Hellaynnea Consulting
hellaynnea@gmail.com

Net Multiplayer

Software Development Plan

Version 1.0

(Ref.: NMP_PDL_YL)

25 January 2006

Table of Content

I Introduction

This document gives a plan for the development of Net Multiplayer, remote monitoring software.

This document will keep tracks of working achievements of this software development. It will be updated all along the development phases, reflecting its state.

The chosen development method and the project organization will be described in this document. For each of the iterations, there will be a detail of the tasks, a calendar and the prototype obtained at the end of the iteration. Afterwards, the risks should be identified and the ways to limit or avoid them should be given. By the end the standards and necessary tools will be exposed.

II References

- The software development plan (SDP) of the Net Multiplayer software system. This document describes the full architecture of the project and interactions between the different pieces of software involved.

III Development Methodology and Calendar

III.1 Development Methodology

This project will be developed using an iterative method. It consists in realizing prototypes regularly enriching functionalities and features at every new step until completion. In this way, for each of the iterations, the software architecture will be updated and this document will be completed.

This methodology allows presentation of prototypes to check perfect matching with the specifications. Above that, using such a methodology will make it possible to see whether the development orientation taken fits the needs or not. Indeed, this project is intended as a futuristic realization and every step taken will be in an unknown field. This methodology will lower the risks of developing too complicated things and avoid non-ending developments.

The project will be derived in 5 iterations:
Iteration 1: Project organization
Iteration 2: Prototype 1
Iteration 3: Prototype 2
Iteration 4: Prototype 3
Iteration 5: Complete system

III.2 Iterations Objectives

- Iteration 1 :
  - Project organization and meeting with project's supervisor.
  - First steps with the architecture
- Iteration 2 :
  - First prototype presentation: fully functional network between a light client and a server.
  - The software architecture should be largely defined for the server.
  - The client's graphical engine development plan should be clarified.
- Iteration 3 :
  - Second prototype presentation: The server's video channels should be fully functional to allow the client to get proper images.
  - The graphical engine should be implemented: we should be able to see video images and interact with the client.
- Iteration 4 :
  - Third prototype presentation: The graphical engine should be complete.
  - The monitoring of video-cameras (PTZ) should be started.
- Iteration 5 :
  - Complete project's presentation.
  - Both the server and the client should be complete. The SipPilot software should be done also.

III.3 Deliveries

Iteration	Documents and products
1	Work report Software architecture document Software development plan
2	Work report Prototype 1 + presentation Software development plan Software architecture document
3	Work report Prototype 2 + presentation Software development plan Software architecture document
4	Work report Prototype 3 + presentation Software development plan Software architecture document
5	Work report Complete software + User's manual Software architecture document Software development plan Project results

III.4 Calendars

The project's main objective is to deliver a complete system for April 31st 2006. Intermediate deadline are give as follows:

Iteration 1: January 9th 2006
Iteration 2: January 23rd 2006
Iteration 3: February 17th 2006
Iteration 4: March 15th 2006
Iteration 5: April 31st 2006

IV Iterations Scheduling

IV.1 Iteration 1

The goal of this iteration is to organize the project:

- Software development plan's redaction
- Server software architecture done except video channels
- Project's development framework created

IV.2 Iteration 2

Net Multiplayer Server:

- Sites management ready
- Users management ready
- Network ready
- PTZ network ready
- Video channels: light version, simple live video requests.

Net Multiplayer Client:

- Network ready
- Sites management ready
- Users management ready
- Video channels: light version, simple live video requests. – No display yet !
- GTK integration ready
- GUI and Graphical Engine development plan ready

Prototype #1:

- Shows fully functional data exchanges between server and client
- Net Multiplayer Server should answer services requests, save and load configuration files.
- GTK GUI should work to display network results.

IV.3 Iteration 3

Net Multiplayer Server:

- Video channels ready

Net Multiplayer Client:

- Video channels ready with display
- GUI integration should be complete, all GUI should be displayable
- A light version of the graphical engine should be functional: channels displays, selection, cameras management, multi-visions management.

Sip Pilot:

- Ready

Prototype #2:

- Sip Pilot should be integrated on a Secur'IP Box and tested.
- Net Multiplayer Server should be complete except for PTZ management.
- Net Multiplayer Client's GUI should be complete, the graphical engine should be functional.

IV.4 Iteration 4

Net Multiplayer Server:

- Completed – PTZ management included

Sip Pilot:

- Completed

Net Multiplayer Client:

- PTZ requests integration
- Graphical engine optimization to gain FPS
- Client and Server administration GUI should be modeled

Prototype #3:

- Full interaction between client and server
- PTZ interaction with the video-cameras should appear

IV.5 Iteration 5

Net Multiplayer Client:

- Completed – Full PTZ interaction with the cameras
- Full optimization of the visualization

Complete System:

- Fully functional system
- Tests done in major cases
- User's manual redaction

IV.6 Graphical Engine Development Plan

This part describes the development plan for the graphical engine.

IV.6.a Framework design

Here we will try to design a basic framework for objects rendering in an OpenGL and GTK context with alternating 3D and 2D scenes.

Plan:

- Defining widgets architecture
- Framework design (in MT context) OpenGL and GTK
  - Try fullscreen rendering (Game Mode)

IV.6.b Basic rendering

Here we will use the framework built in the last section to perform objects rendering performance tests. We will also design algorithm for objects animation. The internal management of graphical widget will also be discussed.

Plan:

- Widgets rendering and selection
  - Performance tests
- Widgets self-animation
  - Performance tests
- Widgets internal management

IV.6.c Video-channels design

Here we will focus on textured widgets. We will implement our first video channels with steady and changing textures. We will also design multi-leveled channels decoration.

Plan:

- Video-channels design
  - Performance tests with fixed texturing in multi-channels context
  - Performance tests with changing textures in multi-channels context
- Video-channels decoration (putting buttons and interactive areas on it).

IV.6.d Multivision design

In this part we will create multivisions from an offscreen and onscreen point of view. How graphical multivisions can be saved and loaded, channels created and moved dynamically, layers integration and so forth. After this part the graphical engine will be considered completed.

Plan:

- Layers building (alpha-blending and hierarchical widget models)
- Multivision building
  - Aligning channels
  - Saving/Loading
  - Managing layers
  - Performance tests and analysis
- Multi-multivisions building: aligning and switching between several multivisions.

V Risks

The main risks are indicated below with possible actions to reduce or avoid them.

Risk	**Actions**
Slow system due to heavy graphical engine.	Switch in configuration to turn down high resolution graphics. Limit number of cameras on the screen at the same time and reduce the FPS for the overall multi-vision.
Network overhead for PTZ requests	Switch from UDP to TCP for PTZ requests.
Security breach – Easy system intrusion	Create password hash; require proper identification for every request.

VI Methodology and Tools

The development method and software architecture will use formal UML for statistical structure description, activity diagrams and objects interaction's description. Some modules internal behavior may be described using state charts.

The tools used will be:

- Visual Studio .Net for the programming parts
- MS Visio for the designing parts
- OpenGL for the graphical engine
- GTK for the Graphical User Interfaces
- 1 Portable computer for project presentations and development
- 1 Secur'IP Box for distant cameras visualization
- 1 NMRouter for distant cameras visualization
- MS Word and MS Excel for all documentation.
- A FTP server will be used to store project's backups.