UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Neo Tandem Technologies

Eye Tracking

Author:	*Student number:*
Duran Cole	u13329414
Michael Nunes	u12104592
Molefe Molefe	u00000000
Tebogo Seshibe	u00000000
Timothy Snayers	u00000000

July 5, 2015

Architecture requirements

Eye Tracking

Github Link:

Version: Version 0.2 Alpha July 5, 2015

Contents

1 Architecture requirements . . . 2
  1.1 Quality Requirements . . . 2
  1.2 Integration and access channel requirements . . . 3
  1.3 Architecture constraints . . . 3
2 Architectural patterns or styles . . . 3
  2.1 Name of pattern or Style . . . 3
3 Architectural tactics or strategies . . . 4
  3.1 Name of Tactic or Strategy . . . 4
4 Use of reference architectures and frameworks . . . 4
5 Technologies . . . 4

1 Architecture requirements

1.1 Quality Requirements

1.1.1 Scalability

The requirements for scalability in the program are that the storage of all the data will forever expand with the more uses of the program.This means that the longer you have the system the more data it will take up,This then states that we nee to ensure that later on the space will not impact on the users performance.The program itself can also grow in terms of features.This means that a developer can extend system by adding a new feature.

- Tactics and strategies
  - Compression:compression of the files will allow more files to be stored on a hard drive.This can lead to greater scalability and improve space management.
- Integration : The integration of scalability in our system will allow the users to decide where the files that are written will be saved.This will enable them to save the files to hardrives as time with the system goes on.This allows the user to

expand their storage capacity. The system is also created in a way that everything is a module/component.The program allows for new modules to be created and then added.This encourages the growth of the system from the start as new and better components can be added.

1.1.2 Performance

Performance of the system is key as the system is constantly reading in data as the eye is being tracked so the performance needs to be optimized so that the tracking of the eye does not stutter and then incorrectly track the eye.The system also needs to perform well and not cause issues when it is run.

- Tactics and strategies
  - Thread pooling:this will allow us to spread the work load across threads so that they can run concurrently and reduce bottlenecks.this also allows us to optimize how we can use the hardware so that we use it to its maximum potential.
- Integration : The system will be able to run the server that will store the data to run concurrently with the eyetracking function.Each of these is run on a thread and this allows them to run at the same time so all the data is collected in real time and the performance is optimized.
  The system is spread into components which means that each component is responsible for a task and this increases performance as there are not waiting processes.

1.1.3 Maintanability

The system will need to be maintained over time as new technologies are introduced.The system is component based wich makes maintainability easier as the components need to be maintained and not the entire system.Maintainability allows a program to be kept up-to-date.Changes in eye tracking is a common thing so we expect new changes to occur frequently.

- Tactics and strategies
  - Removal of Faulty Components :This allows us to remove components that are no longer working in the system and then replace them with a newer version of them all and if needs be a whole new type of component.
  - Deadlock Detection :The detection of deadlock is important and if a deadlock occurs could harm your system.This is important as detecting a deadlock can also then determine if the component is faulty.
- Integration : The system is component based so removing and maintaining the system is made easier.If we know that a certain part of the system is causing a problem such as deadlock or is faulty in any other way we can then decide to remove that component and fix it or add a new system.This way changes are made to a component and not the whole system.

1.1.4 Reliability and Availability

Reliability and Availability are important in this system.The server which is used for capturing the data needs to constantly be available to the program so that data can be read.The program also needs to be reliable and gather the information correctly.The collection of incorrect data can cause mishaps with the system.The system as a whole needs to be running and not crash and fail.

- Tactics and strategies
  - Load Balancing :By balancing the work load we can ensure that a system can continuously run and not have a crash.This call also cause services to constantly be available such as the server.
  - Removal of Faulty Components :The removing of faulty components allows us to increase the reliability of the system by removing the cause of the unreliability.
- Integration : The load balancing can be achieved by separating the specific tasks into different components.This allows the system to share the load across different sub sections.The removal of the components can be done manually so that the system does not contain the faulty component.

1.1.5 Security

The system is connected to the server and this can lead to a security threat as the connection can be compromised and allow an attack on a user.This is a security concern as the system can be indirectly used to harm the user of the system.

- Tactics and strategies
  - Request/Connection Dropping :Dropping the connection to the server can then reduce the risk of a users security to be compromised and thus increasing security.The system must also be secure as not to harm the hardware itself and to also not damage any accompanying hardware such as the eye tracker.The tracker can also be used by a malicious user to spy on the systems users.
- Integration : When the system no longer needs to use the the server to get the data the system must cut the connection to the server and then restart the connection once the server will start connecting once more to the server.This reduces the risk of the channel being used by someone malicious and harming the users of the system.

1.1.6 Usability

When a user needs the system it will need to be usable,in other words the program should allow the user to explore all the features of the system and be able to use them all.

- Tactics and strategies
  - Contracts based decoupling :The system is decoupled by dividing the system into a set of contracts that each have its specification and have its pre and post conditions.This allows the system to be easily decoupled.
- Integration : The contracts created are used to make the system in to components which allows new contracts to be created as the system develops and there is no tight coupling amongst all the components hence there is no reliance on them.

1.1.7 Testablity and Integrability

The system is quite a large system and the testing would need to be done on every aspect of the system this would mean that the system would need to have testing done all over.The system also needs to be integrated with all the systems components and this would mean that the system would need a high level of integration.

- Tactics and strategies
  - Layering :this will allow each component to be comprised of a layer and this leyer can then be changed or removed.This allows good integration in the components.
- Integration : Each layer will have a set of components and these components can be changed and then integrated easily with the layers that are next to them.This allows the system to ave easy integration and allos changes to the system not effect the integration.The testing is made easy with each component allowed to be tested on individually so that the integration can be done seamlessly.

1.2 Integration and access channel requirements

1.3 Human Access Channel

The program will be accessible by human users via a computer that has and that can support the eye tracking equipment. The Eye Tribe eye tracking equipment will only function with a desktop computer or a laptop that supports USB 3.0 technology. The program will only run on a Windows operating system. The user will need a network connection to the OGAMA database.

1.4 System Access Channel

The only systems that are supported by this program are desktops and laptops that have support for USB 3.0. These computers must be running a Microsoft Windows operating system from Windows XP or higher. Framework .NET 4.0 or higher is needed to run the program.

1.5 Integration Channel

The program with OGAMA will use a database that stores all of the eye tracking data. The database can be either a local database or a remote database or both.

1.6 Architecture constraints

1.7 Hardware

The Eye Tribe eye tracking device will be used to test the extensions which will be implemented through this project. The Eye Tribe eye tracking device connects to the computer through a USB 3.0 port and as such It is required that the computer the application would be running on has USB 3.0 ports.

1.8 Operating System

The OGAMA Application only supports installation on computers running a Microsoft Windows operating system. The latest version was only released in 32bit, but can run on 64bit systems.

1.9 Additional Software

To run the OGAMA application you need to have the .net 4.0 Framework installed and depending on the version of the application you are using you would also need SQL Express installed.

1.10 Version Control Management

Github will be used to as the version control management system for the duration of this project

1.11 Development Environment

The software extensions to be added to the OGAMA application will all be programmed in the C sharp programming language through Microsoft Visual Studio 2013.

2 Architectural patterns or styles

2.1 Name of pattern or Style

The project can easily be split as: User, Interface and, Background Calculations. The user interacts with the Interface, the program that will be executed, handling the eye tracking. The interface will convert the received data into a format that the data processing section can work with. The data processing section will take render out/back the heat maps generate from the users focus.

It is with this in mind that the Model-View Controller (MVC) design pattern will be used. The MVC separates the data model, UI, and control logic which is exactly what we want. We want the user to be able to use our system without having to know how it works. Only the basics should be available i.e record, printing out texture/model.

3 Architectural tactics or strategies

3.1 Availability

This refers to whether the user will have constant access to the program and methods to avoid interruptions.

Because of the way in which the system will work, all aspects of the systems will be working as long as it is running. Therefore, an exception based fault detection system. Unless it is the user interface that fails, the exception will be handle in the background, and a connection between the sections of the systems will be re-attempted.

Checkpoints will be saved in order to roll back in the event of errors.

3.2 Modifiablity

This refers to time and cost relating to development, changes, and testing of the system.

The MVC pattern allows us to separate concerns of the system allowing for finer grain implementation and changes to its inner working. This will also allow for potential changes to be ”plugged in” if required.

This is also means that testing will be easy as each section simply needs to test whether they are able to do their own required task. All related task will simply be using the abstraction of the subsystem.

3.3 Performance

This refers to the speed of execution, the time between a request and a response between modules within the system.

In order to simplify the subsystems, any optimised, pre-made algorithms that can help with their executions will be used. As few as possible intermediaries will be used.

3.4 Security

This refers to the safety of the data that the system will be used.

The system is to be designed in such a way that no users can interfere with the inner workings of the subsystems. The overall purpose of this systems is provide extra functionality to a device and so, changes to how it works are not required.

3.5 Testability

This refers to the ease of testing of the system, during successive build releases.

During each build release, all test cases used in the subsystems unit tests, will be combined into a series of big test cases in order to make sure that the expected input provides the same expected output.

3.6 Usability

This refers to the ease of use the user will have with regards to the system.

Since the system will be doing most of the work, the user will only be provided with a very simple interface to interact with the basic functions i.e record screen, print to texture.

4 Use of reference architectures and frameworks

4.0.1 Reference Architecture

Say what you want to say for the reference architecture

5 Technologies

We have chose the following technologies carefully as they incorporate into one another to form a sound basis for our project to operate on.

- **Name of tech** say stuff and repeat
- **Name of tech** say stuff and repeat
- **Name of tech** say stuff and repeat
- **Name of tech** say stuff and repeat

Bibliography