters authentication, which is routed to the application framework.
–	Framework	authenticates and sends “to log” message to the diagnosis service.
–	Diagnosis	writes message into a persistent storage.

–	Framework	assigns a data channel to a UI for remote users using configuration.
–	UI remote	requests a command from the user via the data channel.
–	Outside	User enters the command for turning on air-conditioning.
–	UI remote	requests the air-conditioning service, which is a CNS application.
–	Application	turns on the air-conditioning device.
–	Application	sends status information to user via user interface.

Figure 4 shows this trace using use-case map notation [Buhr 96]. Readers unfamiliar with this notation should see <http://www.usecasemaps.org/>.

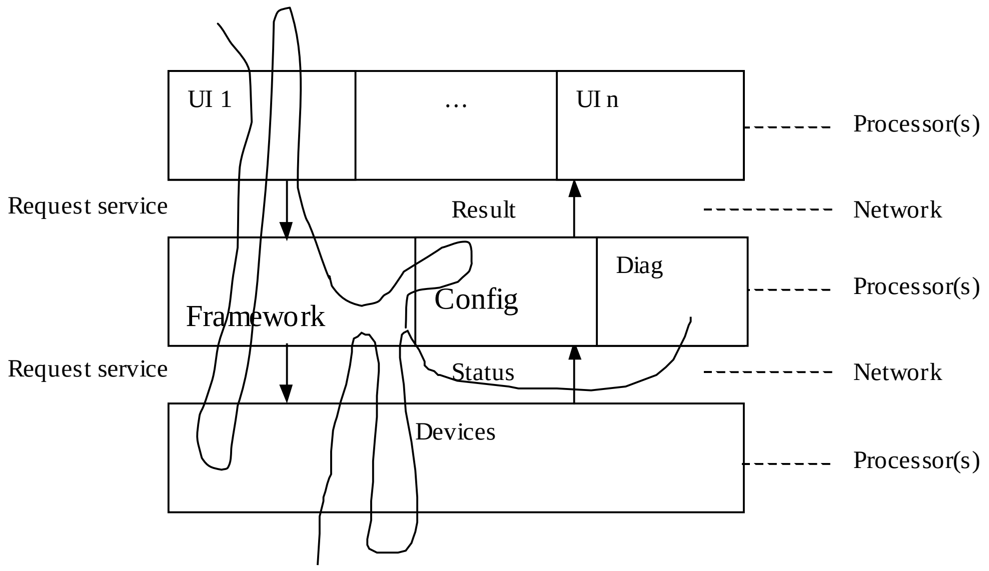


Figure 4: Use-Case Map for Call Arriving to Turn on Air Conditioning

11.4.2 Add New Device to CNS

A new, so far unknown, BBK device is connected to the CNS. The BBK device brings the necessary CNS software with it (as part of the device, not on media).

The sequence of events and the design element responding to these events are as follows:

–	Outside	The BBK device gets connected.
–	Devices	recognize the new devices and request identification information.
–	Outside	Device sends the identification information.
–	Devices	recognize that it is an unknown BBK device (drivers not installed).
–	Devices	request CNS drivers from the new device.
–	Outside	uploads the required software.
–	Devices	install the driver software and inform the configuration
–	Configuration	changes the CNS configuration to reflect the new device.
–	Configuration	updates possible displays that are effected by the change.
–	Configuration	informs the diagnosis to test the configuration of the new device.
–	Diagnosis	requests a self-test of the device and requests status from the software.
–	Diagnosis	updates possible displays that show the status of the new device.
–	Device	puts the new device into a defined default state.

Figure 5 shows the use-case map for this use case.

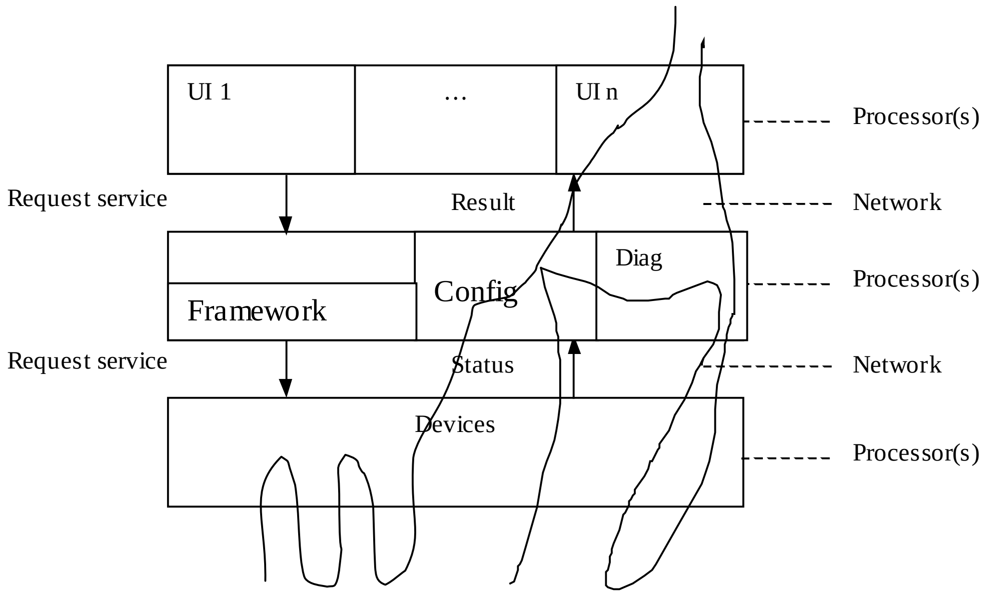


Figure 5: Use-Case Map for Adding New Devices to the CNS

11.4.3 Change Communication Mechanism from COM to CORBA

This is a change scenario. The communication mechanism is changed from COM to CORBA.

- This requires the template with the communication mechanism to be changed.
- There might be some adaptations of the elements that use the communication template.

11.4.4 Add New Device

This is another change scenario. A new non-BBK device that does not adhere to the CNS plug-and-play interface must be supported.

- A new driver for the device must be added to the device's element.
- The configuration must be adapted to the new device.

An application that controls the device must be installed if it isn't possible to control the device with standard applications.

12 Generate Concurrency View (Handout 11)
ABD Example—Home Integration System

12.1 Goals

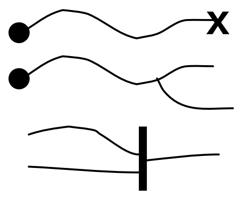
- Design the concurrency aspects of the architecture in a different view.
- Design virtual threads as parallel activities and their synchronization.

12.2 Use Cases

- Start up the CNS system.
- Two users simultaneously access a device.

12.3 Notation

In this example, we use the following simplified notation:



Virtual thread (activity) with start and end point

Another thread starts

Synchronization of threads

12.4 Result

12.4.1 Start-up

The CNS system is started. The system goes through the following sequence of events demonstrated by the use-case map shown in Figure 6.

–	Outside	CNS system is powered on.
–	Framework	OS starts the initialization of the framework.
–	Outside	Framework spawns a thread per user interface for initialization.
–	User Interface	UI initializes and spawns a thread that waits on a synchronization point for user input. The original thread returns to a synchronization point in the framework that ensures that all UI threads return properly.
–	Framework	Framework spawns a thread to initialize the devices.

–	Devices	do the initialization and may spawn another thread to wait for input from devices.
–	Framework	Framework spawns a thread per CNS application.
–	Applications	The applications initialize and some applications, which must be active as in the case for the diagnosis, spawn a local thread to perform repeating activities. The original thread returns to a synchronization point in the framework that ensures that all UI threads return properly.
–	Framework	waits at a synchronization point that all spawned threads return. After that is done the initialization thread terminates.

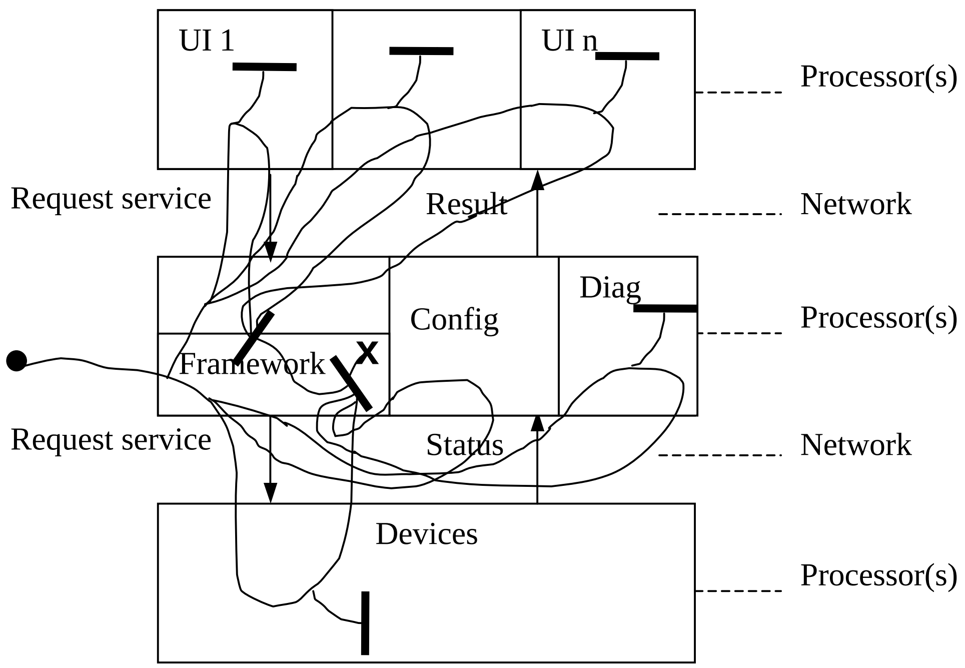


Figure 6: Use-Case Map for Start-Up

12.4.2 Two Simultaneous Users

In this use case, two users simultaneously access a device. This leads to the following sequence of events shown in Figure 7.

–	User Interface	UI1 receives an input from a user, which is a control command to a device. The thread that waited for the input spawns a thread that is responsible for executing the user command. Then the user input thread returns to the wait synchronization point to wait for the next user input.
–	Application	The command execution thread executes the application that is responsible for controlling the appropriate device. After some plausibility checks, the application attempts to access the device via the framework.

– Framework	The framework routes the request to the appropriate device.
– Device	Before accessing the device, a serialization of concurrent request needs to be done. Therefore the device uses a synchronization point (resource management). Since this is the first thread it gains access to the device.
– User Interface	UIn receives an input from another user, which is a control command to the same device. The thread that waited for the input spawns a thread that is responsible for executing the user command. Then the user input thread returns to the wait synchronization point to wait for the next user input.
– Application	The application also executes the request.
– Framework	The framework routes the request to the same device.
– Device	The thread arrives at the synchronization point and, since the device is active, this thread has to wait. The first thread returns to the user interface and frees the device. This is the moment where the second thread gains access to the device, does the control function, and re-turns to the user interface.
– User Interface	After giving some feedback to the user, the command execution threads terminate.

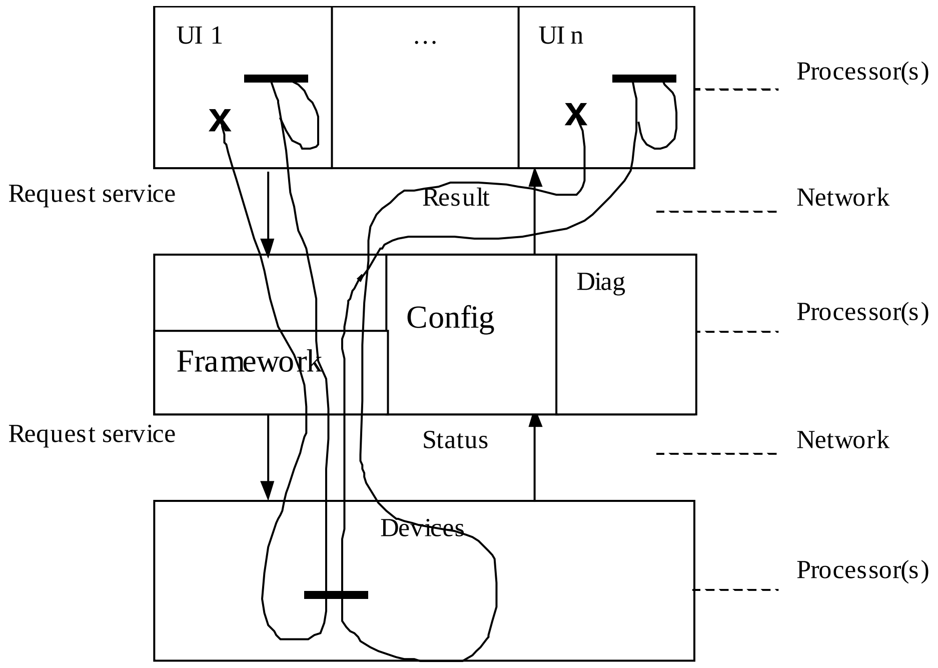


Figure 7: Use-Case Map for Two Simultaneous Users

13 Generate Deployment View (Handout 12)

ABD Example—Home Integration System

13.1 Goals

- Design units of deployment and deploy them on a processor configuration. A unit of deployment is a collection of one or more design elements that can be allocated to a single processor. A unit of deployment is atomic in the deployment process. It cannot be split and deployed on more than one processor.
- Understand the effect of the deployment on the virtual threads shown in the concurrency view.

13.2 Processor Configuration

In this example we assume a two-processor configuration as shown in Figure 8. One processor provides the connectivity for the devices and has a special backup mechanism (power) for safety-critical functions. The other processor is a general-purpose computer. The computers are connected via Ethernet.

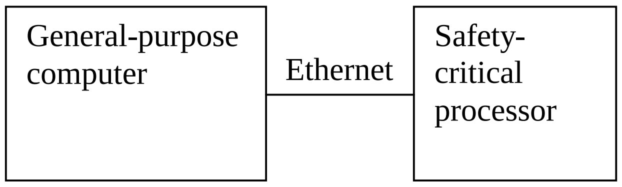


Figure 8: Two-Processor Configuration

13.3 Deployment

Table 11 gives the allocation of design elements to the units of deployment. Since we have only two computers in our configuration, there are no multiple instances of any of the design elements with the exception of the user interface, which comes in two variants for local and remote access.

Table 11: Definition of Units of Deployment

Design Element	**Units of Deployment**	**Comment**
User Interface	User Interface	Every instance of the user interface (e.g., key-board/monitor, voice, remote) becomes a unit of de-ployment.
Diagnosis	Diagnosis	
Configuration	Configuration	
Application Frame-work	Application	So far, we have packed possible applications and the application framework together as one unit of deploy-ment. A further decomposition may change this.
Devices	Devices	So far, we have packed all the software for devices to-gether in one unit of deployment. In the future, the soft-ware for some or all devices might be separated.

Figure 9 shows this allocation pictorially.

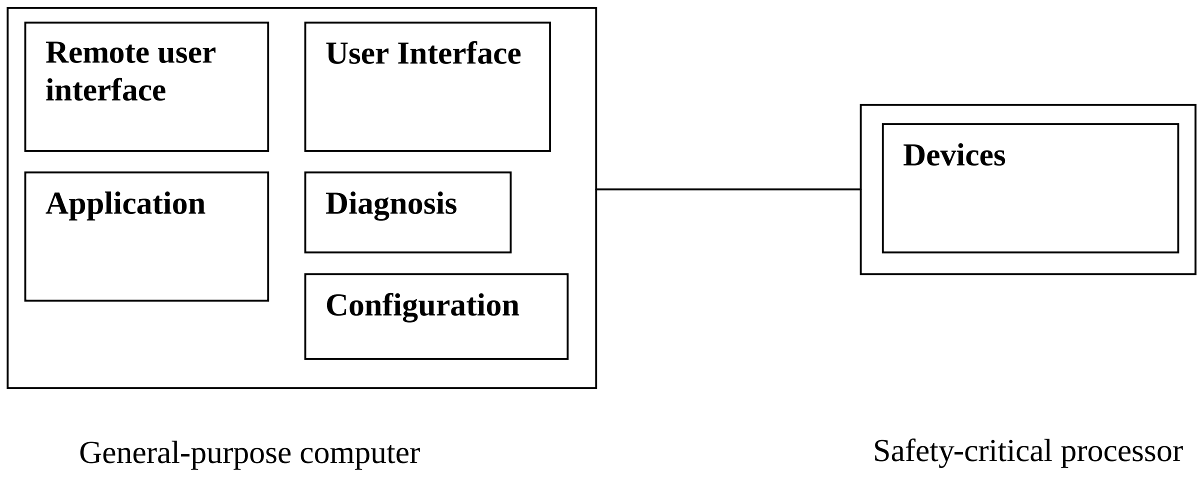


Figure 9: Deployment on the Given Hardware Configuration

13.4 Two Users Simultaneously Access a Device

Figure 10 shows the use-case map for two users simultaneously accessing a device, as it was shown in Figure 7, with the messages passing through the two physical processors that we are assuming. The following is the sequence of events:

In this use case, two users simultaneously access a device. This leads to the following sequence of events shown in Figure 7.

–	User Interface	UI_1 receives an input from a user, which is a control command to a device. The thread that waited for the input spawns a thread that is responsible for executing the user command. Then the user input thread returns to the wait synchronization point to wait for the next user input.
–	Application	The command execution thread executes the application that is responsible for controlling the appropriate device. After some plausibility checks, the application attempts to access the device via the framework.
–	Framework	The framework routes the request to the appropriate device.
–	Device	Before accessing the device, a serialization of concurrent request needs to be done. Therefore the device uses a synchronization point (resource management). Since this is the first thread, it gains access to the device. The thread then crosses on the network to the second processor, accesses the device, and returns across the network to the framework.
–	User Interface	UI_2 receives an input from another user, which is a control command to the same device. The thread that waited for the input spawns a thread that is responsible for executing the user command. Then the user input thread returns to the wait synchronization point to wait for the next user input.
–	Application	The application also executes the request.
–	Framework	The framework routes the request to the same device. The thread travels over the network to the device.
–	Device	The thread arrives at the synchronization point and, since the device is active, this thread has to wait. The first thread returns to the user interface (over the network) and frees the device. This is the moment where the second thread gains access to the device, does the control function, and returns to the user interface (also over the network).
–	User Interface	After giving some feedback to the user, the command execution threads terminate.

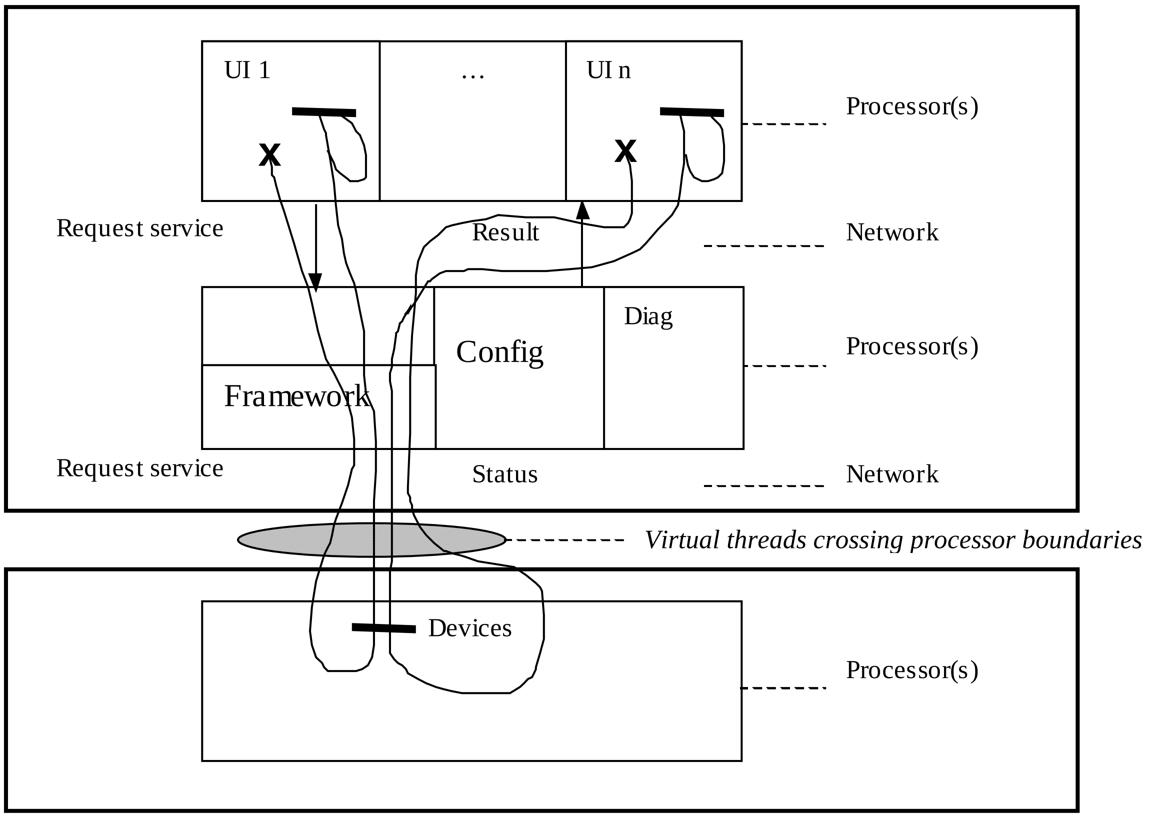


Figure 10: Use-Case Map for Two Simultaneous Users on Deployed System

As can be seen, the virtual threads have to cross the processor boundaries using the network that connects the two processors. This requires communication mechanisms (e.g., remote procedure/method call) that haven’t been mentioned so far.

14 Verify with Quality Scenarios (Handout 13)

ABD Example—Home Integration System

14.1 Goals

- Verify each quality scenario against the structure of the decomposed design elements to examine if it is still possible to satisfy the scenario.
- In the event of a conflict, either the decisions that have been made should be reconsidered or the architect must accept the failure to realize one of the quality scenarios.
- The rationale for accepting a failure to realize one of the quality scenarios should be recorded.

14.2 Quality Scenarios

- In case of power failure of the CNS system, the telephone and the security system must function with at least basic features. Making and receiving phone calls is still possible, an attempt to break into the house is still detected, and an alarm is being sent (Reliability).
- A change of the status of a device that is displayed needs to be updated within one second (Performance).
- A five-year old child must be able to turn on/off the TV that is controlled by the CNS (Usability).

14.3 Result

- In case of power failure of the CNS system, the telephone and the security system must function with at least basic features. Making and receiving phone calls is still possible, an attempt to break into the house is still detected, and an alarm is being sent (Reliability).

  A normal phone is connected in parallel to the CNS system. Therefore it can be used independently. This may have implications for normal use of the phone. The CNS system must be aware of the fact that the phone line may not be available when needed. (This is a responsibility of the phone part of the device structure.)
- A change of the status of a device that is displayed needs to be updated within one second (Performance).

  The performance in this example mostly depends on the bandwidth and the amount of data traffic over the network connection between the two processors. Close attention must be paid to the network connection. Since the change of status must be shown within one second, the refresh rate of the display must be lower than this. So far the structure *doesn't imply any constraints on this. The structure itself doesn't seem to be a performance problem so far.*

- A five-year old child must be able to turn on/off the TV that is controlled by CNS (Usability).

  So far there isn't a more detailed specification of the user interface. This requirement can be fulfilled so far.

References

[Bachmann 00] Bachmann, Felix; Bass, Len; Chastek, Gary; Donohoe, Patrick; and Peruzzi, Fabio. *The Architecture Based Design Method* (CMU/SEI-2000-TR-001, ADA 375851). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, March 2000. Available WWW: <URL: http://www.sei.cmu.edu/publications/documents/00.reports/00tr001.html>.

[Buhr 96] Buhr, R. J. A. and Casselman, R. S. *Use Case Maps for Object-Oriented Systems.* Upper Saddle River, NJ: Prentice Hall, 1996.

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (LEAVE BLANK)	**2.** REPORT DATE **September 2000**	**3.** REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE **An Application of the Architecture-Based Design Method to the Electronic House**		**5.** FUNDING NUMBERS C — F19628-95-C-0003
6. AUTHOR(S) **Felix Bachmann, Len Bass, Mark Klein**		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		**8.** PERFORMING ORGANIZATION REPORT NUMBER **CMU/SEI-2000-SR-009**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		**10.** SPONSORING/MONITORING AGENCY REPORT NUMBER **N/A**
11. SUPPLEMENTARY NOTES		
12.A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		**12.B** DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

The Architecture-Based Design (ABD) Method is a method for designing the software architecture of a product line of systems. It has previously been described in the technical report, *The Architecture Based Design Method* (CMU/SEI-2000-TR-001) [Bachmann 00]. This report elaborates an example of the application of this method to designing the software architecture. The example is the house of the future.

The house of the future is assumed to have a collection of devices within the house that are controlled by a computer network. Entertainment, security, heating/air-conditioning, and utility devices will all interoperate and will be controlled from a central network. The software architecture to support the house must be extendible and flexible, and it must have high security, high performance, and high availability. In this report, we present a first-level decomposition of the software architecture as a demonstration of the ABD Method.

SUBJECT TERMS **Architecture-Based Design (ABD) Method, architecture design, software architecture**	**15.** NUMBER OF PAGES **50**
16. PRICE CODE	

7. SECURITY CLASSIFICATION OF REPORT **UNCLASSIFIED**	**18.** SECURITY CLASSIFICATION OF THIS PAGE **UNCLASSIFIED**	**19.** SECURITY CLASSIFICATION OF ABSTRACT **UNCLASSIFIED**	**20.** LIMITATION OF ABSTRACT **UL**