NTNU
Norwegian University of Science and Technology

Faculty of Informatics, Mathematics and Electronics

Department of Computer and Information Sciences

ENGLISH

Examination results will be announced: 18. June

Exam in the subject
TDT4240 Software Architecture

Friday 28. May 2010
9:00 am – 1:00 pm

Aids code C:

Simple calculator allowed.
These specified printed documents are allowed:

- IEEE (2000), "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems", Software Engineering Standards Committee of the IEEE Computer Society.
- Kruchten, P. (1995), "The 4+1 View Model of Architecture", IEEE Software, 12(6).
- English-Norwegian dictionary (or to your native language if your not Norwegian) and/or an English thesaurus (English-English).

Contact person during the exam:

Alf Inge Wang, phone 7359 4485, mobile phone: 9228 9577

The points show how much each problem is worth in this exam. For each problem, each question has the same weight unless otherwise stated. The exam has 5 problems giving a total of 70 points. The remaining 30 points are credits awarded from the software architecture project.

Good Luck!

Salah Uddin Ahmed, Meng Zhu, and Bian Wu

Controlled 20th of May 2010

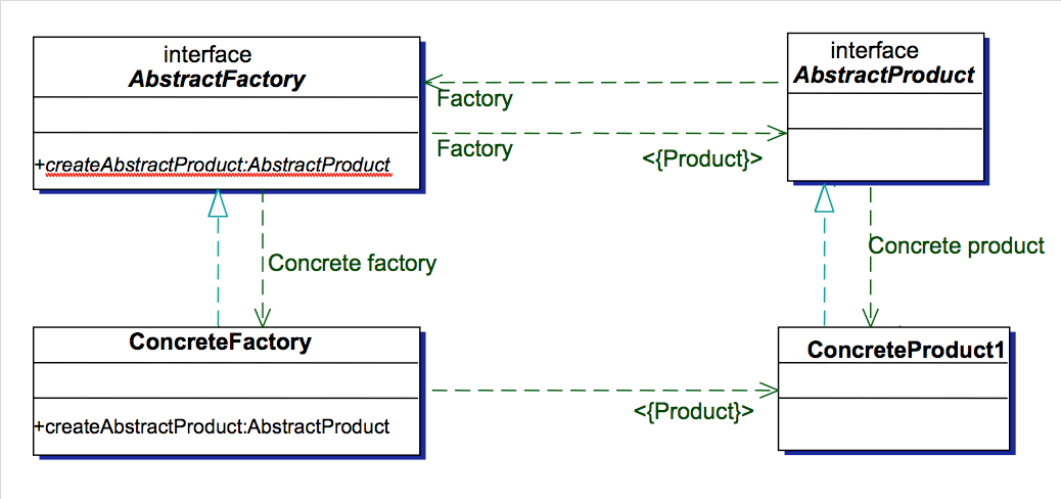
Problem 1: Various questions (20 points)

Answer these questions shortly:

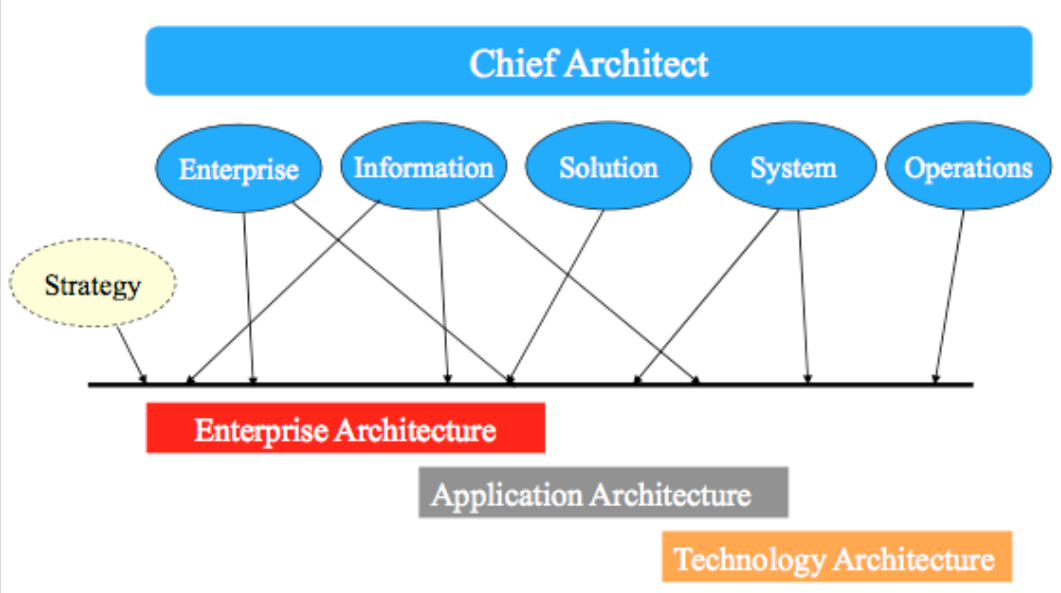
1.1	What is Bass, Clements and Kazman's definition of Software Architecture (the definition in the textbook)?
1.2	What is the quality attribute usability concerned with?
1.3	Give examples of typical response measures for quality attribute performance.
1.4	How can architectural patterns and design patterns be combined?
1.5	Describe the *Abstract Factory* design pattern and advantages of using it.
1.6	Describe the three IT architectures presented by Sigurd Thunem (Telenor) and the areas these architectures address.
1.7	What is the most important architectural driver for commercial IT architectures according to Telenor?
1.8	What problems are addressed through the Service-Oriented Architecture (SOA) approach according to Fredrik Dahl-Jørgensen (Accenture)?
1.9	Describe the Service Identification Framework (SIF) in SOA (the main parts and their relationships).
1.10	How do different stakeholders of a software system influence the software architecture? Give examples for Developer, Marketing, Customer, End-user, and Maintainer?
1.11	Sketch (draw) the relationship between an architectural pattern, a reference model and a reference architecture.
1.12	Describe the three main areas of performance tactics identified by the textbook.
1.13	Why is it necessary to describe a software architecture through several views (more than one view)?
1.14	What is the purpose of evaluating software architecture?
1.15	Describe shortly the process of reconstructing a software architecture as described in the textbook.
1.16	What is the main difference between ATAM and CBAM?
1.17	What is important to focus on when designing a software architecture for a software product line?
1.18	What are the challenges when using Off-The-Shelves (OTS) components in a software architecture?
1.19	What is the difference between a wrapper and a bridge?
1.20	What is architectural drift?

Solution:

- 1.1: A software architecture of a program or computing system is the structure(s) of the system, which comprise of software elements, the externally visible properties of those elements, and the relationships among them.
- 1.2: How easy it is to use a system: learning system features, using a system efficiently, minimizing the impact of errors, adapting the system to user needs, increasing confidence and satisfaction.
- 1.3: Number of simultaneous users, transactions per second, latency, frame rate
- 1.4: Architectural patterns can usually be decomposed into several design patterns. An alternative is to apply design patters for large structures of the system, thus the design pattern will be used as an architectural pattern.
- 1.5: Abstract factory advantage is that the client should not know anything about which variant they are using.

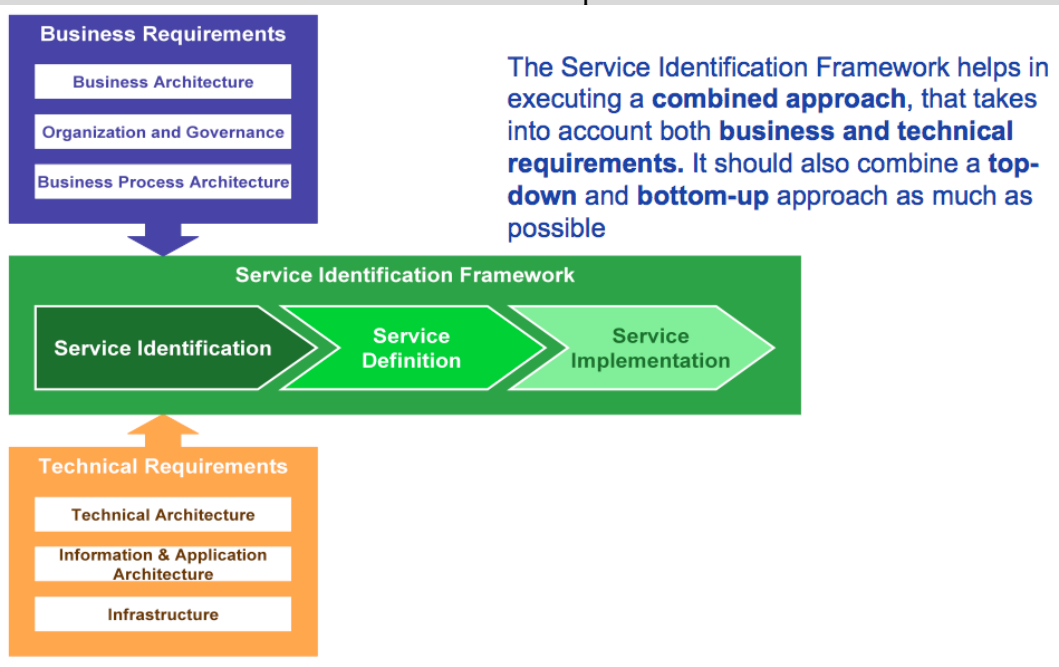


- 1.6: IT architecture consists of enterprise, application and technology architecture and addresses the areas strategy, enterprise, information, solution, system and operations.

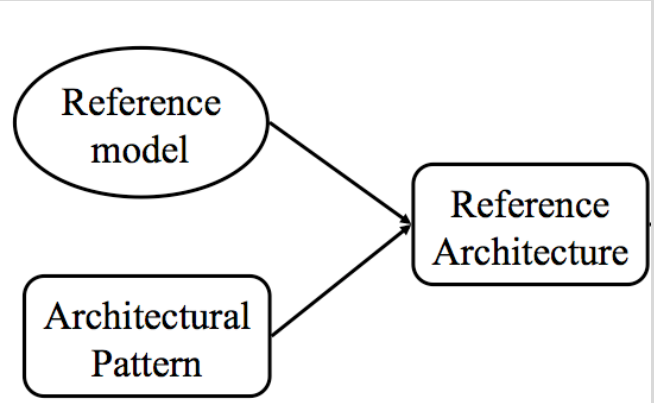


- 1.7: Most important IT architecture driver Telenor: The most cost effective solution to a problem, balancing short term and long-term issues.
- 1.8: SOA addresses tight coupling and strong dependencies between components in a system.

- 1.9: SIF consists of Service identification, Service definition and Service implementation that take into account business and technical requirements.



- 1.10: Different stakeholders have different needs related to the system addressed in the software architecture. Developer: easy to implement; Marketing: features, time-to-marked, low cost; Customer: Low cost, on time; End-user: easy to use, stable; and Maintainer: easy to modify.
- 1.11: Reference architecture is a combination of a reference model (decomposed functionality) and a architectural pattern (components structured to give a certain characteristics).



- 1.12: Performance tactic areas: Resource demand, Resource management, and Resource arbitration.
- 1.13: Why architectural views: Various stakeholders have different interests and needs in the system.
- 1.14: The purpose of evaluating a software architecture is to be sure that the architecture serves the purpose according to the quality you want to obtain (insurance).
- 1.15: Reconstruction process consists of 1) information extraction, 2) database construction, 3) view fusion, and 4) reconstruction of the architecture.
- 1.16: Main difference between ATAM and CBAM is that CBAM computes the return-on-investment of several possible tactics to choose the best one, while ATAM focus on a set of selected tactics. Also the CBAM scenarios describe several possible response levels.

- 1.17: When designing a software architecture for product line the main focus is on what is the stable core and what are the variation points of the software.
- 1.18: The challenge of using OTS in a software architecture is that you do not have control of the quality and the architectural assumption of the components.
- 1.19: The difference between a wrapper and a bridge is that the wrapper changes the interface of a component (both input and output), while a bridge translate between the interfaces of two components.
- 1.20: Architectural drift is when the architecture is changed to be up-to-date with the implementation that will cause of a lack of coherence and clarity.

Problem 2: Choose the most appropriate architectural pattern (5 points)

Nominees:

a) Model-view-controller
b) 3-tier
c) Peer-to-peer
d) Pipe-and-filter
e) Layered
f) Blackboard

Choose the *most appropriate architectural pattern* for these 5 short descriptions of systems. Motivate for your choices:

1. A game engine that provides a high-level API to the programmer. The programmer can also access medium-level and low-level APIs to get a richer set of functionality if required.
2. An application for cooking recipes that can run on various mobile applications with various screen configurations (in size) and input devices (keys, touch-based screens, joysticks, etc).
3. A PC application for analyzing weather data through a set of data transformations.
4. An application for exchanging electronic business cards between mobile devices such as mobile phones, personal data assistants (PDAs), smart phones etc.
5. A distributed collaborative application for sharing various information stored in a common database (repository). The various parts of the application that store and retrieve information from the database should be possible to be replaced dynamically (run-time).

Solution:
1. e) Layered
2. a) Model-view-controller
3. d) Pipe-and-filter
4. c) Peer-to-peer
5. f) Blackboard

Problem 3: ATAM (5 points)

Do the step 6 (Analyze the architectural approaches) in the ATAM process on software for controlling a garage opening system consisting of sensors, motors with activators, and garage controller to operate the garage door.

Utility tree:

- Availability:
  - Scenario A1. The system must be available 99.9% of the time (M,H).
- Performance:
  - Scenario P1. If obstacle is detected during lowering the garage door, it must be reopened within 0.1 second (H,M).
- Security:
  - Scenario S1: It should be less than 0.01% chance to get unauthorized access to the garage controller. (L,M).

Identified architectural tactics:

- AT1: Increase computational efficiency in critical components.
- AT2: Schedule time-critical components wisely.
- AT3: Structure the system to have semantic coherence.
- AT4: Use information hiding.

Solution:

Of the three scenarios A1, P1, S1, we only need to analyze the P1 scenario as there are no tactics for availability and security. Also we only have to consider the tactics AT1 and AT2, as we do not have any modifiability scenarios.

Scenario#: P1 Scenario: Obstacle detected during garage door decent
Attribute(s): Performance
Environment: Normal operations
Stimulus: Obstacle detected during lowering garage door
Response: Reopen within 0.1 sec

Architectural Decisions:	Sensitivity	Tradeoff	Risk	Nonrisk
AT1: Increase computational efficiency	S1		R1	
AT2: Schedule time-critical component		T1		N1

Reasoning: The tactics AT1 and AT2 should be sufficient to ensure P1 as long as the CPU is fast enough.
Architectural diagram: None

Sensitivity points, Tradeoff points, Risk and Nonrisk:
S1 AT1 is only positive sensitive to performance
T1 AT2 is positively sensitive to performance but could be negative sensitive to availability for other parts of the system than when obstacle is detected.
R1 AT1 could be a risk to the project, as it will demand more work on the programmers.
N1 AT2 should be a safe nonrisk decision.

Risk themes: None

Problem 4: CBAM (10 points)

From the Table 1, 2 and 3 and the information below, find:

- **Total benefit** obtained from the 3 architectural strategies.
- **Return-On-Investment** for the 3 architectural strategies
- **Rank the 3 architectural strategies** according to best investment.

Use straight lines between the data points in the graph.

The data is results from applying the Cost Benefit Analysis Method (CBAM) on a software system for managing car rentals.

Table 1: Results from prioritizing scenarios with worst, current, desired and best response levels.

Scenario	Vote	Worst	Current	Desired	Best
1. Performance: Highest number of simultaneous user requests	30	100 users	2000 users	10000 users	100000 users
2. Availability: How much of the time the server can crash	20	10% crash	1% crash	0.5% crash	0 % fail
3. Availability: Time to recover from a crash	20	10 min	3 min	1 min	0 min
4. Modifiability: Time to add support for a new type of vehicle	10	8hours	60min	10min	1min
5. Security: Probability for accessing credit card information	20	1%	0.1%	0.01%	0%

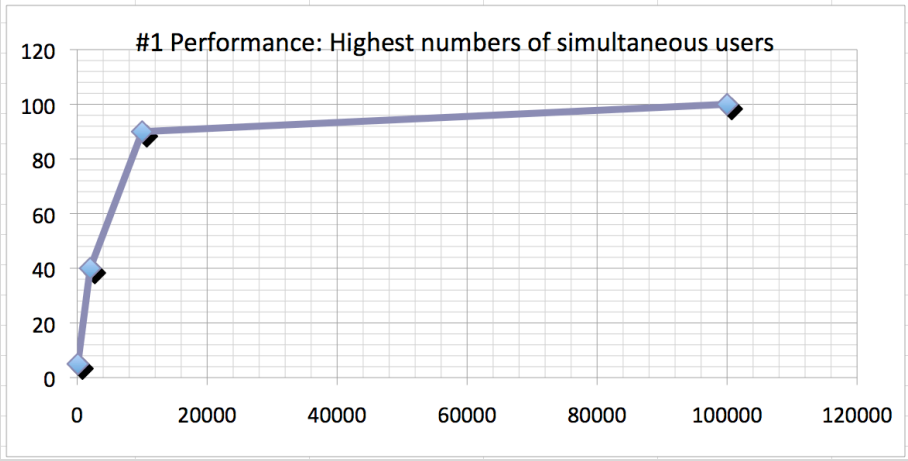
Table 2: Results from assigning utility to the various scenarios.

Scenario	Vote	Worst	Current	Desired	Best
1. Performance	30	5	40	90	100
2. Availability	20	5	50	70	100
3. Availability	20	5	30	70	100
4. Modifiability	10	5	20	50	100
5. Security	20	5	30	80	100

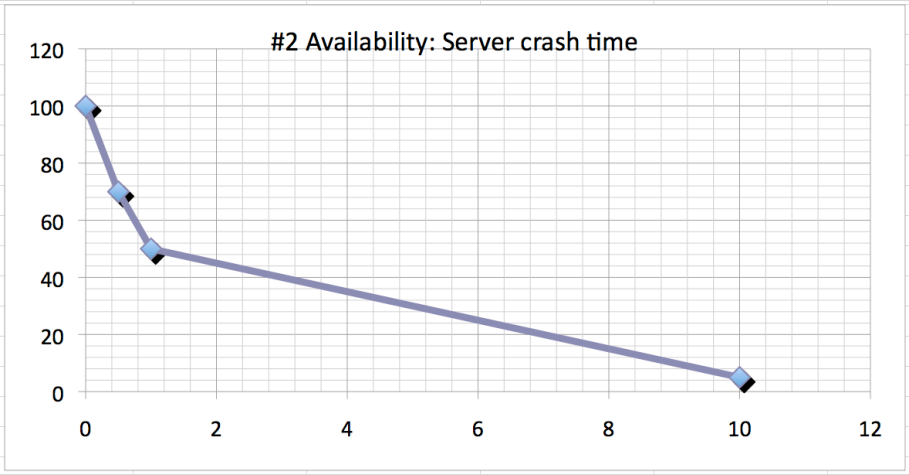
Table 3: Effect and cost of using architectural strategies.

Strategy	Scenario	Cost	Current response	Expected response
1. Replicated servers (hardware and software)	1 2 3	15000	2000 users 1% crash 3 min	8000 users 0.7% crash 30 sec
2. Improved computational efficiency	1	4000	2000 users	4000 users
3. Improved exception handling	2 3	3500	1% crash 3 min	0.4% crash 2 min

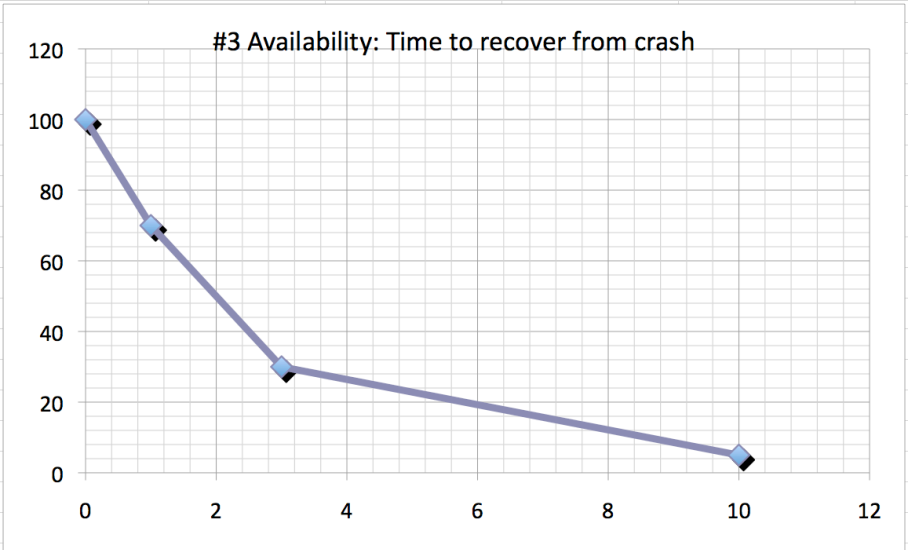
Solution:
We do only have to create utility-response curves for Scenario 1-3, as the architectural strategies affects these three scenarios.
Utility-response curve scenario #1:



Utility-response curve scenario #2:



Utility-response curve scenario #3:



Effects of strategy #1 replicated servers (read from the graphs):

- Scenario 1: From 2000 to 8000 users: 75% utility (from 40%)
- Scenario 2: From 1% to 0.7% crash gives: 60% utility (from 50%)
- Scenario 3: From 3 min to 30 sec recovery gives: 80% utility (from 30%)

Effects of strategy #2 improved computational efficiency:

- Scenario 1: From 2000 to 4000 users gives: 50% utility (from 40%)

Effects of strategy #3 improved exception handling:

- Scenario 2: From 1% to 0,4 % crash gives: 75% utility (from 50%)
- Scenario 3: From 3 min to 2 min recovery gives: 50% utility (from 30%)

Total benefit on tactics 1, 2 and 3:
B_1 = (75-40) x 30 + (60-50) x 20 + (80-30) x 20 = 2450
B_2 = (50-40) x 30 = 300
B_2 = (75-50) x 20 + (50-30) x 20 = 900

Return-on-investment on tactics 1, 2 and 3:
ROI_1 = 2450/15000 = 0.163
ROI_2 = 300/4000 = 0.075
ROI_3 = 900/3500 = 0.257

Ranking of tactics (from best to worst):

1. Tactic 3 Improved exception handling
2. Tactic 1 Replicated servers
3. Tactic 2 Improved computational efficiency

Problem 5 Design a software architecture using ADD (30 points)

Read the description of the Tippeliga-Ticket system below and do an architectural design using the attribute-driven design (ADD) method described in the textbook. Your answer should include:

- Architectural drivers
- Architectural tactics and patterns
- A logical view
- Interfaces
- Verification of the architecture

Note that you should only describe the logical view and only do one level of decomposition! Motivate for your choices and state your assumptions.

Tippeliga-Ticket System (TTS)
The Tippeliga-Ticket System (TTS) is a system where the users can buy tickets using credit cards to football (soccer) matches in the Tippeliga (highest division in Norway) over the Internet using a Web-browser. The user can look at information about future matches from football teams from all over Norway, and see if there are any available seats. The information about the football matches is retrieved from various servers with different interfaces provided by the different teams. Note that the teams in the Tippeliga will change every year. It is critical that the TSS is available to the users all the time, and it cannot be unavailable for more than 2 minutes a week. Before important games, such as Champion League games, it is important that the system does not break down even if over 40000 users try to get tickets at the same time.

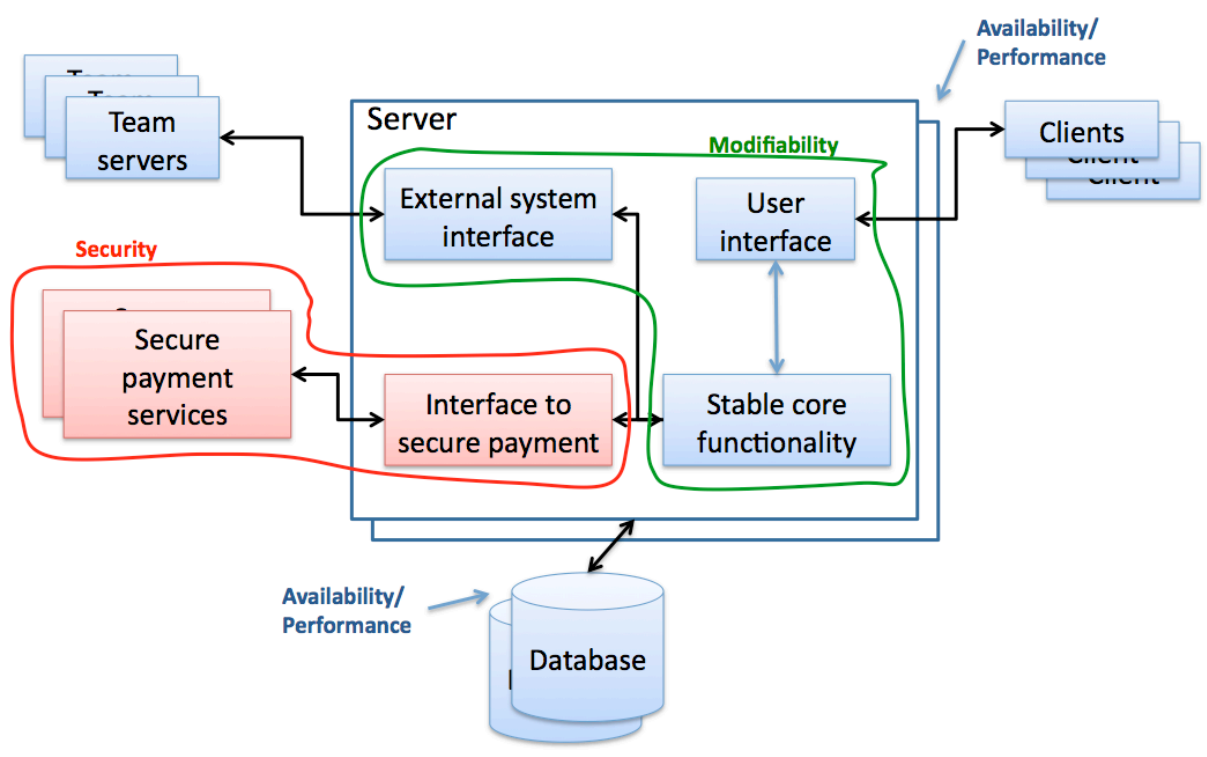
Solution:
Here is the solution according to the attribute-driven design (ADD) process.

Step 1. Choose module to decompose:
The Tippeliga-Ticket System

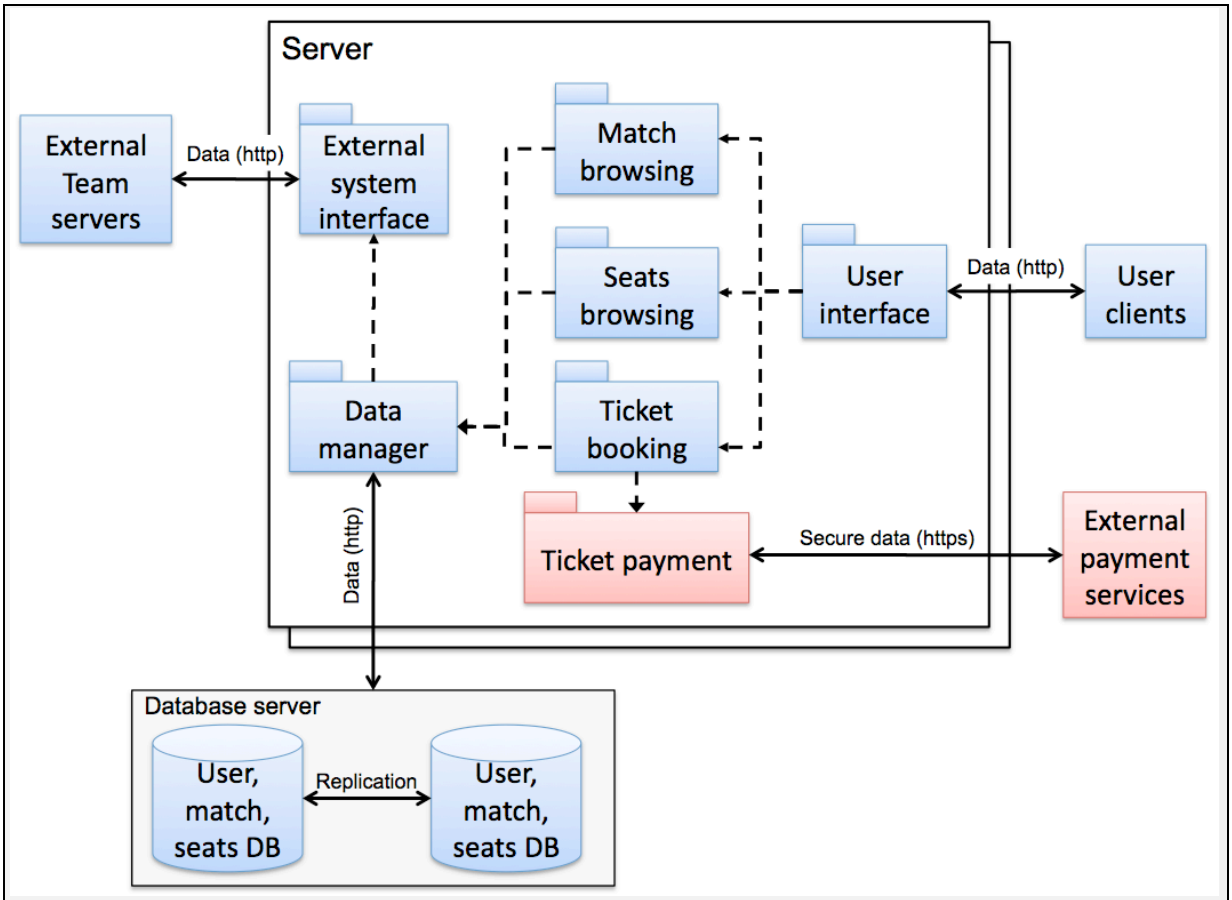
Step 2a. Choose architectural drivers:
AD1: The system cannot be down more than 2 minutes a week (availability)
AD2: The system should provide secure electronic payment (security)
AD3: The system should be able to communicate with various team servers (modifiability)
AD4: The system must handle 40,000 simultaneous users (performance)

Step 2b. Choose architectural patterns:
Tactics for security: Firewall for server, autorize users, authenticate users, Payment handled by third-party and usage of secure connection (e.g. https)
Tactics for modifiability: Divide the functionality into coherent units and plan for changeable interfaces for external systems.
Tactics for performance: Use replication of the server to ensure support for many users, and cache data from team servers.
Tactics for availability: Use replication of the server to cope with downtime.

Architectural patterns:
Use a kind of a model-view controller pattern where the view and the controller is represented in the user interface part and the model is the database. Replication is used on server and database for higher performance and availability. Separation of concern is used to provide modifiability (separate core functionality and external interfaces), and a separate part dealing with secure computation and interfaces.



Step 2c. Instantiate Modules and Allocate Functionality
Applied architectural patterns and instantiated functionality for match browsing, seats browsing, and ticket booking. Also added a data manager that takes care of caching of data from external systems (stored in database).



Step 2d. Define Interfaces of the Child Modules

Interfaces between the server and the external parts of the system:

- Between User interface and user clients: HTML over HTTP
- Between External system interface and External Team server: XML over HTTP
- Between Data manager and Database server: SQL over HTTP
- Between Ticket payment and External payment service: encrypted binary over HTTPS

Interfaces within the server:

- The User interface class provides display methods that uses other classes:
- displayMatch()
- displaySeats()
- displayBooking()

The Ticket payment class offer the method payTicket (used by the Ticket booking class)

The Data manager class offers methods that are used by three other classes:

- getMatchInformation()
- getSeatsInformation()
- getBookingInformation()

The External system interface class, offers a more general method for accessing information from external team servers:

- getInformation()

Step 2e. Verify Use Cases and Quality Scenarios
Check that functional requirements are covered:

- The user can get information about future matches: Match browsing class
- The user can get information about available seats: Seats browsing class
- The users can buy tickets: Ticket booking and Ticket payment classes

Check architectural drivers:

- AD1: The system cannot be down more than 2 minutes a week (availability): Replication of the server and the database, and caching of external servers.
- AD2: The system should provide secure electronic payment (security): Use secure computation in Ticket payment class, use secure transfer (https) and secure external payment service.
- AD3: The system should be able to communicate with various team servers (modifiability): Supported through the External system interface class.
- AD4: The system must handle 40,000 simultaneous users (performance): Replication of the server and database, and caching of external servers.