ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ

Διδακτική της Τεχνολογίας και Ψηφιακών Συστημάτων

ΤΙΤΛΟΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

"Pervasive Computer Systems and Applications: Digital Signage"

Η εργασία επιμελήθηκε από τον:		
Όνομα	:	ΓΕΩΡΓΙΟΣ
Επώνυμο	:	ΣΙΓΑΛΑΣ
Επιβλέπων Καθηγητής :		
Δρ. Νικήτας-Μαρίνος Σγούρος		

1 Table of Contents

2 Table of Figures

3 Project Title

Pervasive Computing Systems are focusing on the development of applications that can be an integrated part of our environments without using classical types of computer hardware and where the user can interact with them by natural way and without using computer equipment (e.g. keyboard, mouse).

Project will focus on the development of a completed environment of editing and distributing digital signage content. The output devises for the content will be rear projection screens where the user can interact through touch (rear projection touch screens).

4 Introduction

Project approach will focus on both the marketing and the technology aspect of digital signage. In chapter 6 the term “Digital signage” will be determined and analyzed thoroughly. The main factors of a successful digital signage approach will be examined.

In chapter 7, a detailed introduction on pervasive systems will focus in the future visions on “Disappearing of Computing”. Information on pervasive display systems and touch screens will be provided in this chapter, since the output devices of the implementation of this project will be rear projection screens where the user can interact through touch.

In the following chapters there will be information about the emerging technologies. Our approach used specific architecture and technologies. This does not mean that there are not alternative solutions. The alternative solutions have been recorded and evaluated.

Some of the factors that have been taken under consideration are:

- Productivity
- Design
- Scalability
- Interoperability
- Security
- Maintenance
- Deployment

Database design is presented in chapter 11. Basic functionality is presented in chapter 13 and finally in chapter 14, this functionality is presented in action via a test case, where screenshots illustrate the components and the way that they interoperate with each other.

5 Terminology

Calendar (Windows Form Application: Calendar.exe): Calendar is a tool that allows user to schedule updates for the Players.

ClickOnce: ClickOnce deployment allows you to publish Windows-based applications to a Web server or a network file share for simplified installation. Visual Studio provides full support for publishing and updating applications deployed with ClickOnce. ClickOnce deployment is available for projects created with Visual Basic, Visual C#, and Visual J#, but not for Visual C++.

Controls: The System.Windows.Forms namespace provides a variety of control classes that you can use to create rich user interfaces. Some controls are designed for data entry within the application, such as TextBox and ComboBox controls. Other controls display application data, such as Label and ListView. The namespace also provides controls for invoking commands within the application, such as Button. The WebBrowser control and managed HTML classes, such as HtmlDocument, let you display and manipulate HTML pages within your managed Windows Forms application.

Designer/Developer: Designer is the specialist who is responsible for the implementation of the Software Version of a Player.

Digital Signage: Digital signage is a name given to any number of methods used to display multimedia content in public venues. Alternatively known as dynamic signage, electronic signage or narrowcasting, networks of digital signs have been deployed across numerous retail chains, banks, travel hubs and corporate headquarters to deliver informative and entertaining content to captive audiences and passers by.

Network Manager (Windows Form Application: Network_Manager.exe): Network Manager manages Players, Content Updates and Software Updates.

P2P Server (Windows Form Application: P2P_Server.exe): P2P Server broadcast messages (requests) to peers (Players).

Peer-to-peer (P2P): The term “peer-to-peer” (P2P) refers to a class of systems and applications that employ distributed resources to perform a critical function in a decentralized manner. The critical function can be distributed computing, data/content sharing, communication and collaboration, or platform services.

Pervasive Systems: Pervasive computing is a cross-disciplinary area of research that extends the application of computing to diverse usage models [1]. It is viewed as the next generation computing environment with information and communication technology everywhere, for everyone, at all times. Information and communication technology will be an integrated part of our environments: from toys, milk cartons and desktops to cars, factories and whole city areas - with integrated processors, sensors, and actuators connected via high-speed networks and combined with new visualization devices ranging from projections directly into the eye to large panorama displays.

Player (Windows Form Application: PlayerX.exe): Player is the application on the remote machine that presents content to users.

Publishing Location: Publishing Location could be a Web Server, a File Server or a File Share.

Schedule Server (Windows Form Application: Schedule_Server.exe): Schedule Server checks if there is a request for a scheduled update to commit it.

Smart clients: Smart clients are easily deployed and managed client applications that provide an adaptive, responsive and rich interactive experience by leveraging local resources and intelligently connecting to distributed data sources.
A smart client:

- has local resources and user experience
- is connected
- is offline capable
- has intelligent deployment and update

6 Digital Signage

6.1 What is Digital Signage?

Digital signage is a name given to any number of methods used to display multimedia content in public venues. Alternatively known as dynamic signage, electronic signage or narrowcasting, networks of digital signs have been deployed across numerous retail chains, banks, travel hubs and corporate headquarters to deliver informative and entertaining content to captive audiences and passers by.

In its most basic implementation, a digital sign consists of a playback device (such as a computer, VCR or DVD player) connected to a display. Depending on the application, the display might be a small LCD screen, a plasma display panel, or even a video wall composed of a number of connected screens. With a number of affordable options available, anybody with a message to send to their out-of-home audience can benefit from a digital signage installation. [11]

6.2 Making Digital Signage Work

The concept of out-of-home messages is not new. Billboards, window treatments and point-of-purchase displays are widely used for out-of-home advertising, while bulletin boards (both traditional and electronic), flyers, faxes, memos, and email have been used to send corporate communications and educate employees for decades. However, true dynamic signs first came into popular use with the advent of in-store closed circuit television networks in the 1970s. With the widespread availability of affordable VCRs, retail stores and corporate headquarters were able to play back pre-recorded content to their patrons and employees, providing timely information and entertaining content. Soon, closed circuit networks would be augmented (and in some cases, supplanted) with affordable TV/VCR combination units for smaller displays, and projection screens and video walls for eye catching, large format presentations. With satellite distribution, it even became (relatively) affordable to syndicate the same content to thousands of sites at once.

In recent years, several factors have combined to make digital signage a more powerful, eye-catching, and affordable display medium than ever before, contributing to its widespread adoption. Key factors include the nearly ubiquitous availability of high-speed Internet access, new large format displays like plasma screens and LCD panels, and new compression formats that can compress large amounts of content into small file sizes. A modern digital sign adds several additional components to the traditional setup described above. The controller, typically a powerful computer or media playback appliance, uses a digital connection to deliver a crisp output signal to a digital display, like a plasma screen or LCD panel. The playback device uses a digital storage medium (such as a hard drive or solid-state

flash disk) to store digital content locally, ensuring smooth playback. In many cases, the device can be remotely managed over the Internet to allow for content updates, schedule changes, and compliance reporting. In its simplest implementation, a digital signage installation includes a content stream (from a CD, DVD or the Internet), a playback device, and a display (e.g. a television, plasma display, or LCD panel).

Even the smallest digital signage networks can benefit from remote management. Aside from offsetting the costs of producing, replicating and distributing VHS tapes or DVDs, Internet connected signage devices provide network administrators with the ability to closely monitor playback, ensuring that the desired content is being displayed. Delivering content over the Internet ensures that the content arrives and is displayed according to the proper schedule, eliminates errors in shipping and handling, and removes reliance upon on-site personnel to change tapes or DVDs (since these individuals are typically not very motivated to perform such tasks).

Additionally, compliance reporting gives network owners a complete record of what every screen has displayed. Finally, user-friendly management tools mean that people throughout the organization - from advertising and creative services to operations and general management - are empowered to change content and generate reporting metrics.

More sophisticated remote management suites also give network operators the ability to perform near real-time adjustments to every display's playback schedule, enabling real-time marketing experiments, emergency announcements, and even live content feeds. [11]

6.3 Choosing the Right Digital Signage Technology

Choosing the right digital signage technology depends on the intended application. For example, a digital signage application for internal corporate communications would probably benefit from large, eye catching plasma displays placed in common areas such as cafeterias and break rooms. Depending on the size of the deployment, the network owner might opt for either local or remote management. On the other end of the spectrum, a manufacturer looking to improve their point-of-purchase advertising displays might choose to employ small, lightweight LCD panels in conjunction with traditional product displays in an aisle or end cap fixture, managing the content centrally via a web-based interface.

Of course, budget constraints must also be taken into consideration. Digital signage applications that require large displays have a number of options to choose from, including rear projection TVs, LCDs, plasmas, DLPs, wall projectors, and traditional CRTs. Smaller signs are typically either small LCD or CRT displays. Touch screens are available for virtually any sized display, and can add an interactive component to an otherwise non-interactive display medium.

Finally, though many digital sign networks start modestly, it is important to plan for the future. A network of a few screens may be easy to manage by shipping and swapping DVDs. However, if that network grows to 25, 50 or 100 screens, this type of content management

becomes much more cumbersome. Additionally, features that may not seem relevant during the early stages of a deployment might prove to be essential later on. For example, the ability to add or delete a piece of content on short notice might not seem important to a small sign network owner until the necessity arises, often at the insistence of a large advertiser. Similarly, other business opportunities may rely on advanced features like real-time scheduling or live content insertions. What's more, the availability of turnkey, hosted digital signage software can actually make it cheaper to deploy the initial systems with full remote management capability - providing a solution that is affordable for today and scalable for tomorrow. [11]

6.4 Key Markets

Digital Signage can be found in a variety of key market segments including:

- **Retail Stores**: Increase revenue potential with dynamic eye-catching ads for plasmas, video walls, billboards or interactive kiosks. Easily broadcast or narrowcast national or local scheduled messages. Interface with POS systems and RS-232 devices. Monitor playback status and measure ad run times for billing.
- **Corporate Communications:** Improve company communications with the power of television in work areas, lobbies and offices. Easily create and customize in-house TV channels, presentations or training programs. Boost morale and maximize productivity by getting the right message to the right people at the right time.
- **Hospitality and Entertainment:** Improve your guests' experience by helping them find and enjoy everything your location has to offer. Distribute video and multimedia messages to multiple locations or target a specific screen to deliver valuable updates, directions or schedules directly to the people who need it.
- **Cable TV Channels:** Create your own high quality TV channels with text, graphics and video that can be scheduled and updated even while on the air. Enhance your local programming or photo advertising with data, weather and news feeds, and show digital video or live input via remotely controlled switchers and decks.
- **Government Communications**: Quickly update visually impactful messages for public information, emergency management or military communications. Target your communications wherever and whenever you need. Instantly display important updates as full screen displays or crawls.
- **Education**: Create powerful educational media for TV and interactive applications with ease and speed. Combine full screen effects, animation and video to communicate with impact.
- **Transportation**: Provide real-time scheduling and directions to people waiting for trains or airplanes. Create airport and railway control systems that provide scheduling, On-Time/Late messages, availability and directions.

- **Hospitals & Medical Offices**: Inform the public of the latest innovations in medical technology, provide directions using interactive Way finder maps, educate patients about the latest medicines and healthy lifestyles.
- **Financial Centers & Banks**: Advertise products and services provided by your financial institution while your customer waits in line for a teller or an ATM machine. Provide real-time stock information to investors in remote offices from a central location.
- **Casinos & Gambling Venues**: Locate restaurants, entertainment venues, hotel rooms, and casino services by strategically placing large-format plasma or LCD screens or interactive kiosks.
- **Cinemas**: Increase the revenue from your snack bars by using dynamic menu boards that display snack choices and pricing to entice customers to spend more while they enjoy their cinema experience. Display movie listings, schedules and availability of tickets. [10]

Figure 6-1 : *Key Markets*

7 Pervasive Computing

Pervasive computing is a cross-disciplinary area of research that extends the application of computing to diverse usage models [1]. It is viewed as the next generation computing environment with information and communication technology everywhere, for everyone, at all times. Information and communication technology will be an integrated part of our environments: from toys, milk cartons and desktops to cars, factories and whole city areas - with integrated processors, sensors, and actuators connected via high-speed networks and combined with new visualization devices ranging from projections directly into the eye to large panorama displays. [2]

7.1 The Disappearing Computer

It seems like a paradox but it will soon become reality: The rate at which computers disappear will be matched by the rate at which information technology will increasingly permeate our environment and our lives.

Computers are with us everywhere and we are aware of their increasing significance for our lives. In parallel, the spread of computers caused a shift in our activities: away from real, physical objects in the environment as the sources of information toward computer monitors as the interfaces to information.

This shift had implications for the design of information systems. Computers became primary objects of our attention resulting in an area called “human computer interaction.” Today, however, we must ask: Are we actually interested in interacting with computers? Isn’t our goal rather to interact with information, to communicate and to collaborate with people? Shouldn’t the computer move into the background and disappear?

This disappearance can take different forms: physical and mental disappearance. Physical disappearance refers to the miniaturization of devices and their integration in other everyday artefacts as, for example, clothes. In the case of mental disappearance, the artefacts can still be large but they are not perceived as computers because people discern them as, say, interactive walls or interactive tables. This leads us to the core issue: How can we design human-information interaction and support human-human communication and cooperation by exploiting the affordances of existing objects in our environment? And in doing so, how do we exploit the potential of computer based support augmenting these activities?

The increasing ubiquity of computers and related devices (such as sensors) and their diffusion into our environment requires reconsidering of the complex interplay between technology and the human. One early view was expressed by Mark Weiser, who observed “that the most profound technologies are those that disappear” [6], arguing for a vision of an unobtrusive computer technology called “calm technology.”

Ubiquitous computing (also known as pervasive computing, proactive computing, ambient computing, and, in related contexts, ambient intelligence) is now an area of significant

research activities worldwide both commercially and in academia; and is central to the research strategies of many national and transnational organizations [2–5]. Within this significant and emerging field of research, agendas have been evolving highlighting different aspects.

Within Europe, the proactive research initiative "The Disappearing Computer" (www.disappearing-computer.net) was envisaged to explore how daily life can be supported and enhanced through the use of collections of interacting artefacts.

Together, these artefacts will create new people friendly environments where the "computer-as-we know-it" has no role.

7.2 Future Visions on Disappearing Computer

This special section presents a collection of different perspectives, reflections, and future visions on the disappearing computer. They were especially inspired by discussions and debates at the April 2004 EU-NSF joint advanced research workshop [1] in Vienna. Indeed, strong themes emerged from the workshop. They cover basic technology and infrastructure issues, the role of sensors, and the pressing issues of privacy and security, as well as how to design the interaction of humans with computers that disappear.

The recurring themes throughout this section and these efforts include:

- **Interaction design**. As computers disappear from the scene, become invisible, and disappear from the perception of the users, a new set of issues is created concerning the interaction with computers embedded in everyday objects resulting in smart artefacts: How can people interact with *invisible* devices? How can we design implicit interaction for sensor-based interfaces? How do people migrate from traditional explicit to future implicit interaction? How can we design for transparency and coherent experiences? Returning to the real world as the starting point for design and trying to exploit the affordances of what real-world objects provide seems to be one way of tackling these problems. Therefore, a major approach in this domain is and will be to combine the best of real and virtual worlds resulting in hybrid worlds.
- **Sensing and context.** How can we sense and capture the world around us, the parameters of our external physical and internal (for example, body) environments that inform and guide human behaviour? What are the relevant parameters that can be used by the systems to support us in our activities? Location is certainly central, but it is one parameter of a larger set determining the overall context. If context is key, what constitutes context and how can it be captured, processed, and exploited for providing the services appropriate in a given situation? How do we arrive at context-aware systems? Does the collection of every facet of the sensed world, storage of every bit of information, and predicting user behaviour point in the right direction? Are the underlying mental models of interaction and perception sufficient? It seems there are still gaps toward solutions for real-world situations, not only in terms of scale but also regarding open questions and decisions such as: How much

should the system (or the infrastructure) remember? When does the system or the infrastructure) try to predict the user's intentions and when are the users presented with choices?

- **Essential infrastructure.** Any infrastructure deployed to support ambient and ubiquitous computing must, by definition, be long lived and robust. Consequently new approaches to the evolution of the infrastructure, *in situ* upgrade and update, will be required. Given the potentially vast collection of devices, sensors, and personalized applications, this update problem is significantly more complex than previously encountered. Additionally, since the infrastructure is meant to be invisible it will be necessary to develop an understanding of what failure means and how malfunctioning is communicated to the users. Consequently, new approaches to developing robust systems and applications will be required; ones that are fault tolerant, highly available, and that degrade gracefully.
- **Discovery.** One of the key requirements to the provision of any disappeared computing infrastructure is an approach or service capable of assimilating and filtering information from various sources and determining relevance. This is essential to allow the user and the application to discover the necessary information from the environment to achieve a defined goal or complete an activity.
- **Privacy, trust, and security.** The vast amounts of personal information collected by ubiquitous systems has led to growing concerns about the security, privacy, and trustworthiness of such systems and the data they hold. The areas of security, privacy, and trust are critical components for the next stages of research and deployment of ubiquitous systems. Moreover, it was identified that these observations are not merely an amplification of the current concerns of Internet users with desktop computers. New approaches are required that take even more into account regarding both the social and technical aspects of this problem to ultimately determine the acceptance of this technology by the general public.[3-8]

7.3 Pervasive Display Systems (PDS)

A PDS is made up of public displays, i.e. displays that are not under the control of a single user. This is probably the main conceptual difference between PDS and Distributed Displays Environments (DDE) research, where the focus is on supporting computer systems that present output to more than one physical display.

A PDS is multi-purpose, and thus is not tailored at the needs of any application in particular. It provides generic support to an open-ended set of applications that may require display services. The main function of the PDS is to manage display requests and arbitrate display resources. A direct consequence of being multi-purpose is the need to provide some control interface where the behaviour of the system can be programmed.

Finally, a PDS is assumed to support coordination between multiple displays. Even if individual displays may control their own level of integration with a shared infrastructure, and in some cases be operated in isolation, the assumption is that the system is able to support the integration of multiple displays into an infra-structure that is perceived by users as a single coordinated display system. Furthermore, those multiple displays are potentially dispersed across many sites spanning a vast geographical area and they are typically placed at such a distance from each other that a user is only able to interact with one display at each point in time. Again, this differentiates PDS from Distributed Display Environments where displays are assumed to be co-located. [9]

7.4 A Taxonomy of Pervasive Display Systems Usage Models

The central issue when considering multi-purpose display systems is how to identify the potential requirements that an open-ended set of applications may impose on those systems. This may prove to be an unfeasible task given the broad variety of scenarios in which pervasive displays systems have been used, and, since this is an area in which we are just starting to explore its many possibilities, all the new usage types that are yet to be identified.
Still, as an incremental step, we can identify at large what are the main usage models and their key requirements. Such categorisation of the main usage models from the perspective of the requirements imposed upon the infrastructure can help us clustering the multiple requirements posed by individual applications, and inform the design of pervasive display systems so that they can be targeted at their main usage. Furthermore, it may also help to guide user expectations on display systems and manage their perceived affordances. From our literature survey, we have grouped existing applications into the following five usage models:

- **Experience oriented** - Experience oriented displays are typically part of a media and sensor rich installation conceived for providing some sort of strong and engaging user experience, e.g. interactive art installations or games. The displays are normally interactive, but all the interactions are directly related with the narrative of the experience itself. A very important point for these systems is the ability of the experience creators to maintain a strong control over the behaviour of the displays so that they can know exactly what the experience is going to be like.
- **Content oriented** - Content oriented displays are focused on the dissemination of content to people passing-by. Content can be controlled centrally in a one-to-many broadcast model, as in the digital signage networks that control most of the displays that we can find today in public places, or created locally, as in the plasma Poster network, in which community shared multimedia content is displayed. In either case, the main role of the system is to arbitrate content presentation by separating it in time or space.

- **Sign oriented** - Sign oriented displays are primarily used as digital replacements for traditional signs, e.g. room numbers or directional signs. Their objective is thus to present specific and short pieces of information concerning things such as directions to some nearby event, the name of the place, the current status of a meeting room, or an occasional location-based announcement. Even if we admit that a sign oriented display may alternate its presentation between a small set of different views, they are not expected to change as often as content oriented systems. An appropriate space model may enable displays to generate dynamic directions that are location and orientation specific as in.
- **Ambient oriented** - Ambient oriented displays are mainly used for promoting awareness through the periphery of attention. They are expected to change smoothly, providing a continuous, but distraction-free, output of background information. Examples may include GUI-based approaches such as informative art and Info Canvas, or displays based on light and sound patterns.
- **Personal oriented** - Personal oriented displays are mainly designed to support individual access to digital services. An individual may approach the display and use it at its own convenience for the time needed to complete a particular service. They differ from typical information kiosks in that the display is public, and thus visible to other proximate people. If the system is able to support multiple users simultaneously interactions may be longer, otherwise interactions should be short in order to avoid pre-emption of the display by a single user. Another major concern is handling user data which may be private and thus not shown on the display. Examples of multi-user systems designed for personal services are the Dynamo and Blue Board systems. More common examples include the store assistance displays in which customers may approach the barcode of a product and obtain further information about that product. [9]

7.5 Touch Screens Application Tips

Ten simple pointers that can make your touch-enabled application a success.

1. Run your application full screen.

Remove title bars and menu bars so your application can take full advantage of the entire display area.

2. Use bright background colours (not black).

Bright backgrounds in your application will hide fingerprints and reduce glare. Dithering or other patterned backgrounds (for example, the "crumpled paper look") help the eye focus on the screen image instead of reflections, even in areas where there are no icons or menu choices.

3. Use a simple point-and-click interface with large buttons.

Dragging, double-clicks, scroll bars, drop-down menus, multiple windows, or other elements can confuse the typical user and detract from user-friendliness and efficiency.

4. Turn the cursor off so your user will focus on the entire screen instead of the arrow.

A cursor on the screen makes the user think, "How do I get the arrow to do what I want?" Remove the cursor, and the user's thinking and actions become direct instead of indirect—thereby unlocking the true power of touch screens.

5. Always give your users feedback as soon as they touch the screen.

Immediate feedback is critical to reassure the user that a touch has registered. Responses can be visual, such as 3-D button effects similar to those found on a standard Windows button. Or you can provide an audio response, such as a "click" or other sound output whenever a user touches the screen.

6. Make your application fun and fast.

Users will walk away from a sluggish system. You can keep their attention with a quick response to touches. Speedy systems also reduce vandalism. Avoid graphics modes offering excessive colours or high resolution—these will only slow down your system.

7. Make the application intuitive, limit choices, and guide the user as much as possible.

Test your application with users. If they pause in confusion—even for a moment—you've identified the areas than need improvement.

8. Digitized speech can talk users through your application.

Because the human brain can simultaneously process a voice and absorb an image, there is something almost magical about a user interface that provides voice prompts and touch response. The better kiosk applications exploit this knowledge for maximum effect. For example: "Touch the first letter of the company you are looking for." Click. "Now touch OK." Click.

9. Make your application part of an attractive package.

Animation and large fonts help attract users to kiosk applications. The actual design of the kiosk cabinet should also be attractive and sturdy.

10. Keep the following in mind when designing a kiosk cabinet.

Are you using forced-air ventilation? Put your fan at the top, near the monitor's vents. To minimize the airborne dust from footsteps, keep the intake away from the floor. Keep air from entering around the monitor face. Point your speakers in the direction of your user's ears. Allow for variations in the physical dimensions of monitor models, as they change frequently. The display should also be mounted securely or have a steady base so it feels solid to the touch. Finally, choose a finish that does not show fingerprints—avoid polished stainless steel, chrome, or glossy black paint. [27]

8 Peer to Peer Computing

The term “peer-to-peer” (P2P) refers to a class of systems and applications that employ distributed resources to perform a critical function in a decentralized manner. The critical function can be distributed computing, data/content sharing, communication and collaboration, or platform services. Decentralization may apply to algorithms, data, and meta-data, or to all of them. This does not preclude retaining centralization in some parts of the systems and applications if it meets their requirements. Typical P2P systems reside on the edge of the Internet or in ad-hoc networks.

With the pervasive deployment of computers, P2P is increasingly receiving attention in research, product development, and investment circles. This interest ranges from enthusiasm, through hype, to disbelief in its potential. Some of the benefits of a P2P approach include: improving scalability by avoiding dependency on centralized points; eliminating the need for costly infrastructure by enabling direct communication among clients and enabling resource aggregation. [12]

8.1 Goals of Peer to Peer Computing

As with any computing system, the goal of P2P systems is to support applications that satisfy the needs of users. Selecting a P2P approach is often driven by one or more of the following goals.

- **Cost sharing/reduction**. Centralized systems that serve many clients typically bear the majority of the cost of the system. When that main cost becomes too large, a P2P architecture can help spread the cost over all the peers. For example, in the file-sharing space, the Napster system enabled the cost sharing of file storage, and was able to maintain the index required for sharing. In the end, the legal costs of maintaining the index became too large, and so more radical P2P systems such as Gnutella were able to share the costs even further. Much of the cost sharing is realized by the utilization and aggregation of otherwise unused resources (e.g. SETI@home), which results both in net marginal cost reductions and a lower cost for the *most* costly system component. Because peers tend to be autonomous, it is important for costs to be shared reasonably equitably.
- **Improved scalability/reliability**. With the lack of strong central authority for autonomous ears, improving system scalability and reliability is an important goal. As a result, algorithmic innovation in the area of resource discovery and search has been a clear area of research, resulting in new algorithms for existing systems, and the development of new P2P platforms (such as CAN, Chord, and PAST).
- **Resource aggregation and interoperability**. A decentralized approach lends itself naturally to aggregation of resources. Each node in the P2P system brings with it certain resources such as compute power or storage space. Applications that benefit from huge amounts of these resources, such as compute-intensive simulations or

distributed file systems, naturally lean toward a P2P structure to aggregate these resources to solve the larger problem. Distributed computing systems, such as SETI@Home, distributed.net, and Endeavours are obvious examples of this approach. By aggregating compute resources at thousands of nodes, they are able to perform computationally intensive functions. File sharing systems, such as Napster, Gnutella, and so forth, also aggregate resources. In these cases, it is both disk space to store the community's collection of data and bandwidth to move the data that is aggregated. Interoperability is also an important requirement for the aggregation of diverse resources.

- **Increased autonomy**. In many cases, users of a distributed system are unwilling to rely on any centralized service provider. Instead, they prefer that all data and work on their behalf be performed locally. P2P systems support this level of autonomy simply because they require that the local node do work on behalf of its user. The principle example of this is the various file sharing systems such as Napster, Gnutella, and FreeNet. In each case, users are able to get files that would not be available at any central server because of licensing restrictions. However, individuals autonomously running their own servers have been able to share the files because they are more difficult to find than a server operator would be.
- **Anonymity/privacy**. Related to autonomy is the notion of anonymity and privacy. A user may not want anyone or any service provider to know about his or her involvement in the system. With a central server, it is difficult to ensure anonymity because the server will typically be able to identify the client, at least by Internet address. By employing a P2P structure in which activities are performed locally, users can avoid having to provide any information about themselves to anyone else. FreeNet is a prime example of how anonymity can be built into a P2P application. It uses a forwarding scheme for messages to ensure that the original requestor of a service cannot be tracked. It increases anonymity by using probabilistic algorithms so that origins cannot be easily tracked by analyzing network traffic.
- **Dynamism**. P2P systems assume that the computing environment is highly dynamic. That is, resources, such as compute nodes, will be entering and leaving the system continuously. When an application is intended to support a highly dynamic environment, the P2P approach is a natural fit. In communication applications, such as Instant Messaging, so-called "buddy lists" are used to inform users when persons with whom they wish to communicate become available. Without this support, users would be required to "poll" for chat partners by sending periodic messages to them. Likewise, distributed computing applications such as distributed.net and SETI@Home must adapt to changing participants. They, therefore, must re-issue computation jobs to other participants to ensure that work is not lost if earlier participants drop out of the network while they were performing a computation step.

- **Enabling ad-hoc communication and collaboration**. Related to dynamism is the notion of supporting ad-hoc environments. By ad hoc, we mean environments where members come and go based perhaps on their current physical location or their current interests. Again, P2P fits these applications because it naturally takes into account changes in the group of participants. P2P systems typically do not rely on established infrastructure — they build their own, e.g., logical overlay in CAN and PAST. [12]

8.2 Peer to Peer Taxonomies

All computer systems can be classified into centralized and distributed. Distributed systems can be further classified into the client-server model and the P2P model. The client-server model can be flat where all clients only communicate with a single server (possibly replicated for improved reliability), or it can be hierarchical for improved scalability. In a hierarchal model, the servers of one level are acting as clients to higher level servers. Examples of a flat model include traditional middleware solutions, such as object request brokers and distributed objects. Examples of a hierarchical model include DNS server and mounted file systems. The P2P model can either be *pure* or it can be *hybrid*. In a pure model, a centralized server does not exist. [12]

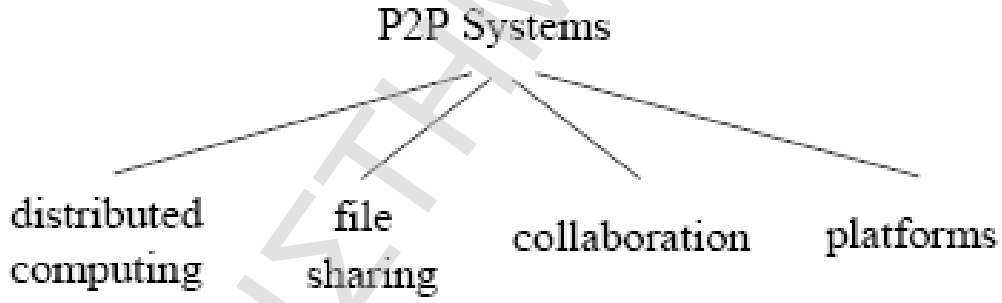


Figure 8-1 : *Taxonomy of P2P systems*

Figure 8-2 : *Hybrid Peer-to-Peer Model. (1) Initial communication with a server, e.g., to obtain the location/identity of a peer, followed by (2) direct communication with a peer.*

9 Tools and Technologies

Tools and technologies that were used for the implementation are:

- For the database design : Visual Database Designer of Visual Studio 2005
- For the database Implementation : Microsoft SQL Server (Express)
- For the implementation of the system's applications (Windows forms) : Visual Studio 2005, Visual C#
- Web Server : Internet Information Services (IIS) version 5.1
- Operational system : Windows XP, Service Pack2 with .Net framework 2.0
- For system architecture design : Microsoft Visio 2003
- For content creation and editing : Adobe Photoshop CS

Further details on the implementation's concept and enabling technologies are described in Chapter 12.

10 Requirements Analysis

Requirements will be examined from the aspect of each application:

- Network Manager
- Player
- Calendar
- P2P Server
- Schedule Server

10.1 Requirements from the aspect of Network Manager

- Network Manager can report “activities”. “Activities” represents actions that refer in the whole distributive system. They are categorized in three categories according to their derivation. Time of the activity is always recorded. Activities messages includes :
  - From Network Manager
    - Network Manager added PlayerX
    - Network Manager removed PlayerX
    - Network Manager set as current content update ContentUpdateX of PlayerX softwareversionX
    - Network Manager added contentupdateX of PlayerX versionX
    - Network Manager deleted ContentupdateX of PlayerX versionX
    - Network Manager added VersionX of PlayerX
    - Network Manager triggered a content update on PlayerX
    - Network Manager triggered software version update on PlayerX
  - From Players
    - PlayerX started
    - PlayerX stopped
    - PlayerX have been updated (content update)
    - PlayerX have been updated (software version update)
  - From Calendar
    - A new schedule have been added (PlayerX, ContentupdateX, TIME)
- Network Manager can create activity messages (the messages are listed above)
- Network Manager can present Players and its details
- Network Manager can present Players’ current software version and its details
- Network Manager can present Players’ available content updates (of the current software version) and its details
- Network Manager can add a new Player
- Network Manager can delete an existing Player

- Network Manager can add a new software version on an existing Player. This task includes several steps as the generation of the default content update and the determination (registration) of its available controls.
- Network Manager can add a new content update. The new content update will be a replication of the default content update. After the creation of the new content update user can edit the controls.
- Network Manager can present all the available content updates, its details and the available controls. The editor panel allows controls' editing. Each control type has different editable properties. The properties of each control type are listed below:
    - Forms (This control has properties that represent the properties of the main form player's application)
        - BackColor : Background color of the form
        - BackgroundImageUrl : Background image url of the form
        - BackgroundImageLayout : Background image layout of the form
    - Labels
        - Text : Text of the label
        - BackColor : Background color of the label
        - BorderStyle : Border style of the label
        - Enabled : Enable status of the label
        - Font : Font of the label (includes font name, font size, etc)
        - ForeColor : Foreground color of the label
        - Size : Size of the label
        - TextAlign : Text align of the label
        - Visible : Visible status of the label
        - AutoSize : Autosize status of the label
        - Location : Location of the label
    - Textboxes
        - Text : Text of the textbox
        - BackColor : Background color of the textbox
        - BorderStyle : Border style of the textbox
        - Enabled : Enable status of the textbox
        - Font : Font of the textbox (includes font name, font size, etc)
        - ForeColor : Foreground color of the textbox
        - Size : Size of the textbox
        - TextAlign : Text align of the textbox
        - RedaOnly : Read only status of the textbox
        - Visible : Visible status of the textbox
        - MultiLine : Multiline status of the textbox
        - ScrollBars : Scrollbars of the textbox
        - Location : Location of the textbox

- Buttons
  - Text : Text of the button
  - BackColor : Background color of the button
  - Enabled : Enable status of the button
  - Font : Font of the button (includes font name, font size, etc)
  - ForeColor : Foreground color of the button
  - Size : Size of the button
  - TextAlign : Text align of the button
  - FlatStyle : Flat style of the button
  - AutoSize : Autosize status of the button
  - Visible : Visible status of the button
  - Location : Location of the button
- Pictureboxes
  - BackColor : Background color of the picturebox
  - BorderStyle : Border style of the picturebox
  - Enabled : Enable status of the picturebox
  - Size : Size of the picturebox
  - SizeMode : Sizemode of the picturebox
  - Visible : Visible status of the picturebox
  - Location : Location of the picturebox
- Webbrowser
  - Url : Url of the web browser
  - AllowNavigation : Allow navigation status of the web browser
  - ScrollBarsEnabled : Scroll bars enabled status of the web browser
  - Visible : Visible status of the web browser
  - Size : Size of the web browser
  - Location : Location of the web browser
- Videoplayers
  - URL : Url of the media file of the video player
  - Size : Size of the video player
  - Visible : Visible status of the video player
  - UiMode : Uimode of the video player (specifies if the videoplayers buttons are visible)
  - StretchToFit : Stretch to fit status of the video player
  - Location : Location of the video player

Ø Network Manager can manage content updates. Available controls can be previewed in the preview panel. An available content update can be selected to become the current one.

Ø Network Manager can delete a content update

- Network Manager can connect with P2P Server and ask for a list of the online connected Players (Connected Players are active). Network Manager can select an active Player and triggers a content update or a software version update

10.2 Requirements from the aspect of the player

- Player can be published and then installed in the remote location by three ways. Installation should be triggered from the remote location manually. (An alternative way can be to install Player's application remotely by using remote desktop application)
    1. publish Player to the web
    2. publish Player to a file share
    3. publish Player to a cd-rom or dvd-rom
- Player start up should be made from the remote location manually (An alternative way can be to start Player's application remotely by using remote desktop application)
- When the Player starts should try to connect to P2P Server. If this is not possible Player will try to connect to P2P Server till connection will be established. P2P Server connection ensures that the player will be updated when Network Manager or Schedule Server triggers an update
- Player can be updated on demand remotely
- Player should create and inform Network Manager with activity messages
- Player should interact with users. Interaction will be performed without keyboard. Touch screen is the input devise for the Player
- Player should disconnect with P2P Server on closing
- Player can offer services according to the case that is being developed

10.3 Requirements from the aspect of Calendar

- The user of calendar application can select an available Player and schedule an update of content update.
- The user of calendar application can select a date or a period and see all the scheduled updates, which ones are active and the details of each one
- Calendar should create activity messages and inform Network Manager when a scheduled update is added.

10.4 Requirements from the aspect of Schedule Server

- Ø When Schedule Server application loads should try to connect to P2P Server. If the connection is not possible, it will try to establish connection every 10 seconds. When the connection is established the Schedule Server can be started.
- Ø Schedule Server will check for scheduled updates every 10 seconds.
- Ø If Schedule Server locate an scheduled update will trigger the update
- Ø In case where Connection with P2P Server is lost, Schedule Server will track it and stop running till it reconnects.
- Ø Messages that are related with all these states informs Schedule Server's user
- Ø Schedule Server should be disconnected with the P2P Server on closing (automatically)

10.5 Requirements from the aspect of P2P Server

- Ø P2P Server can understand if an application (peer) is connected
- Ø P2P Server can send a list of the connected peers if it is requested
- Ø P2P Server can broadcast a message of a peer to the other peers

11 Database Design

Visual Database Designer of Visual studio 2005 is the tool that has been used for the database design and database implementation.

11.1 Database relationship diagram

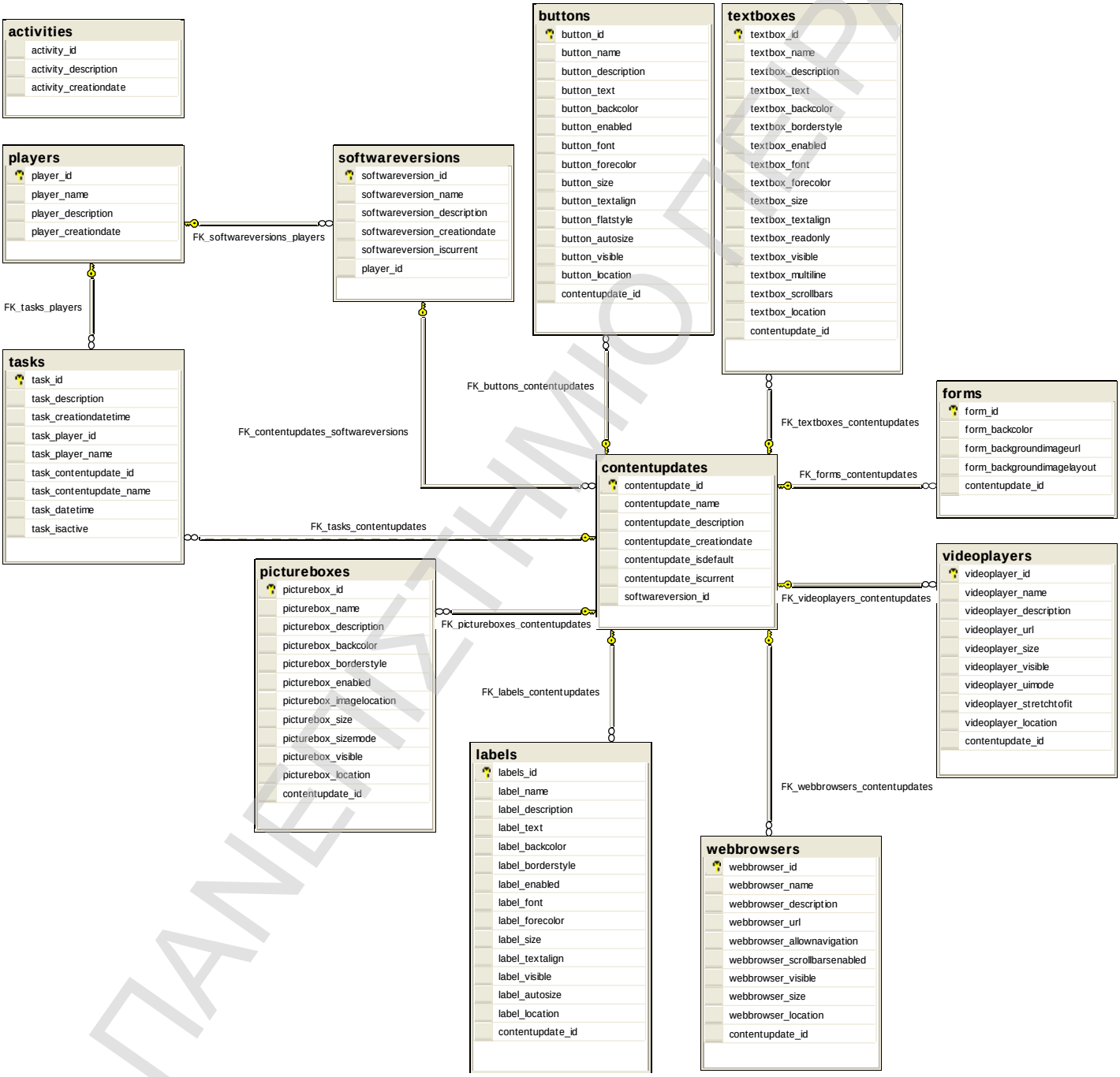


Figure 11-1 : *Database relationship diagram*

11.2 Database table definition

The tables that have been created are:

- players
- softwareversions
- contentupdates
- forms
- labels
- textboxes
- buttons
- pictureboxes
- webbrowsers
- videoplayers
- tasks
- activities

Detailed description is presented below :

Column Name	: The name of the column
Condensed Type	: The condensed type of the column
Allow Nulls	: Specifies if null values are allowed
Identity	: Specifies if the value is auto increment one
Description	: Description of the column
Key image	: Primary Key (PK)
Foreign Keys (FK) are not presented in the tables but are referenced in the description of the table	

11.2.1 Table “players”

Table “Players” stores data of the Players

players

Column Name	Condensed Type	Allow Nulls	Identity	Description
player_id	int	☐	☑	id of the player
player_name	varchar(50)	☐	☐	name of the player
player_description	varchar(MAX)	☐	☐	description of the player
player_creationdate	datetime	☐	☐	creation datetime of the player
		☐	☐	

Figure 11-2: *Table "players"*

11.2.2 Table “softwareversions”

Softwareversions : Stores data of Player’s software versions (FK : player_id)

softwareversions

Column Name	Condensed Type	Allow Nulls	Identity	Description
softwareversion_id	int	☐	☑	id of the software version
softwareversion_name	varchar(50)	☐	☐	name of the software version
softwareversion_description	varchar(MAX)	☐	☐	description of the software version
softwareversion_creationdate	datetime	☐	☐	creation datetime of the software version
softwareversion_iscurrent	bit	☐	☐	flag that shows if the software version is the current one
player_id	int	☐	☐	player's id
		☐	☐	

Figure 11-3 : *Table "softwareversions"*

11.2.3 Table “contentsupdates”

Contentupdates : Stores data of software versions’ content updates (FK : softwareversion_id)

contentupdates

Column Name	Condensed Type	Allow Nulls	Identity	Description
contentupdate_id	int	☐	☑	id of the content update
contentupdate_name	varchar(50)	☐	☐	name of the content update
contentupdate_description	varchar(MAX)	☐	☐	description of the content update
contentupdate_creationdate	datetime	☐	☐	creation date time of the content update
contentupdate_isdefault	bit	☐	☐	flag that show if the content update is the default one
contentupdate_iscurrent	bit	☐	☐	flag that shows if the content update is the current one
softwareversion_id	int	☐	☐	id of the software version that the content update belongs to
		☐	☐	

Figure 11-4 : *Table "contentupdates"*

11.2.4 Table “forms”

Forms : Stores data of the main form of the content update (FK : contentupdate_id)

forms

Column Name	Condensed Type	Allow Nulls	Identity	Description
form_id	int	☐	☑	id of the main form
form_backcolor	varchar(MAX)	☐	☐	background color of the main form
form_backgroundimageurl	varchar(MAX)	☑	☐	background image url of the main form
form_backgroundimagelayout	varchar(MAX)	☑	☐	background image layout of the main form
contentupdate_id	int	☐	☐	id of the content update that the main form belongs to
		☐	☐	

Figure 11-5 : *Table "forms"*

11.2.5 Table “labels”

Labels : Stores data of the labels of the content update (FK : contentupdates_id)

labels

Column Name	Condensed Type	Allow Nulls	Identity	Description
labels_id	int	☐	☑	id of the label
label_name	varchar(50)	☐	☐	name of the label
label_description	varchar(MAX)	☐	☐	description of the label
label_text	varchar(MAX)	☐	☐	text of the label
label_backcolor	varchar(MAX)	☐	☐	bachground color of the label
label_borderstyle	varchar(MAX)	☐	☐	border style of the label
label_enabled	varchar(MAX)	☐	☐	enable status of the label
label_font	varchar(MAX)	☐	☐	font of the label
label_forecolor	varchar(MAX)	☐	☐	foreground color of the label
label_size	varchar(MAX)	☐	☐	size of the label
label_textalign	varchar(MAX)	☐	☐	text align of the label
label_visible	varchar(MAX)	☐	☐	visible status of the label
label_autosize	varchar(MAX)	☐	☐	autosize status of the label
label_location	varchar(MAX)	☐	☐	location of the label
contentupdate_id	int	☐	☐	id of the content update that the label belongs to
		☐	☐	

Figure 11-6 : Table "labels"

11.2.6 Table “textboxes”

Textboxes : Stores data of the textboxes of the content update (FK : contentupdate_id)

textboxes

Column Name	Condensed Type	Allow Nulls	Identity	Description
textbox_id	int	☐	☑	id of the textbox
textbox_name	varchar(50)	☐	☐	name of the textbox
textbox_description	varchar(MAX)	☐	☐	description of the textbox
textbox_text	varchar(MAX)	☐	☐	text of the textbox
textbox_backcolor	varchar(MAX)	☐	☐	background color of the textbox
textbox_borderstyle	varchar(MAX)	☐	☐	border style of the textbox
textbox_enabled	varchar(MAX)	☐	☐	enabled status of the textbox
textbox_font	varchar(MAX)	☐	☐	font of the textbox
textbox_forecolor	varchar(MAX)	☐	☐	foreground color of the textbox
textbox_size	varchar(MAX)	☐	☐	size of the textbox
textbox_textalign	varchar(MAX)	☐	☐	textalign of the textbox
textbox_readonly	varchar(MAX)	☐	☐	read only status of the textbox
textbox_visible	varchar(MAX)	☐	☐	visible of the textbox
textbox_multiline	varchar(MAX)	☐	☐	multiline status of the textbox
textbox_scrollbars	varchar(MAX)	☐	☐	scrollbars of the textbox
textbox_location	varchar(MAX)	☐	☐	location of the textbox
contentupdate_id	int	☐	☐	id of the content update that the textbox belongs to
		☐	☐	

Figure 11-7 : *Table "textboxes"*

11.2.7 Table “buttons”

Buttons : Stores data of the buttons of the content update (FK : contentupdate_id)

buttons

Column Name	Condensed Type	Allow Nulls	Identity	Description
button_id	int	☐	☑	id of the button
button_name	varchar(50)	☐	☐	name of the button
button_description	varchar(MAX)	☐	☐	description of the button
button_text	varchar(MAX)	☐	☐	text of the button
button_backcolor	varchar(MAX)	☐	☐	background color of the button
button_enabled	varchar(MAX)	☐	☐	enabled status of the button
button_font	varchar(MAX)	☐	☐	font of the button
button_forecolor	varchar(MAX)	☐	☐	foreground color of the button
button_size	varchar(MAX)	☐	☐	size of the button
button_textalign	varchar(MAX)	☐	☐	text align of the button
button_flatstyle	varchar(MAX)	☐	☐	flat style of the button
button_autosize	varchar(MAX)	☐	☐	autosize status of the button
button_visible	varchar(MAX)	☐	☐	visible status of the button
button_location	varchar(MAX)	☐	☐	location of the button
contentupdate_id	int	☐	☐	id of the content update that the button belongs to
		☐	☐	

Figure 11-8 : *Table "buttons"*

11.2.8 Table “pictureboxes”

Pictureboxes : Stores data of the pictureboxes of the content update (FK : contentupdate_id)

pictureboxes

Column Name	Condensed Type	Allow Nulls	Identity	Description
picturebox_id	int	☐	☑	id of the picturebox
picturebox_name	varchar(50)	☐	☐	name of the picturebox
picturebox_description	varchar(MAX)	☐	☐	description of the picturebox
picturebox_backcolor	varchar(MAX)	☐	☐	background color of the picturebox
picturebox_borderstyle	varchar(MAX)	☐	☐	border style of the picturebox
picturebox_enabled	varchar(MAX)	☐	☐	enabled status of the picturebox
picturebox_imagelocation	varchar(MAX)	☐	☐	image location of the picturebox
picturebox_size	varchar(MAX)	☐	☐	size of the picturebox
picturebox_sizemode	varchar(MAX)	☐	☐	size mode of the picturebox
picturebox_visible	varchar(MAX)	☐	☐	visible status of the picturebox
picturebox_location	varchar(MAX)	☐	☐	location of the picturebox
contentupdate_id	int	☐	☐	id of the content update that the picturebox belongs to
		☐	☐	

Figure 11-9 : *Table "pictureboxes"*

11.2.9 Table “webbrowsers”

Webbrowsers : Stores data of the web browsers of the content update (FK : contentupdate_id)

webbrowsers

Column Name	Condensed Type	Allow Nulls	Identity	Description
webbrowser_id	int	☐	☑	id of the web browser
webbrowser_name	varchar(50)	☐	☐	name of the web browser
webbrowser_description	varchar(MAX)	☐	☐	description of the web browser
webbrowser_url	varchar(MAX)	☐	☐	url of the web browser
webbrowser_allownavigation	varchar(MAX)	☐	☐	allow navigation of the web browser
webbrowser_scrollbarsenabled	varchar(MAX)	☐	☐	scrollbars enabled of the web browser
webbrowser_visible	varchar(MAX)	☐	☐	visible status of the web browser
webbrowser_size	varchar(MAX)	☐	☐	size of the web browser
webbrowser_location	varchar(MAX)	☐	☐	location of the web browser
contentupdate_id	int	☐	☐	id of the content update that the web browser belongs to
		☐	☐	

Figure 11-10 : *Table "webbrowser"*

11.2.10 Table “videoplayers”

Videoplayers : Stores data of the video players of the content update (FK : contentupdate_id)

videoplayers

Column Name	Condensed Type	Allow Nulls	Identity	Description
videoplayer_id	int	☐	☑	id of the video player
videoplayer_name	varchar(50)	☐	☐	name of the video player
videoplayer_description	varchar(MAX)	☐	☐	description of the video player
videoplayer_url	varchar(MAX)	☐	☐	url of the video player
videoplayer_size	varchar(MAX)	☐	☐	size of the video player
videoplayer_visible	varchar(MAX)	☐	☐	visible of the video player
videoplayer_uimode	varchar(MAX)	☐	☐	uimode of the video player
videoplayer_stretchtofit	varchar(MAX)	☐	☐	stretch to fit of the video player
videoplayer_location	varchar(MAX)	☐	☐	location of the video player
contentupdate_id	int	☐	☐	id of the content update that the video player belongs to
		☐	☐	

Figure 11-11 : *Table "videoplayers"*

11.2.11 Table “tasks”

Tasks : Stores data of the tasks. Tasks are managed from the calendar and the schedule server (FK : task_player_id)

tasks

Column Name	Condensed Type	Allow Nulls	Identity	Description
task_id	int	☐	☑	id of the task
task_description	varchar(MAX)	☐	☐	description of the task
task_creationdatetime	datetime	☐	☐	creation datetime of the task
task_player_id	int	☐	☐	player id of the task
task_player_name	varchar(MAX)	☐	☐	player name of the task
task_contentupdate_id	int	☐	☐	content update id of the task
task_contentupdate_name	varchar(MAX)	☐	☐	content update name of the task
task_datetime	datetime	☐	☐	date time of the task
task_isactive	bit	☐	☐	flag that shows if the task is active
		☐	☐	

Figure 11-12 : *Table "tasks"*

11.2.12 Table “activities”

Activities : Stores data according to the use of the whole platform

activities

Column Name	Condensed Type	Allow Nulls	Identity	Description
activity_id	int	☐	☑	id of the activity
activity_description	varchar(MAX)	☐	☐	description of the activity
activity_creationdate	datetime	☐	☐	creattion datetime of the activity
		☐	☐	

Figure 11-13 : *Table "activities"*

12 Implementation

12.1 System Architecture

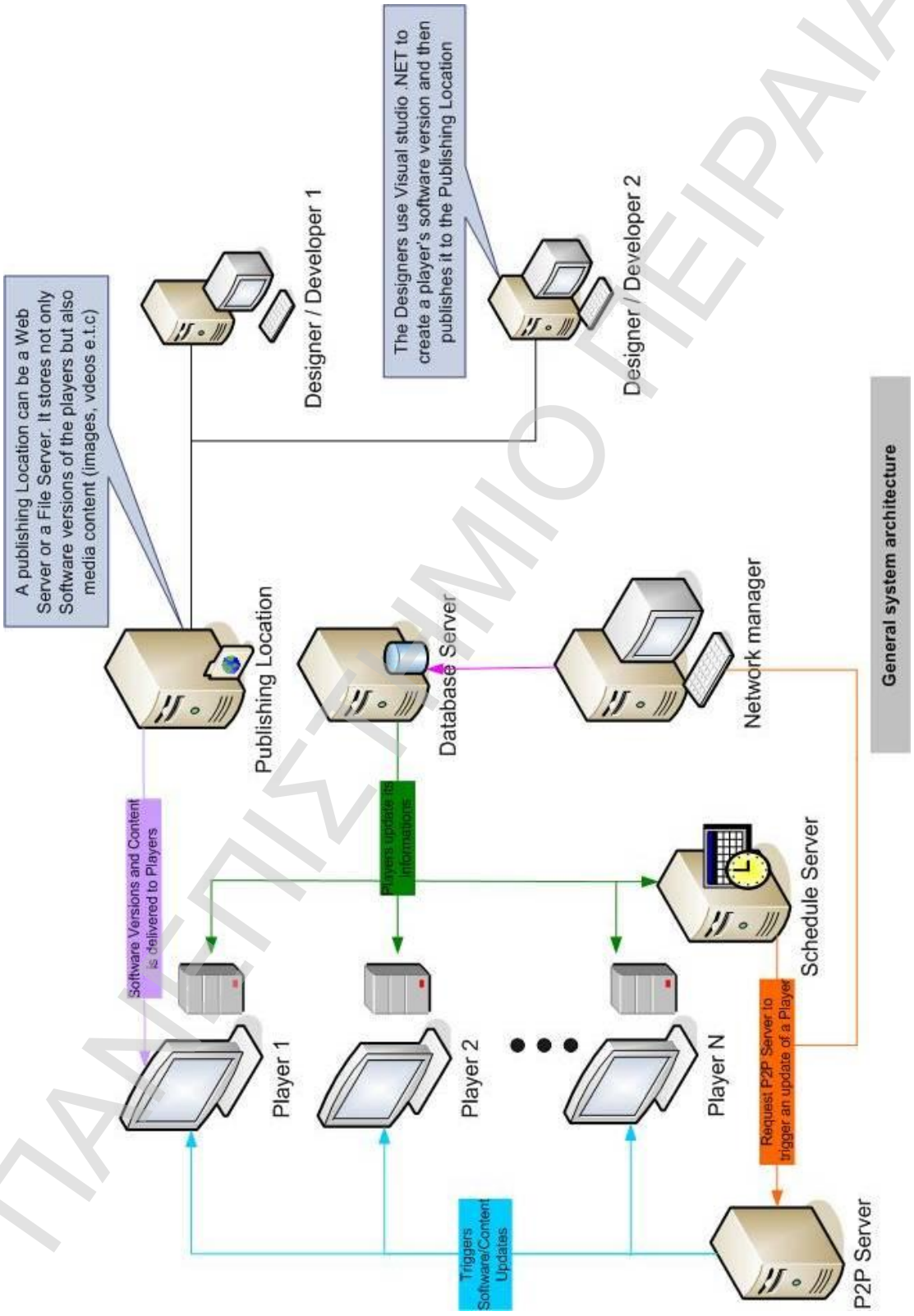


Figure 12-1 : *System Architecture*

The components of the systems are:

1. **Database Server** (CentralDB.mdf, CentralDB_log.ldf.): All applications (Network Manager, Players, Schedule Server, Calendar) have remote access to Database Server (Chapter 11).
2. **Network Manager** (Windows Form Application: Network_Manager.exe): Network Manager manages Players, Content Updates and Software Updates. Activity Monitoring System gives an easy way to track of the status of all the Players and monitor previous actions. It is the first line of defense against unscheduled Player outages and network malfunctions. Any changes of Players, its Software Versions or its Content Updates are stored in the central database. Players will recognize the changes only on start or if P2P Server triggers them to do so. Network Manager can make a real time request to P2P Server to trigger a Player that is active (on-line). A wizard will allow user of Network Manager to add a new Software Version. During this task he will be asked to add (register) the necessary editable controls of the software version and the default Content Update will be automatically generated. Afterwards he can add new Content Updates and edit their control with an easy to use editor panel. A preview Panel is also available so he can preview the controls of Content Updates and decide which one should be the current. A number of other facilities are available and messages informs user if an action was committed successfully or not.(Chapter 10.1)
3. **Publishing Location** (It can be one or more): Publishing Location could be a Web Server, a File Server or a File Share. For the need of the implementation we create a Web server. When the designer creates a software version of a Player publish it to this Web Server. The software version can be installed or updated to the remote machine of the Player through World Wide Web. Furthermore we stored all media files (images and video files) in this Web Server. The user of Network Manager should upload all files to the appropriate folder of the server and ensure that he uses the right Urls.
4. **Designer/Developer**: Designer is the specialist who is responsible for the implementation of the Software Version of a Player. He is obligated to use Visual Studio .NET 2005 to implement the application. ClickOnce technology (Detailed description will be given in the next chapter 12.2) is available only in this software, since we have not implemented this facility in this study. When the application is ready, designer publishes it to the Publishing Location. Then Network Manager has to take action and add this software version.
5. **Calendar** (Windows Form Application: Calendar.exe): Calendar is a tool that allows user to a schedule updates of Players. User can select an available Player, a Content Update and pick date and time for the update. Update details are stored in database and Schedule Server will take over to commit the update. Scheduled tasks are presented to the user as active or inactive if has already committed.

6. **Schedule Server** (Windows Form Application: Schedule_Server.exe): Schedule Server checks if there is a request for a scheduled update to commit it. Schedule Server could have been implemented as a Windows Service. We preferred Windows Form Application option so that we can track changes and events by using the activity panel of the application. Schedule Server recognizes automatically if P2P Server is active. Messages inform user if something fails. A failure on P2P Server will be distinguished and faced. The application will be stopped when P2P Server is down and will prompt user for restart when P2P Server is back again. Two treads are running. The first one checks the state of P2P Server and the second checks database for scheduled updates. Checkpoint time circle is hard-coded to 10 seconds.
7. **P2P Server** (Windows Form Application: P2P_Server.exe): P2P Server broadcasts messages (requests) to peers (Players). P2P Server could have been implemented as a Windows Service. We preferred Windows Form Application option so that we can track changes and events by using the activity panel of the application.
8. **Player** (Windows Form Application: PlayerX.exe): Player is the application on the remote machine that presents content to users. Players' applications have been designed as smart clients (see Chapter 12.2). Players can be one to many. There isn't any upper limit that restricts the number of the overall Players. A player has Software Versions (1.0.0.1, 1.0.0.2, ..) and every Software Version has Content Updates. Content Updates can be managed from Network Manager (add, delete, edit, make current). There isn't any limitation on the designing of the application's user interface. Developer can implement Player as he desires but he should determine which of the controls (labels, textboxes, buttons, pictureboxes, web browsers, video players) of the application will be the editable ones. Components that have to do with Player's update operations should be implemented in every Player.

12.2 Smart Clients

The overall design of the system and particular the design of the Players was based on Smart clients. Smart client applications are a powerful alternative to thin client applications. They can provide users with a rich and responsive user interface, the ability to work offline, and a way to take advantage of local hardware and software resources. In addition, they can be designed to run on a broad spectrum of client devices, including desktop PCs, Tablet PCs, and handheld mobile devices such as Pocket PCs and Smart phones.

Smart clients give users access to information and remote services within a powerful and intuitive client environment, and are an effective solution for flexible user-oriented applications and for increasing user productivity and satisfaction. Smart client applications can be designed to combine the traditional benefits of a rich client application with the manageability benefits of a thin client application.

12.2.1 What is a Smart Client?

To fully understand how smart clients combine the benefits of rich clients and thin clients, it is useful to examine the history and underlying principles behind the rich and thin client application models, and review some of the advantages and disadvantages associated with each.

12.2.1.1 Rich Client Applications

In the mid-1990s, the number of rich client applications developed for the Microsoft® Windows® operating system increased dramatically. These clients were designed to take advantage of the local hardware resources and the features of the client operating system platform.

Despite the impressive functionality of many of these applications, they have limitations. Many of these applications are stand-alone and operate on the client computer, with little or no awareness of the environment in which they operate. This environment includes the other computers and any services on the network, as well as any other applications on the user's computer. Very often, integration between applications is limited to using the cut or copy and paste features provided by Windows to transfer small amounts of data between applications.

There are technologies to help increase the connectivity of rich client applications. For example, two-tier applications allow multiple users to access common data residing on the network, and DCOM allows applications to become more distributed. (With DCOM, logic and state are no longer tied to the client computer, and instead are encapsulated within objects that are then distributed across multiple computers.) However, connected applications are considerably more complex to develop. As the size and complexity of these distributed applications grows, any tight coupling between client applications and the services they consume becomes increasingly difficult to maintain.

While rich clients typically provide a high-quality, responsive user experience and have good developer and platform support, they are very difficult to deploy and maintain. As the complexity of the applications and the client platform increases, so do the difficulties associated with deploying the application to the client computer in a reliable and secure way. One application can easily break another application if an incompatible shared component or library is deployed, a phenomenon known as *application fragility*. New versions of the application are typically made available by redeploying the entire application, which can increase an application fragility problem.

12.2.1.2 Thin Client Applications

The Internet provides an alternative to the traditional rich client model that solves many of the problems associated with application deployment and maintenance. Thin client, browser-based applications are deployed and updated on a central Web server; therefore, they remove the need to explicitly deploy and manage any part of the application to the client computer.

This model allows companies to very efficiently expose their applications to a large and diverse external audience. Because thin clients have proven to be effective at solving some of the deployment and manageability problems, they are now used to provide access to many line-of-business (LOB) applications to users within an organization, as well as access to externally facing applications to customers and partners. This is despite the fact that the needs and expectations of these two types of users are often radically different.

Thin client applications have some disadvantages. The browser must have a network connection at all times. This means that mobile users have no access to applications if they are disconnected, so they must re-enter data when they return to the office. Also, common application features such as drag-and-drop, undo-redo, and context-sensitive help may be unavailable, which can reduce the usability of the application.

Because the vast majority of the application logic and state lives on the server, thin clients make frequent requests back to the server for data and processing. The browser must wait for a response before the user can continue to use the application; therefore, the application will typically be much less responsive than an equivalent rich client application. This problem is exacerbated in low bandwidth or high latency conditions, and the resulting performance problems can lead to a significant reduction in application usability and user efficiency. An LOB application that requires heavy data entry and/or frequent navigation across multiple windows can be particularly affected by this problem.

12.2.1.3 Smart Client Applications

Smart client applications can be designed to combine the benefits of a rich client application with the deployment and manageability strengths of a thin client application, although the precise nature of the balance between the two approaches depends on the exact scenario.

Smart client applications often have very diverse requirements, and so vary greatly in design and implementation. However, all smart clients share some or all of the following characteristics:

- Make use of local resources
- Make use of network resources
- Support occasionally connected users
- Provide intelligent installation and update (ClickOnce)
- Provide client device flexibility

Many applications do not need all of these characteristics. As you design your smart clients, you will need to carefully consider your application scenario and decide which of these characteristics your smart client application requires. Incorporating all of these characteristics into your application will require very careful planning and design, and in many cases you will need significant implementation resources.

12.2.1.4 Choosing Between Smart Clients and Thin Clients

To choose the right application architecture for your situation, you must consider a number of factors. To determine whether a smart client approach is the most suitable for your application, carefully consider your current and future business application needs. If your application is based on an unsuitable architecture, it may fail to meet the requirements and expectations of the users and the business as a whole. Changing the architecture later to meet new requirements or to take advantage of new opportunities may be extremely expensive.

A thin client architecture is often the most appropriate if you need to make an externally facing application available to a diverse external audience, while a smart client architecture is often the most suitable for an internal application that needs to integrate with or coordinate other client-side applications or hardware, or that is required to work offline or provide specific high-performance functionality through a responsive user interface.

In reality these two approaches overlap to a great extent, and each has distinct advantages and disadvantages. You will only be able to choose the right approach after you carefully consider your requirements and understand how each approach would apply in your situation.

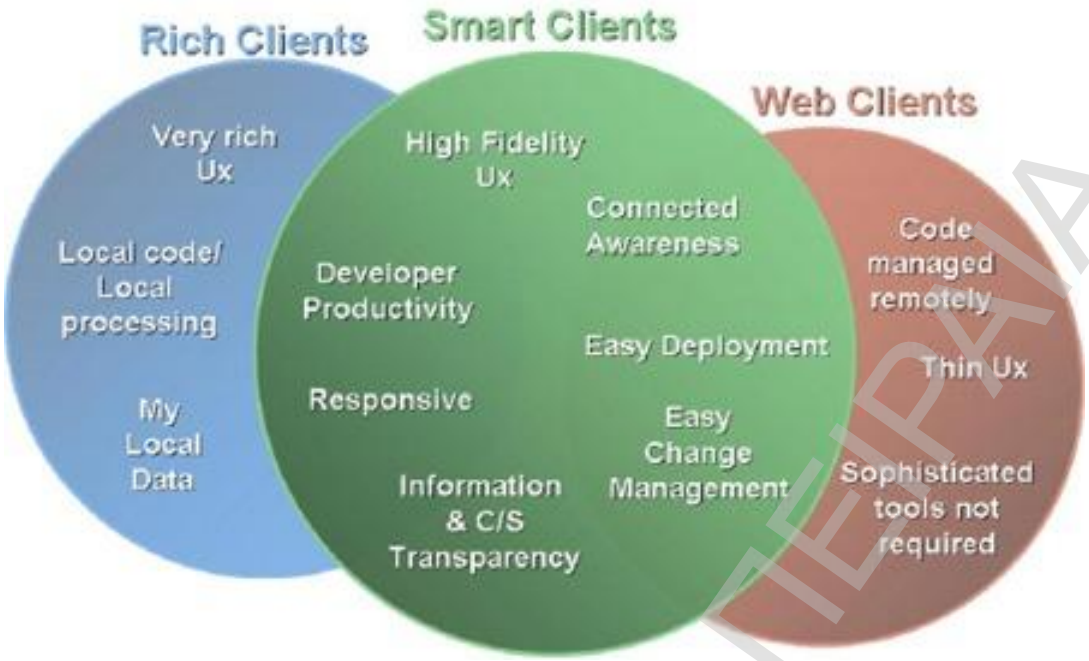


Figure 12-2 : *Smart Client definition diagram*

Feature	Thin client	Smart client
Provides a rich user interface	Yes, but difficult to develop, test, and debug. Generally ties the application to a single browser.	Yes. Easier to develop, test, and debug.
Can take advantage of hardware resources on local computer	Yes, but only through COM components.	Yes
Can interact with other local applications	No	Yes
Can be multithreaded	No	Yes
Can function offline	No	Yes
Can perform well in low bandwidth and high latency environments	No	Yes
Easy to deploy	Yes	Varies. Difficulty depends on application requirements.
Low maintenance and change management costs	Yes	Varies. Costs depend on application requirements.
Can be deployed to a wide variety of clients with varying capabilities	Yes, although more complex thin clients may require a single browser.	Yes. Can be deployed on any platform that supports the .NET Framework (including the .NET Compact Framework).

Figure 12-3 : *Features of Smart Clients and Thin Clients*

12.2.2 Deploying Smart Clients

12.2.2.1 Introduction

Web applications are limited in many ways, yet a large number of Web applications have been built over the last few years and more will continue to be developed going forward. Why do companies choose a Web-based solution over a rich client experience? There are a few good reasons, but the number one reason I hear is deployment.

When a company decides to create a new application for their employees, regardless of the type of system being designed, the discussion eventually moves to the issue of deployment. The system will need to be rolled out to the target users, and there needs to be a plan in place to handle ongoing updates (bug fixes, new feature releases, and so on). Years of experience with rolling out desktop applications have left most developers and IT staff with a good idea of the pain involved in client deployments, and a Web application ends up being the easier path. Of course, there are some tradeoffs in going with a browser-based Web application versus a rich-client application, but the fear of deployment usually makes those compromises acceptable. What we really need is a model for deploying client applications that is as easy and as safe as deploying a Web application, removing the need to compromise on the functionality of our applications. That is what "ClickOnce" brings to the table.

"ClickOnce" is a code name for a set of functionality in the next version of Microsoft® Visual Studio® .NET and the Microsoft® .NET Framework. It will allow us to create desktop applications that are deployed with a safe, system-controlled installation, and are automatically updated as needed from a central location.

12.2.2.2 Why Is Client Deployment So Hard?

ClickOnce exists to simplify client deployment, but what does that mean? What problems is ClickOnce going to enable us to avoid? Client deployment is hard because it happens on each of many clients, potentially at many locations, and your code and all of the associated components have to work on all of the target machines.

Consider a medium-sized application deployment of 10,000 seats (not all that unusual for a business application within a corporation). For a desktop application, that is 10,000 different machines that could each contain a slightly different mix of software configurations. In the case of a Web application, there may be anywhere from 1 server to a small "Web garden"" of machines—an extreme difference in scope for your deployment. So, in each case, you will need to ensure that your code will:

- Run on each target machine.
- Not break any existing software by being installed.
- Not be broken by the installation of any future software.

Quite a daunting task if you are dealing with 10,000 machines. What impact will it have on your testing to include all of the possible software combinations? The last two items on my list are affectionately known as "DLL Hell," where the installation or removal of a DLL for one application can break another. While there are many ways to reduce the likelihood of a conflict between applications, it is still a possibility when deploying onto client machines. All of these issues can make a Web application look very appealing. After all, when was the last time a Web application broke your desktop?

The act of installing your software the first time, or of installing any future updates, is also a challenge. For the Web server case, you could just handle the installation manually, assigning someone to go to each machine and run the setup program—a process that isn't feasible for 10,000 clients. When dealing with a large number of clients, you will likely need to rely on the user to take some action to run the initial installation, and to run a new install program for each update over the life of the application. While this approach has already been used successfully on countless occasions, it adds complexity to the release and to each update of the application.

Security is an issue for application installation as well. Depending on the policies of the company in question, a regular user may not have sufficient permission to run a traditional application installation.

12.2.2.3 Limitations of Browser-Based Applications

Web-based applications have their limits. There are some obvious issues with browser-based systems. The lack of offline support alone means that a Web-version of Microsoft® Outlook® isn't enough, and many issues with Web applications can be seen in even the most popular of sites.

Consider the act of clicking "checkout" or "submit" on an online shopping site once you've entered your credit card info. Right at that exact point on many occasions you may have received a browser error of "page not found" or "server is not responding." What does that mean for your order? Did it go through or not? This type of confusion is hard to avoid in a Web application—although it can be mitigated with e-mail confirmations and other methods—because it is not a mistake on the part of the developer. Rather, it is a side effect of the thin-client architecture of the Web.

Client applications can suffer from such issues as well, but in most cases these should be seen as a limitation or a bug; it can be done better. The developer of a client application has the option of saving your order locally and periodically retrying the transaction or querying for the state of the transaction after a loss of network connectivity—choices that are unavailable in a Web application. I'm not going to spend any more time discussing Web applications.

12.2.2.4 ClickOnce Provides the Best of Both Models

There are two main reasons to develop for the Web instead of for the client machine:

The first reason is a need to limit your application to the lowest common denominator (the Web browser) in an effort to reach almost any device that can access the network. Web applications aim for "reach" not "rich," supporting the widest number of clients at the cost of some functionality.

The second reason is the ease of installation and ongoing updates. The ability to apply a bug fix onto a single machine or a small set of machines, instead of requiring the new code to be applied on every single client, is an amazing time saver in the maintenance of an application.

Going for reach is critical when your goal is to ensure availability to almost any device that can connect to the Web, but if you are dealing with a slightly narrower target audience, such as "employees and partners of my company," then reach isn't as much of an issue. Once you've reduced your target audience to something a little bit less than everything and everyone, you can constrain the target platform (Windows machines capable of supporting the .NET Framework) and then take advantage of that platform by building a full desktop application. ClickOnce allows you to do this by providing the second part of the equation: easy installation and automatic updates for your applications.

The table below illustrates the features of each deployment model and shows how ClickOnce bridges the gap between the traditional Web and client worlds.

Features	Web	ClickOnce	MSI/Client
Reach	Y		
Auto-Deployment	Y	Y	
Low System Impact	Y	Y	
Install / Run Per-User	Y	Y	
Rich / Interactive Experience		Y	Y
Offline		Y	Y
Windows Shell Integration		Y	Y
Unrestricted Install			Y

Figure 12-4 : *Comparing Web, ClickOnce and MSI*

12.2.2.5 Existing Deployment Improvements Under .NET

Although ClickOnce is a new feature in the next version of the CLR and the .NET Framework, it is made possible by many existing aspects of managed code. Even in the initial version 1.0 release of .NET, managed applications were designed for application isolation and zero-impact deployment (often described as xcopy deployment), making it easier than ever to install and update your applications.

With the first release of .NET, applications could also be run directly from a Web site (http://mysite/myapp.exe) ensuring that they were always up-to-date. Nevertheless, the technical limitations of this approach were significant. Href-exes (as applications run from a HTTP address are known) often required security policy changes on the client (which meant that an .MSI had to be run before the application would work). They also ran slower than the

same application installed locally, the user experience was lacking (when a new assembly needed loading from the Web server, the UI would freeze for a few moments), and they had no real offline model.

The availability of two auto-updating solutions, Jamie Cool's "AppUpdater" and the Updater Application Block released on MSDN, provided a better experience for the developer and the end user. With both solutions, applications were installed to the local machine, so they didn't have the performance or security issues of href-exes, and it was possible for an application to support offline use.

12.2.2.6 Basic ClickOnce Concepts

ClickOnce is a combination of a set of features in the .NET runtime (the CLR) and integrated design-time support in Visual Studio .NET. Together, the feature and the IDE tools allow you to create an application that can be automatically installed and updated. You have a great deal of flexibility as to how your application is deployed, launched, and updated.

12.2.2.7 Deployment Options

ClickOnce applications can be deployed to a client machine from a Web location, a UNC share, or even from a file location such as a CD. When an application is deployed in this fashion from a Web location, the user will click on a link on a Web page, which will cause the application to download and install itself on the client machine (complete with Start Menu shortcuts and a line in the Add/Remove Programs dialog). Once the installation completes, the application will then appear, having been downloaded and installed over the course of a few moments.

Alternatively, ClickOnce applications can be run without deploying them to the client machine; they can be launched directly from a network location (such as an UNC path or an http URL), with no impact to the client. Applications launched in this fashion are cached locally, but they are not installed in the traditional sense.

If you need an install that impacts the client machine (to create file associations, install MSDE, or perform other similar actions) then you still have the ability to handle that side of the installation through an .MSI file. Of course, everything you do outside of the regular ClickOnce deployment process can reduce the ClickOnce functionality, but if required then it is certainly possible.

For either launched or deployed applications to work, the .NET Framework is required on the target machine. Depending on your particular situation, there are a variety of ways to get the Framework onto a machine in advance of your install or as a part of the install itself.

12.2.2.8 Updating the Application

Updates are as flexible as deployment options, allowing you to update at certain times (such as the start of the application), or whenever the application developer chooses to call the

appropriate update APIs. It is also possible to specify that a certain update is required, removing the user's ability to ignore an available update. This required update feature is critical for managing ongoing updates and ensuring that the entire user-base can be moved up to a new version in a timely fashion.

Other key features of the update process include the ability to do rollbacks of an application release on either the client or the server. In the case of server rollbacks, this administrative feature allows for the quick removal of a flawed update and the automatic downgrade of your clients to the previous version. The client side of the rollback is a critical feature for occasionally connected clients; if they connect, update, and disconnect, only to find that the new application version fails on their machine, they can rollback to the previous version without having to reconnect. With all of the options that are available, you will be able to achieve whatever specific update functionality you require.

12.2.2.9 Manifest Files

The deployment and update activities can be controlled through the use of a pair of XML manifest files that describe the components of the system and the deployment activities that should occur. The application manifest is authored by the developer (with the help of Visual Studio .NET) and details the files, dependencies, and base security requirements of the application. The deployment manifest is intended to be created by the administrator (although the developer will certainly be creating this manifest during the early phases of development), and it details how the application is supposed to be deployed and updated.

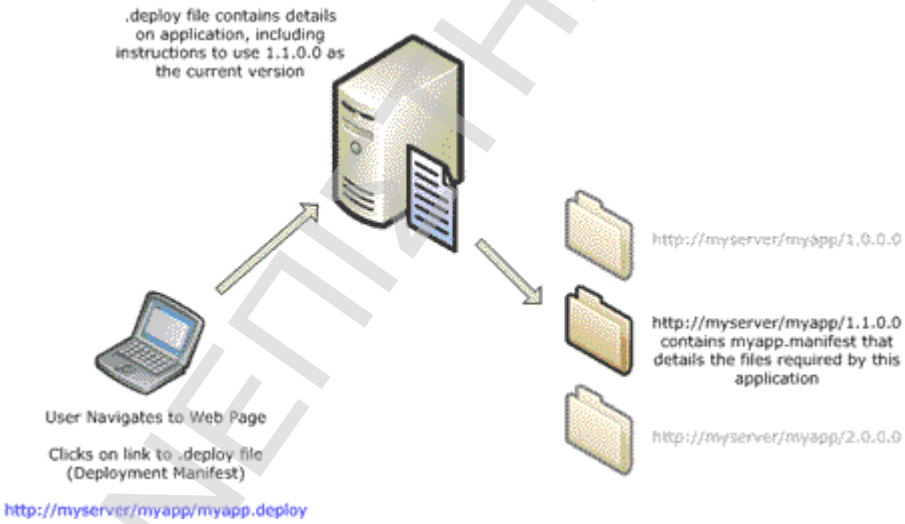


Figure 12-5: *The manifest files work together to enable the installation and update of your ClickOnce application.*

12.2.2.10 Security Concepts

Prior to ClickOnce, security was one of the biggest hurdles in getting an href exe to run correctly on the client's machine, and it is still a critical concept for ClickOnce applications.

Thankfully, the security "sandbox" that your application is allowed to play in has been increased in size, and the boundaries of this sandbox are much easier to understand and explore. Due to the auto-install and auto-update functionality of ClickOnce applications, they are intended to support a safe deployment model. Deployed or automatically downloaded code is limited as to what information it can access and what actions it can take. Whether an application is launched from a URL or deployed from a CD, it is not assumed to have full trust on the target machine.

Although the limitations of the default sandbox for auto-updating applications have been decreased, it has been difficult to troubleshoot a problem that only occurs in a reduced security situation, until now. The Whidbey release of Visual Studio .NET allows you to run your application in a reduced security situation from within the IDE, and with the debugging tools attached. This new feature will allow you to simulate various security settings and restrictions while developing your application, instead of having to deploy your application to a test machine to view any security issues.

12.2.2.11 Visual Studio Integration

ClickOnce is a key feature of Whidbey, which means that, in addition to being documented and supported, it is tightly integrated into the Whidbey version of Visual Studio .NET. Using the property dialogs, you can specify a variety of ClickOnce information (that will be used to create the two manifest files discussed earlier), including the publishing details.

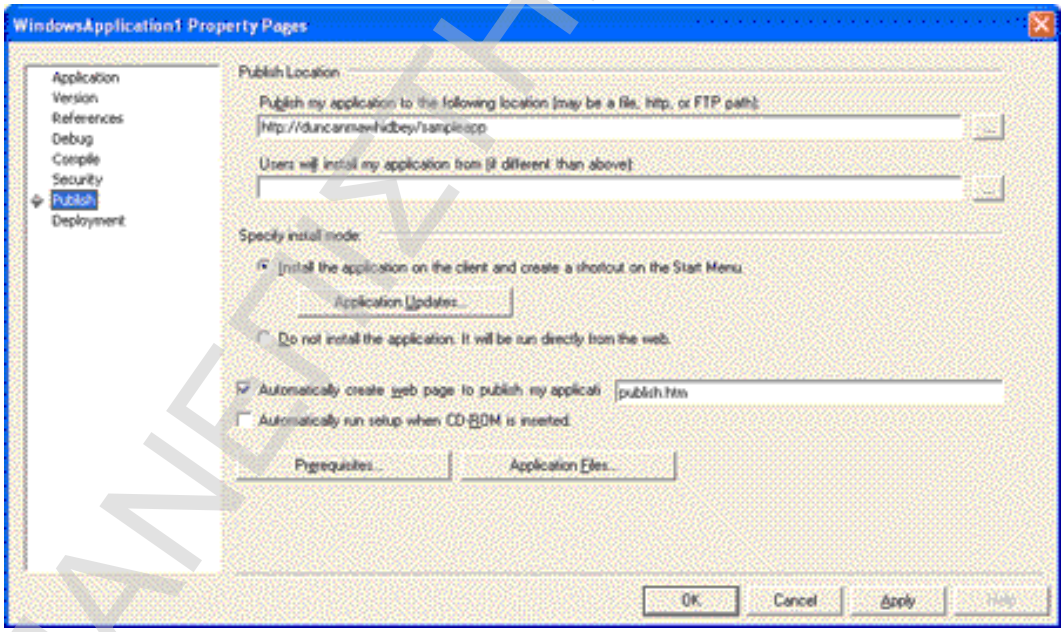


Figure 12-6: *You can specify the location your files should be published to (and accessed from) either in this property dialog or as part of the wizard that runs when you publish your application.*

The Visual Studio .NET integration extends to more than just configuring your publishing and updating details inside the IDE. You can also publish the application directly from within the IDE by clicking **Publish** from the Project menu, and then walking through a quick wizard of options. I will walk through this process a little later in this chapter, but it is quite simple to use. Although not only aimed at ClickOnce users, another powerful Visual Studio .NET feature allows you to run your application in the IDE (with a debugger attached) under whatever security context you wish. This feature will really help you to understand the security restrictions your application will be running under. It will also help you to properly debug security-related issues using the full tools of the Visual Studio .NET IDE. [19], [21], [22]

13 System Presentation – User Manual

13.1 Basic Functionality of System Components

13.1.1 Network Manager

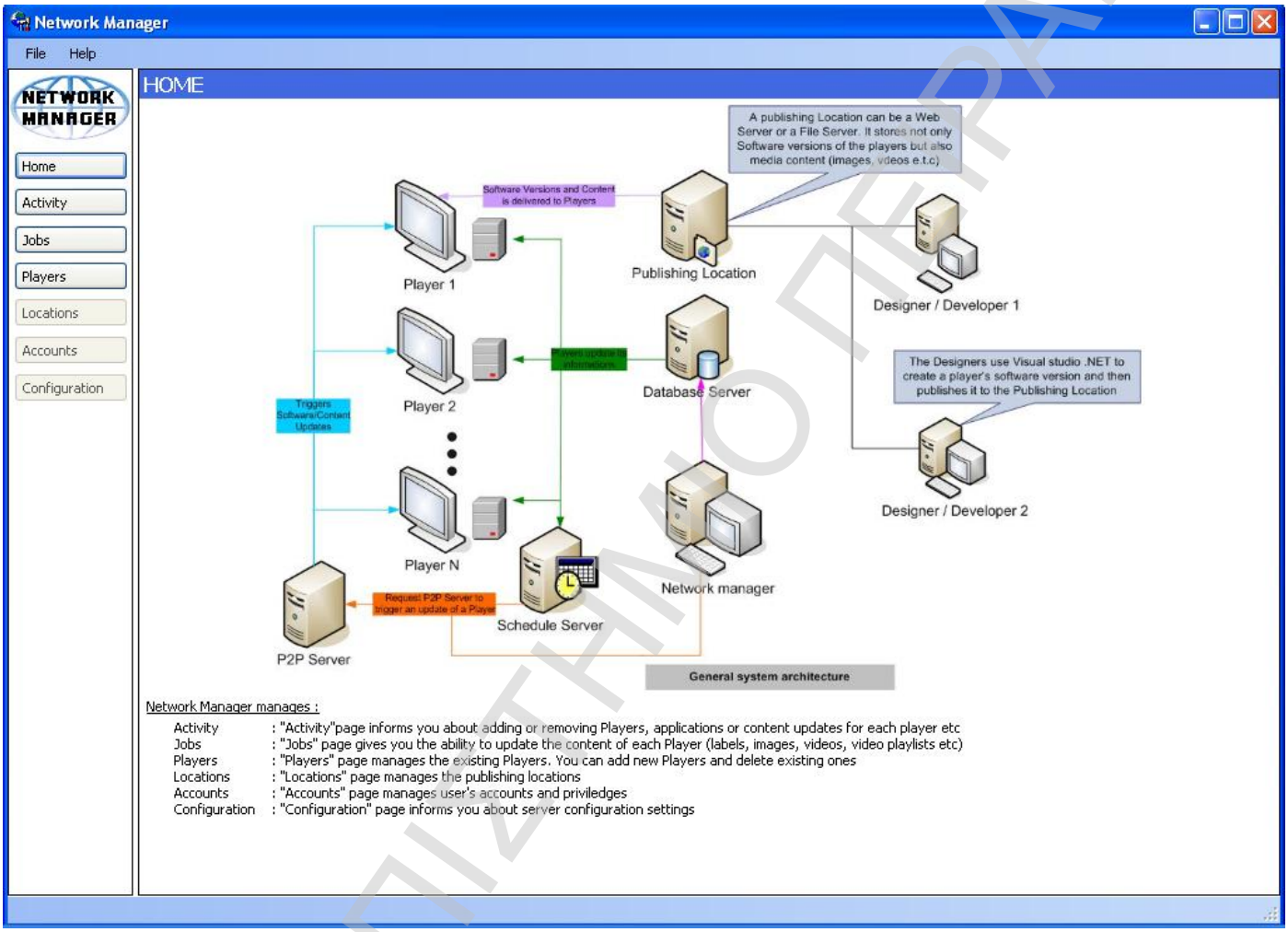


Figure 13-1 : *Network Manager – "Home" page*

Figure 13-1 present the user interface of the home page of Network Manager. User can navigate through the basic Menu on the top left corner of the application. The available buttons are:

- Home: It returns user to the "Home" page of the application
- Activity: It navigates user to the "Activity" page of the application
- Jobs: It navigates user to the "Jobs" page of the application
- Players: It navigates user to the "Players" page of the application
- Locations: Not available
- Accounts: Not available
- Configuration : Not available

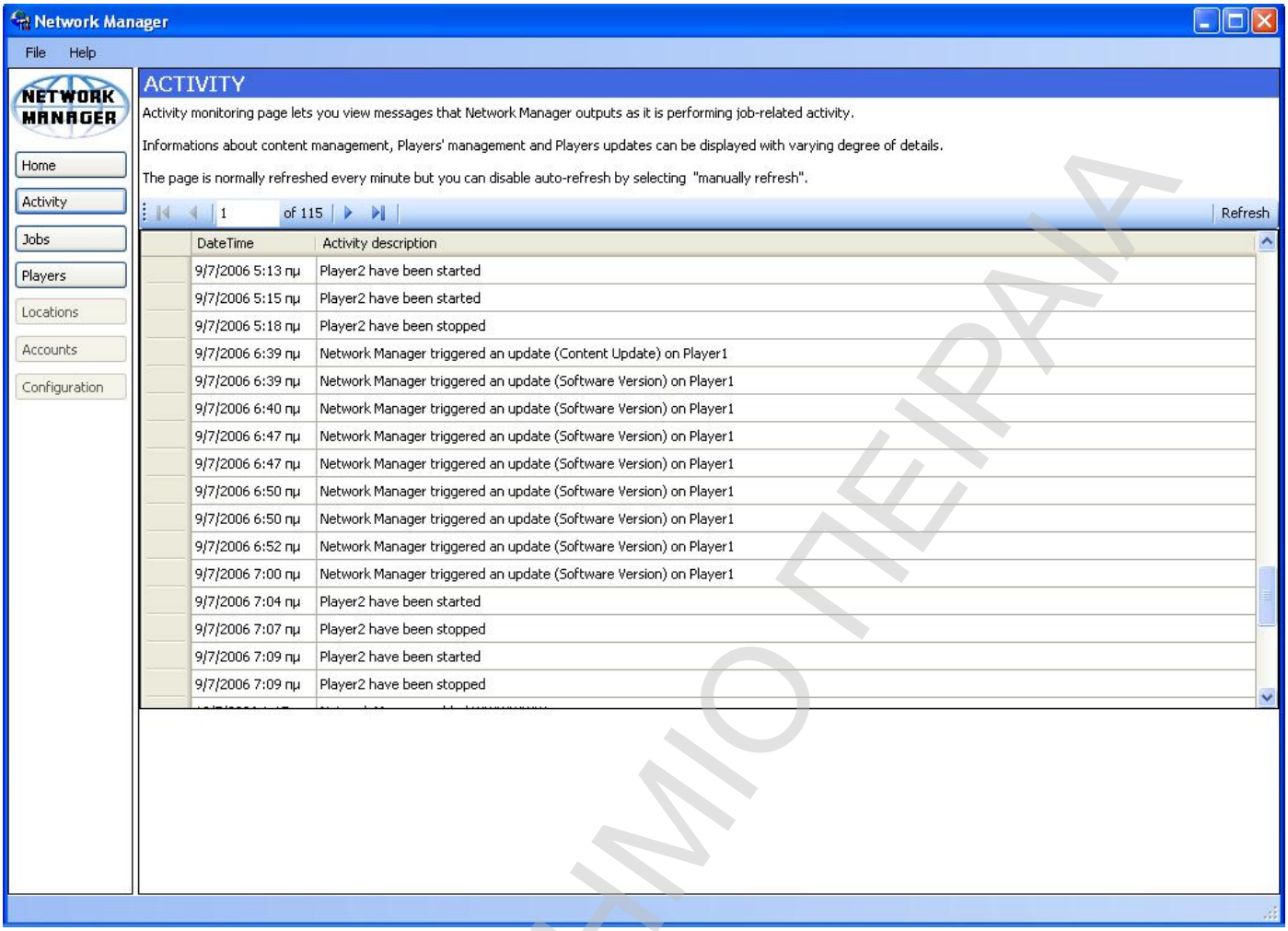


Figure 13-2: *Network Manager - "Activity" page*

Activity Monitoring page (Figure 13-2) lets user view messages that Network Manager, Players and all the others system components output as they are performing job–related activity.

“Activities” represents actions that refer in the whole distributive system. They are categorized in three categories according to their derivation. Time of the activity is always recorded. Activities messages include :

- From Network Manager
  - Network Manager added PlayerX
  - Network Manager removed PlayerX
  - Network Manager set as current content update ContentUpdateX of PlayerX softwareversionX
  - Network Manager added contentupdateX of PlayerX versionX
  - Network Manager deleted ContentupdateX of PlayerX versionX
  - Network Manager added VersionX of PlayerX
  - Network Manager triggered a content update on PlayerX
  - Network Manager triggered software version update on PlayerX

- From Players
  - PlayerX started
  - PlayerX stopped
  - PlayerX have been updated (content update)
  - PlayerX have been updated (software version update)
- From Calendar
  - A new schedule have been added (PlayerX, ContentupdateX, TIME)

A navigator helps user to navigate through recorded activities. There is also a “refresh” buttons that refresh activity panel.

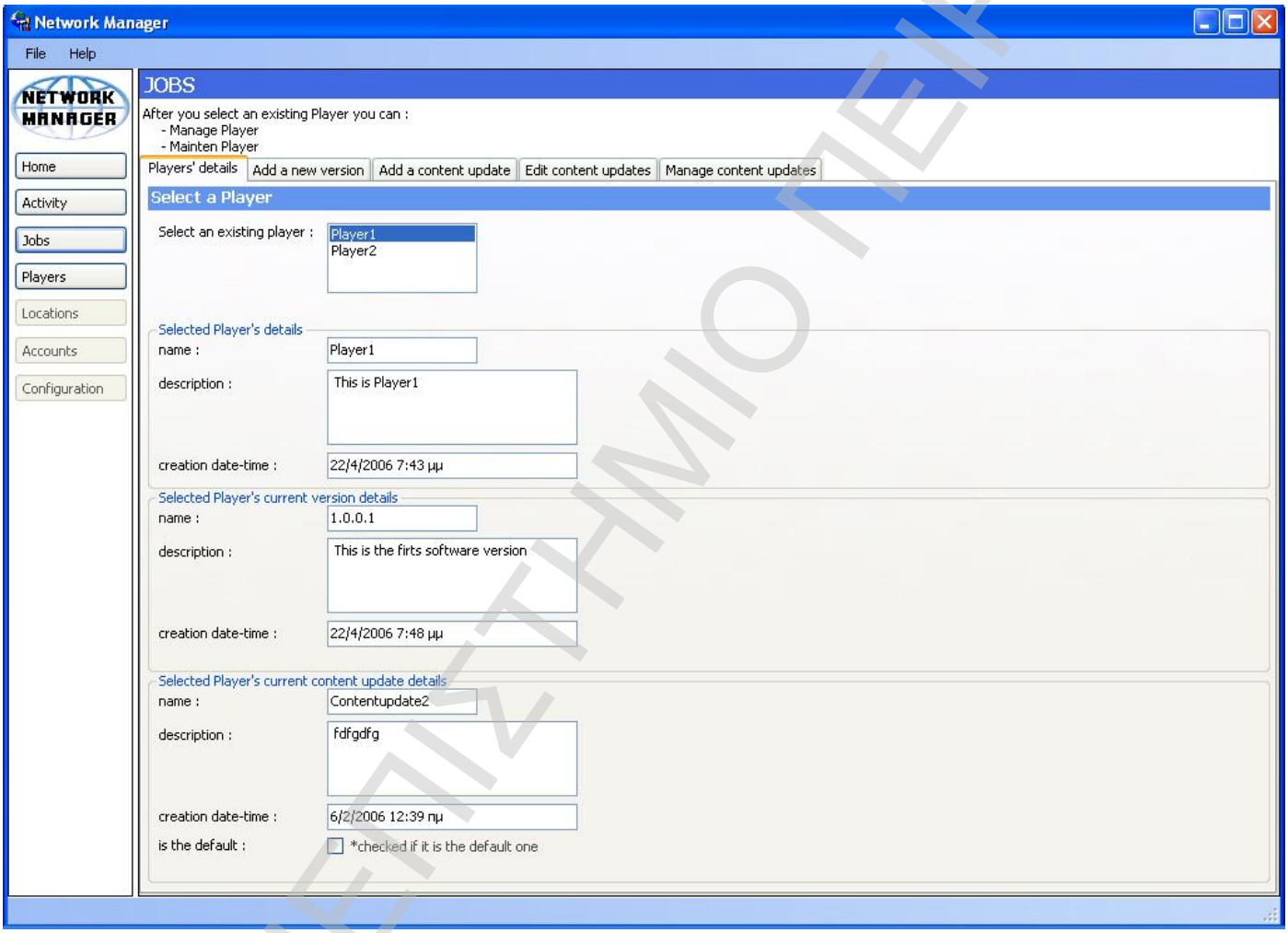


Figure 13-3 : *Network Manager - "Jobs" page’s Players’ details panel*

“Jobs” page’s “Player’s details” panel (Figure 13-3) allow user to have an overall view of all Players. He can see how many Players are available and further details for each Player (name, description, current software version details and current content updates details).

“Jobs” page’s “Add a new version” panel enables user to ad a new software version of a registered Player (a player that has been added).

There is a wizard of five steps (Figures 13-4 – 13-9):

1. STEP1: The user has to select an available Player

2. The user has to fill the name and the description of the software update
3. The user has to fill the name and the description of the default content update
4. The user has to specify the properties of the main form of the default content update
5. The user has to register which controls will be the editable ones and fill their names and descriptions

If user will stop the wizard manually or step out by navigating of the main menu, the wizard and all the actions, since then, will be rolled-back.

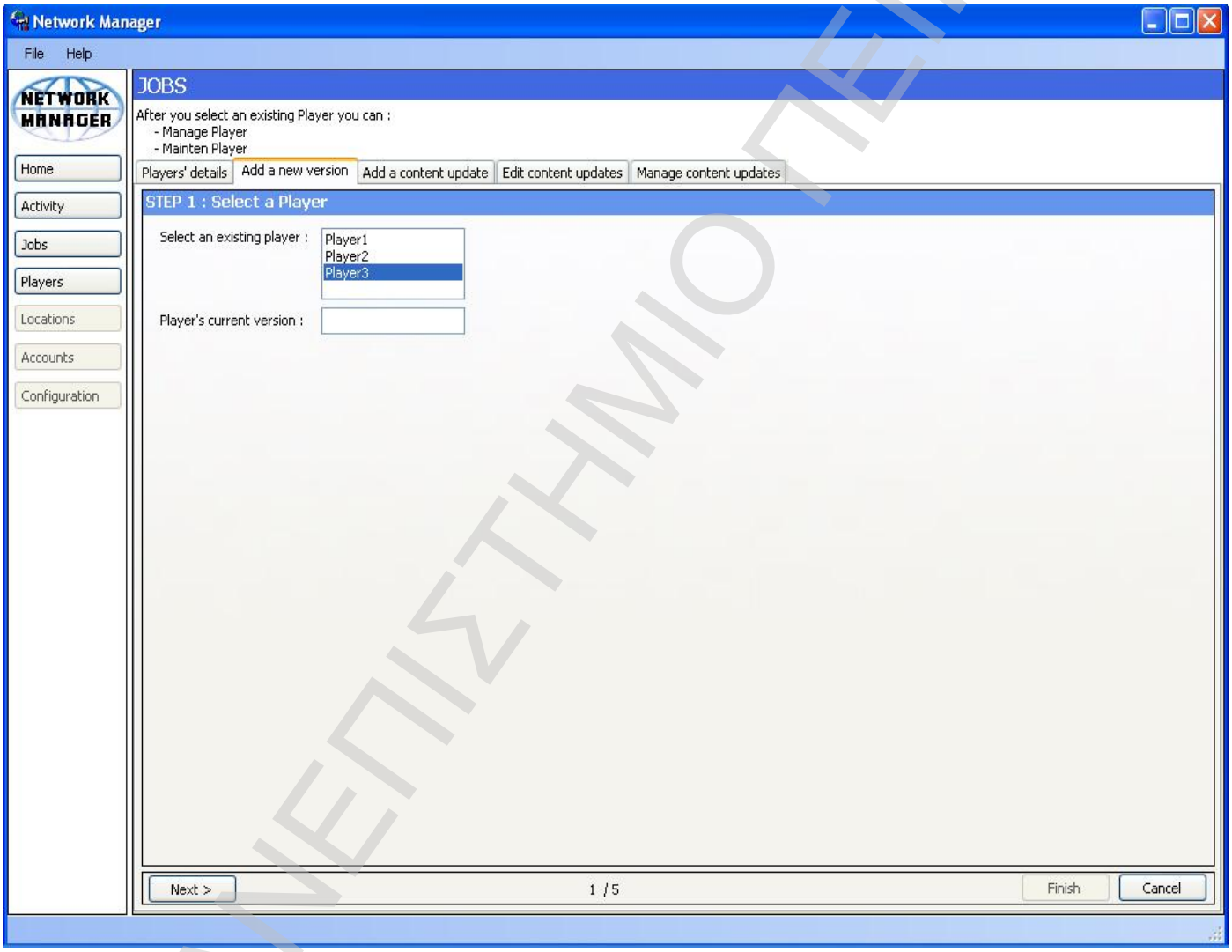


Figure 13-4 : *Network Manager - "Jobs" page's "Add a new version" panel (Step 1)*

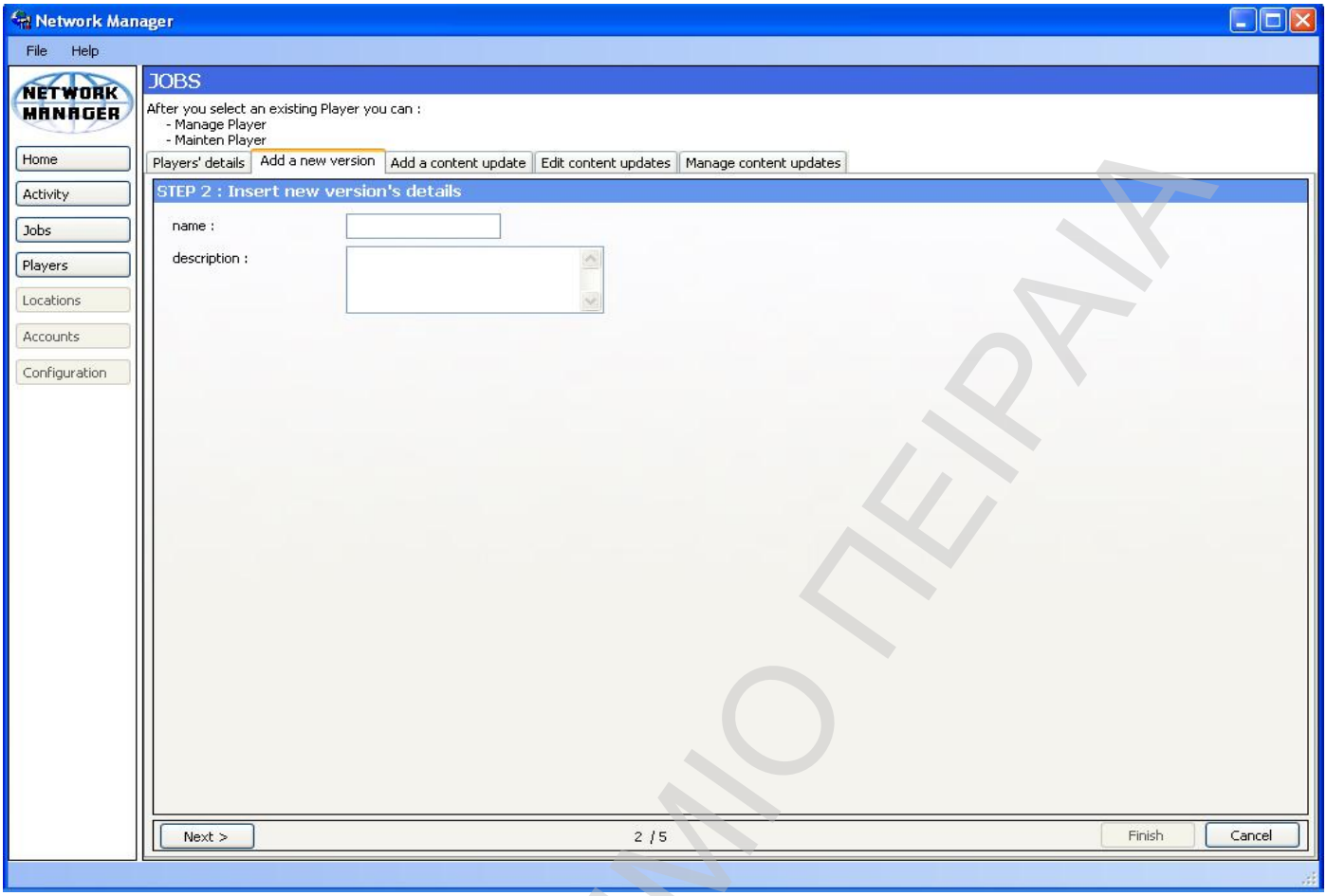


Figure 13-5 : *Network Manager - "Jobs" page's "Add a new version" panel (Step 2)*

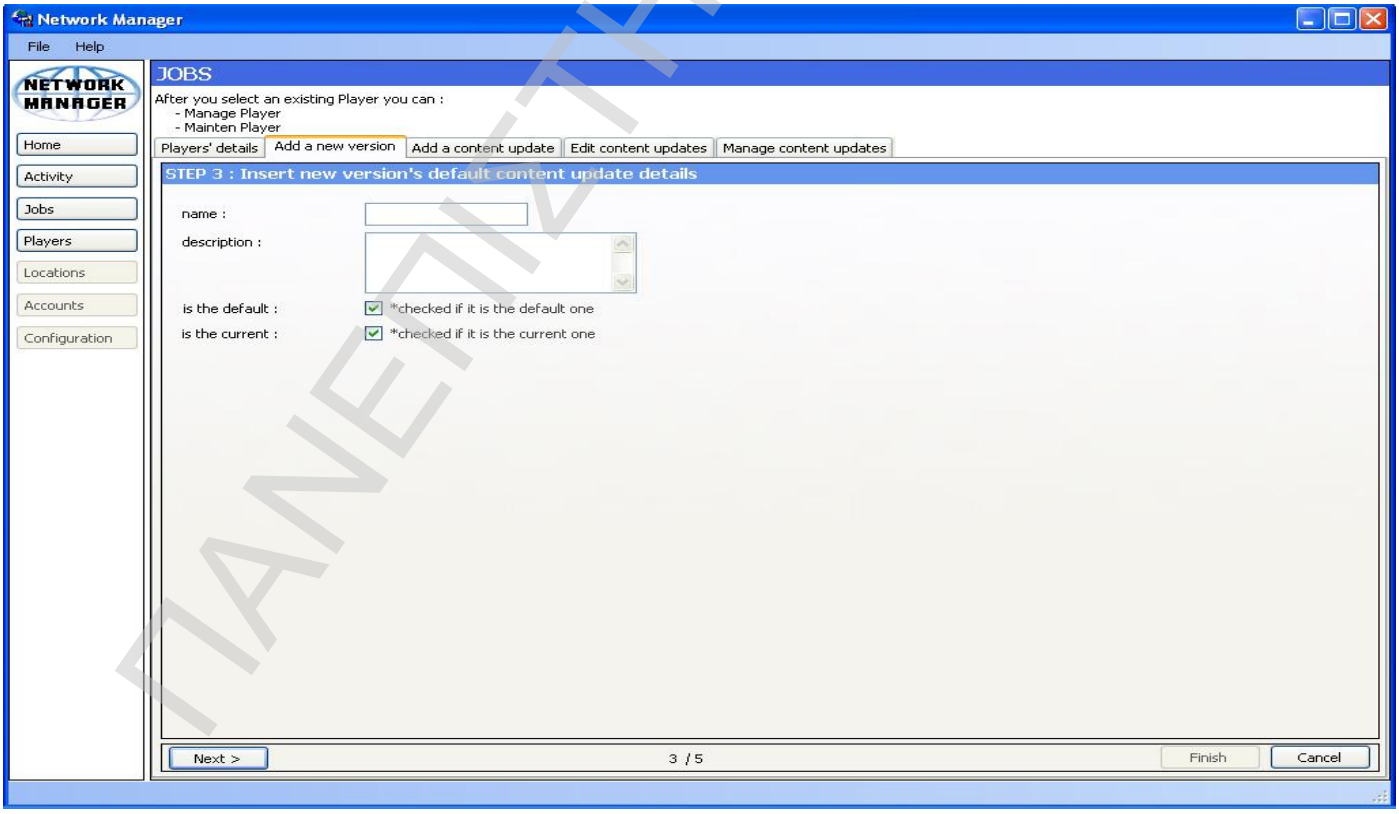


Figure 13-6 : *Network Manager - "Jobs" page's "Add a new version" panel (Step 3)*

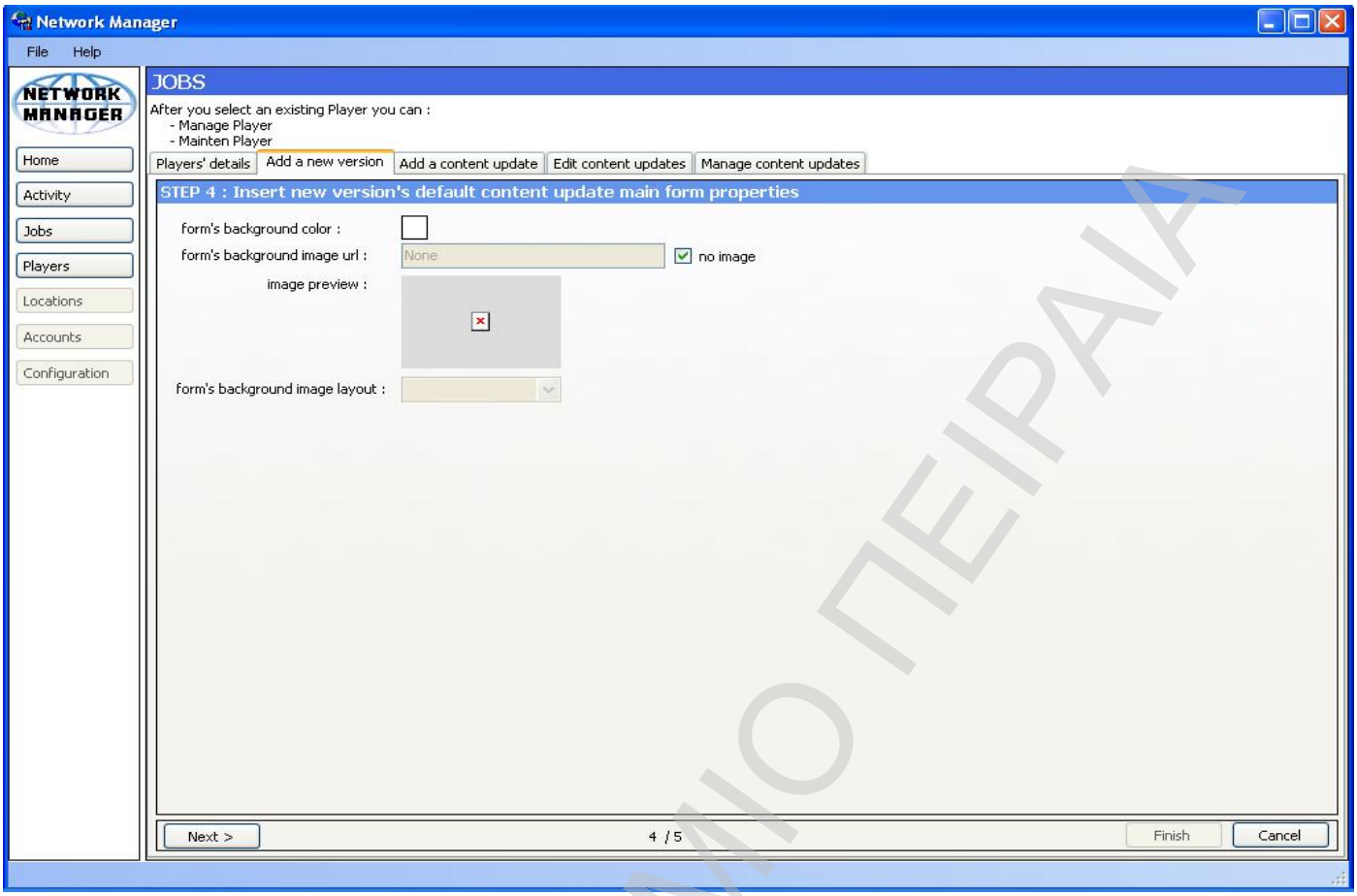


Figure 13-7: *Network Manager - "Jobs" page's "Add a new version" panel (Step 4)*

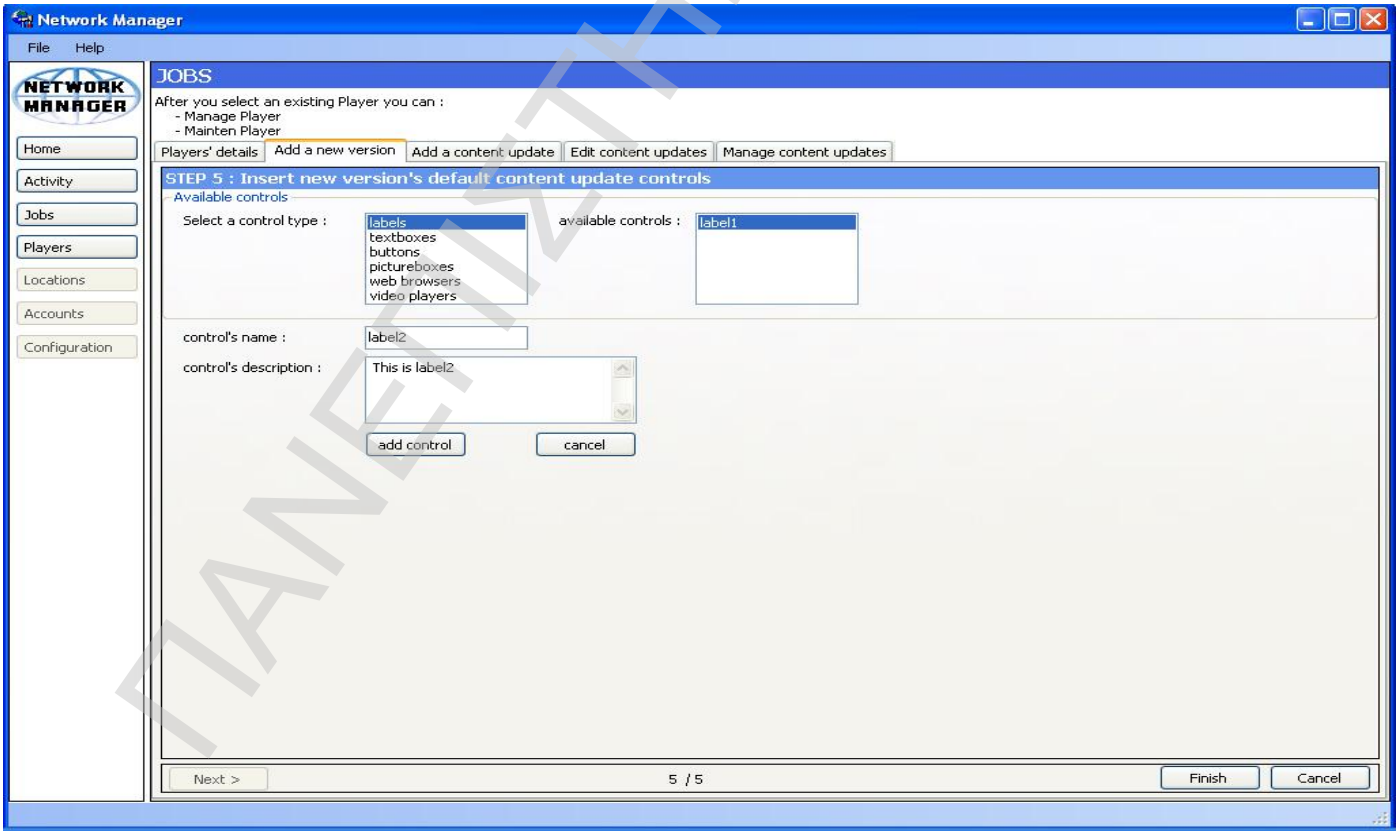


Figure 13-8: *Network Manager - "Jobs" page's "Add a new version" panel (Step 5)*

"Jobs" page's "Add a content update" panel (Figure 13-9) enables user to add a new content update. First he selects an available Player and then he fills new content update's name and description. The new content update inherits controls from the default content update.

After, the user can edit the new content update in the "Edit content updates" panel (Figure 13-10 – 13-13). "Edit content updates" panel allows user to delete an available content update too. The editor panel allows controls' editing. Each control type has different editable properties. The properties of each control type are listed below:

- Forms (This control has properties that represent the properties of the main form player's application)
  - BackColor : Background color of the form
  - BackgroundImageUrl : Background image url of the form
  - BackgroundImageLayout : Background image layout of the form
- Labels
  - Text : Text of the label
  - BackColor : Background color of the label
  - BorderStyle : Border style of the label
  - Enabled : Enable status of the label
  - Font : Font of the label (includes font name, font size, etc)
  - ForeColor : Foreground color of the label
  - Size : Size of the label
  - TextAlign : Text align of the label
  - Visible : Visible status of the label
  - AutoSize : Autosize status of the label
  - Location : Location of the label
- Textboxes
  - Text : Text of the textbox
  - BackColor : Background color of the textbox
  - BorderStyle : Border style of the textbox
  - Enabled : Enable status of the textbox
  - Font : Font of the textbox (includes font name, font size, etc)
  - ForeColor : Foreground color of the textbox
  - Size : Size of the textbox
  - TextAlign : Text align of the textbox
  - RedaOnly : Read only status of the textbox
  - Visible : Visible status of the textbox
  - MultiLine : Multiline status of the textbox
  - ScrollBars : Scrollbars of the textbox
  - Location : Location of the textbox
- Buttons
  - Text : Text of the button

- § BackColor : Background color of the button
- § Enabled : Enable status of the button
- § Font : Font of the button (includes font name, font size, etc)
- § ForeColor : Foreground color of the button
- § Size : Size of the button
- § TextAlign : Text align of the button
- § FlatStyle : Flat style of the button
- § AutoSize : Autosize status of the button
- § Visible : Visible status of the button
- § Location : Location of the button

Ø Pictureboxes

- § BackColor : Background color of the picturebox
- § BorderStyle : Border style of the picturebox
- § Enabled : Enable status of the picturebox
- § Size : Size of the picturebox
- § SizeMode : Sizemode of the picturebox
- § Visible : Visible status of the picturebox
- § Location : Location of the picturebox

Ø Webbrowser

- § Url : Url of the web browser
- § AllowNavigation : Allow navigation status of the web browser
- § ScrollBarsEnabled : Scroll bars enabled status of the web browser
- § Visible : Visible status of the web browser
- § Size : Size of the web browser
- § Location : Location of the web browser

Ø Videoplayers

- § URL : Url of the media file of the video player
- § Size : Size of the video player
- § Visible : Visible status of the video player
- § UiMode : Uimode of the video player (specifies if the videoplayers buttons are visible)
- § StretchToFit : Stretch to fit status of the video player
- § Location : Location of the video player

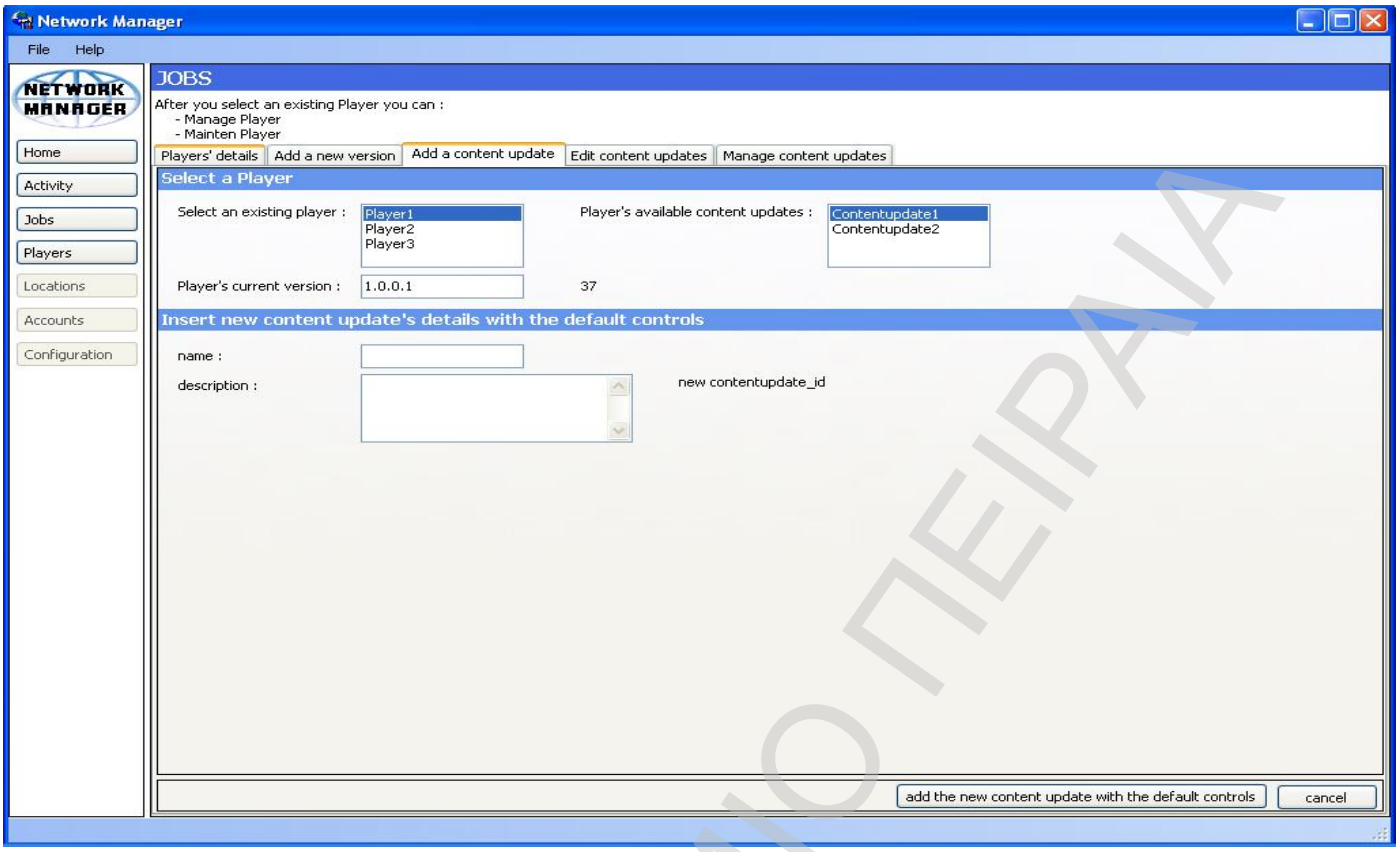
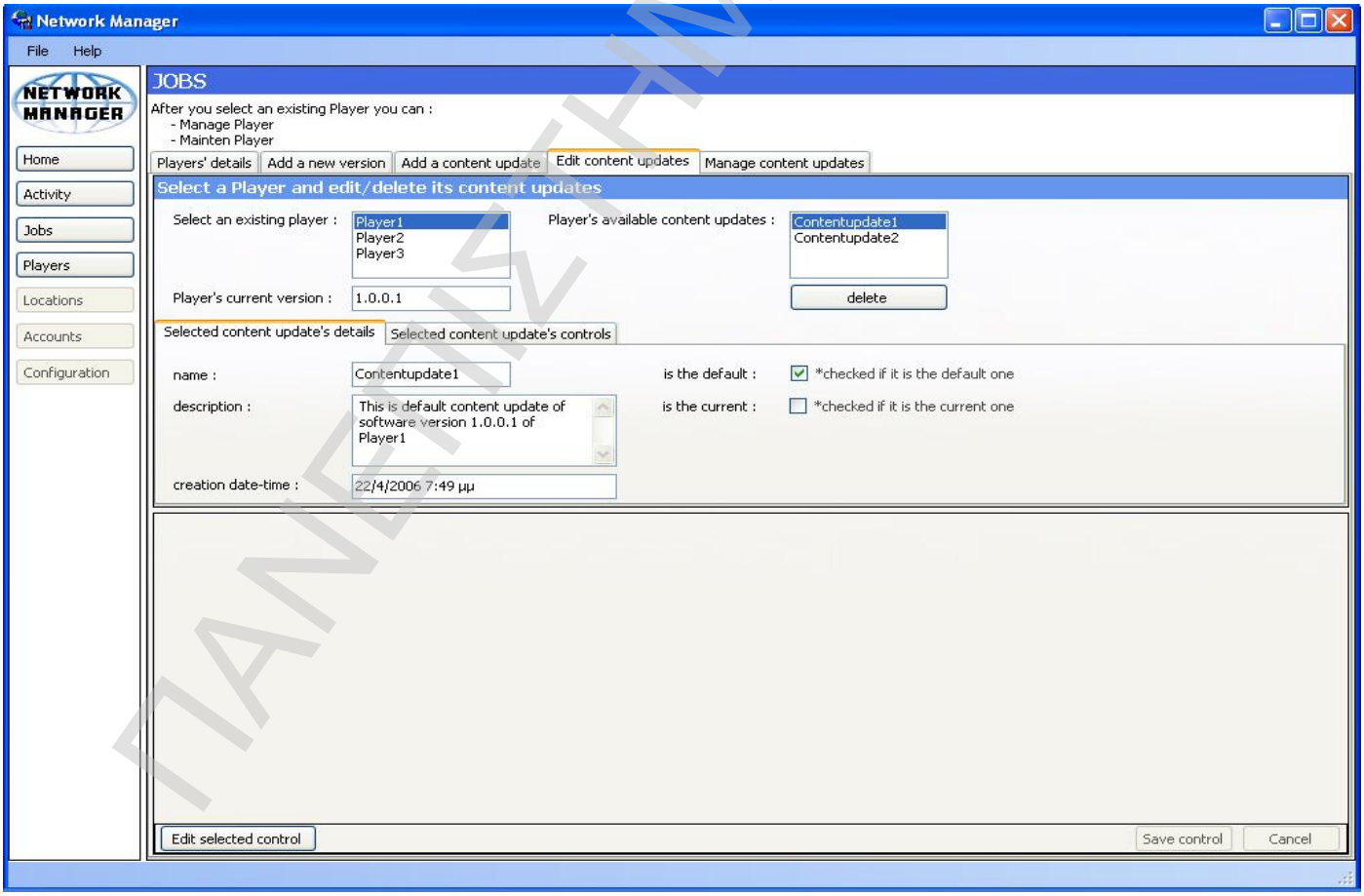


Figure 13-9: *Network Manager - "Jobs" page's "Add a new content update" panel*



Εικόνα 13-10: *Network Manager - "Jobs" page's "Edit content updates" panel*

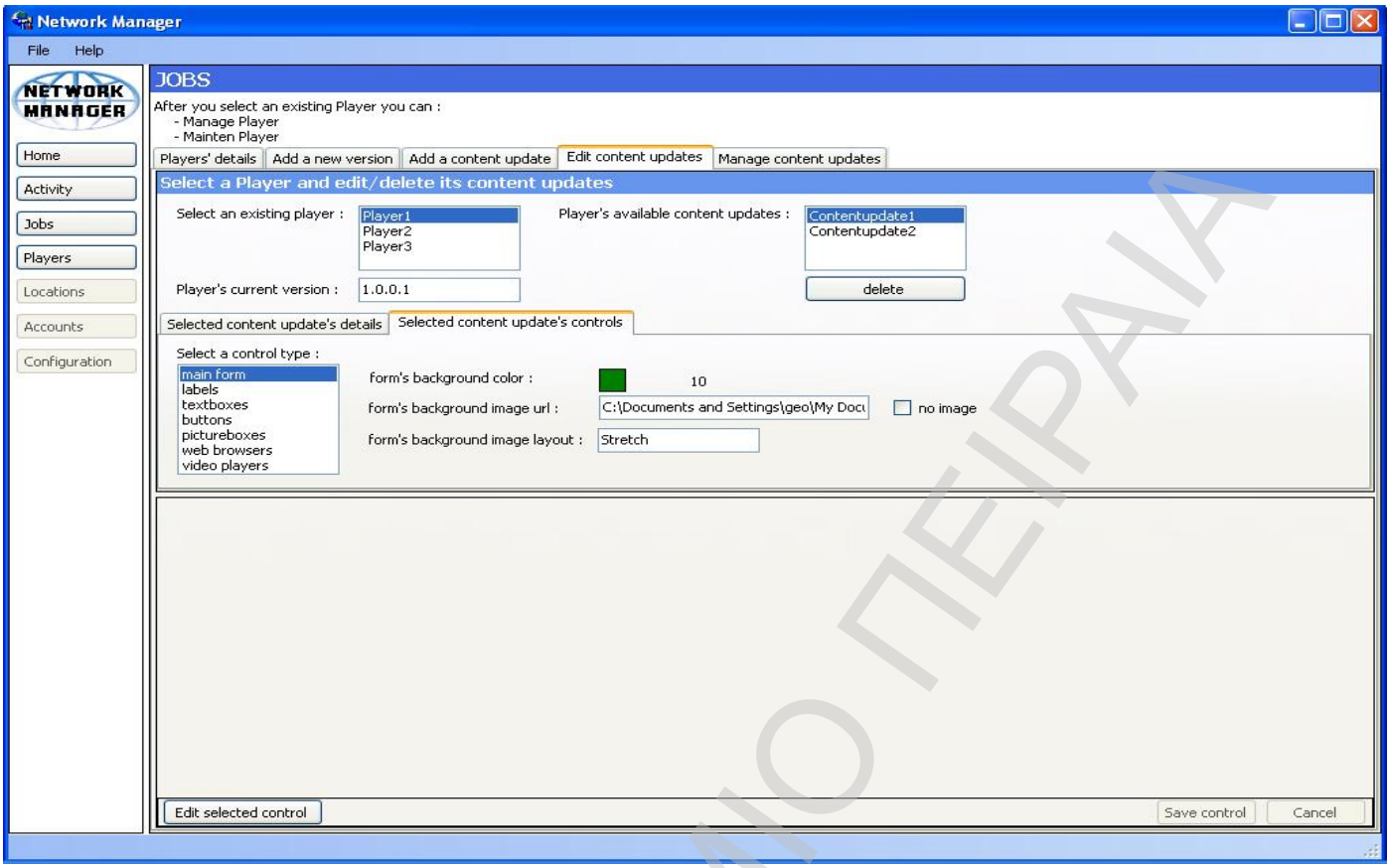


Figure 13-11: *Network Manager - "Jobs" page's "Edit content updates" panel*

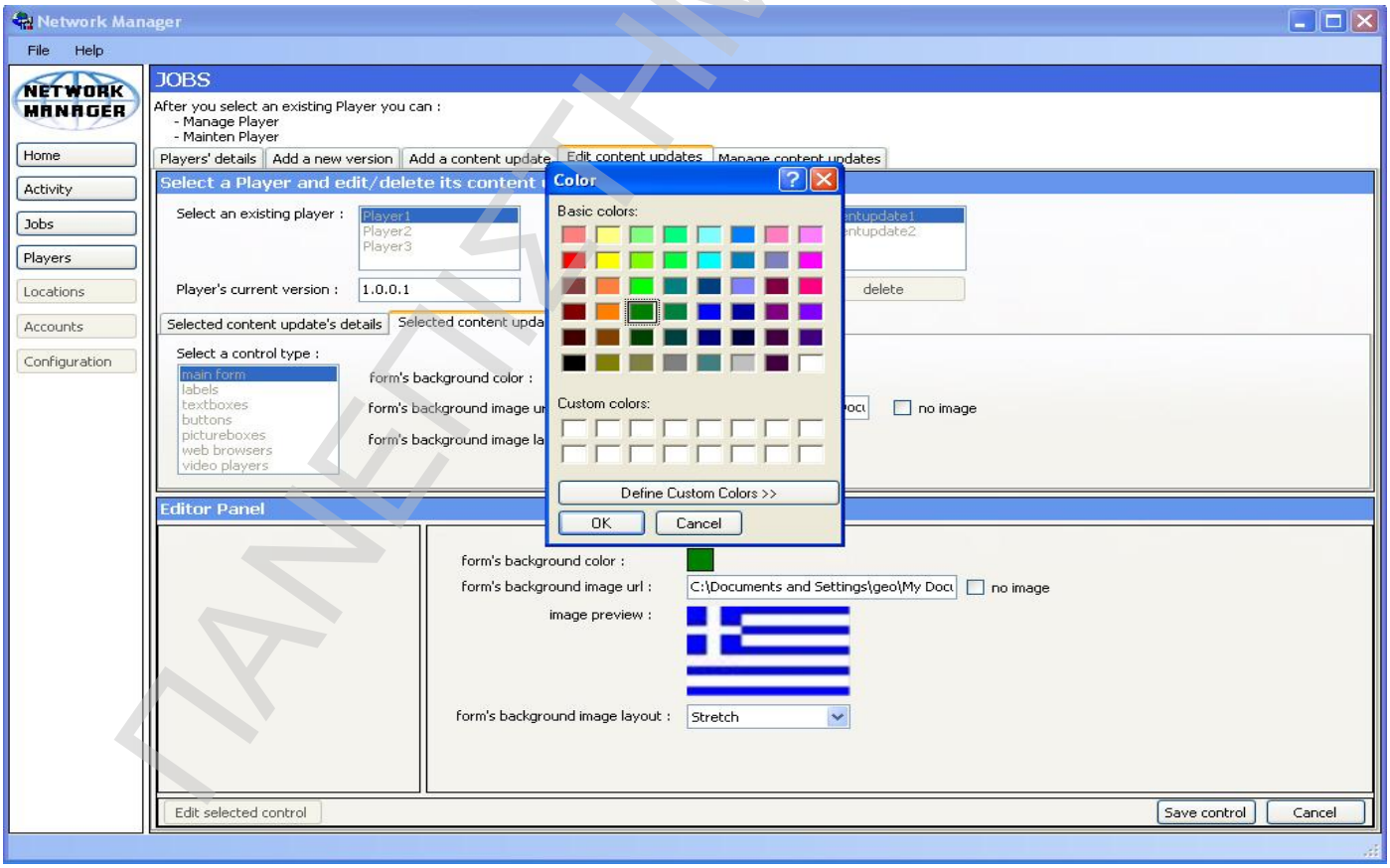


Figure 13-12: *Network Manager - "Jobs" page's "Edit content updates" panel*

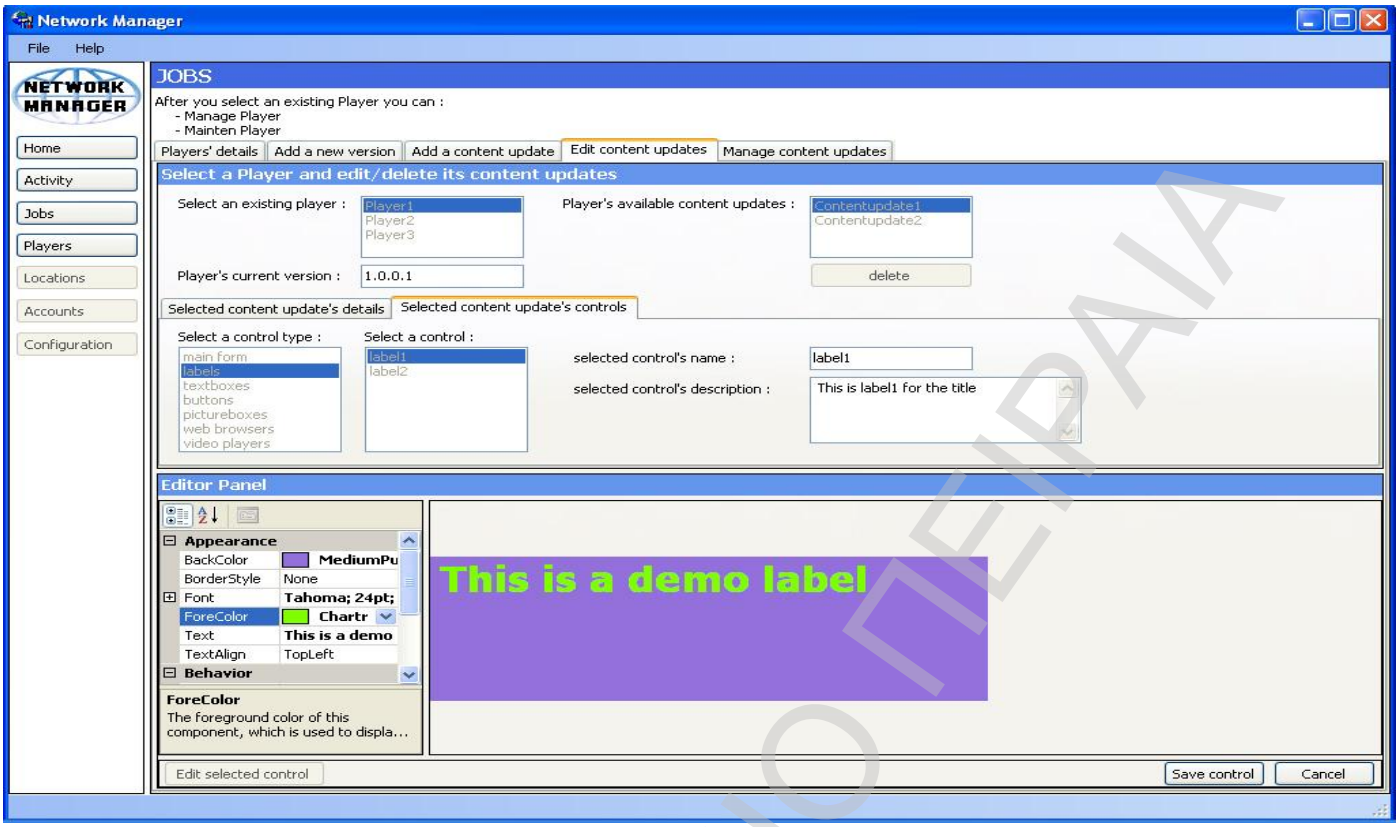


Figure 13-13 : *Network Manager - "Jobs" page's "Edit content updates" panel*

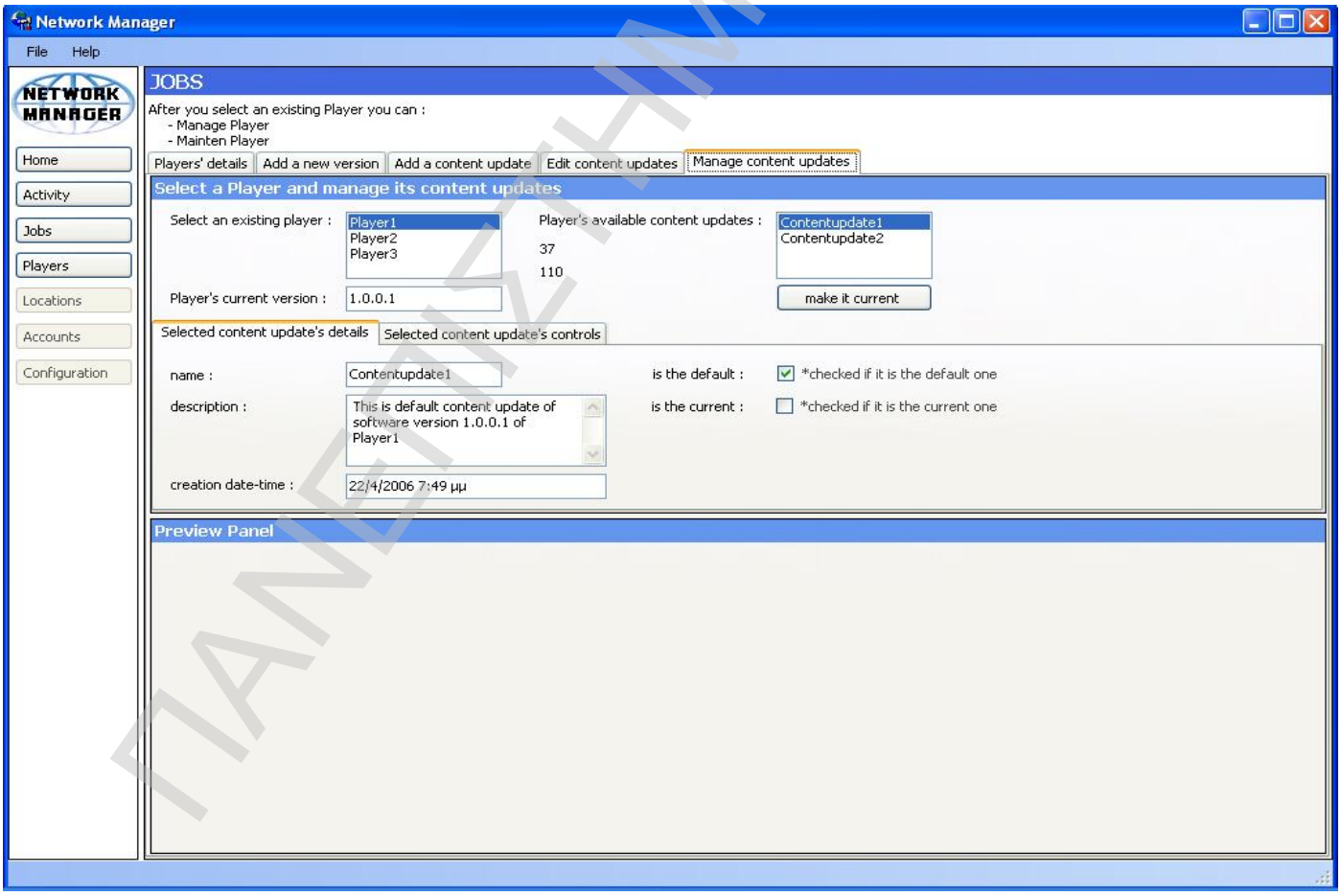


Figure 13-14 : *Network Manager - "Jobs" page's "Manage content updates" panel*

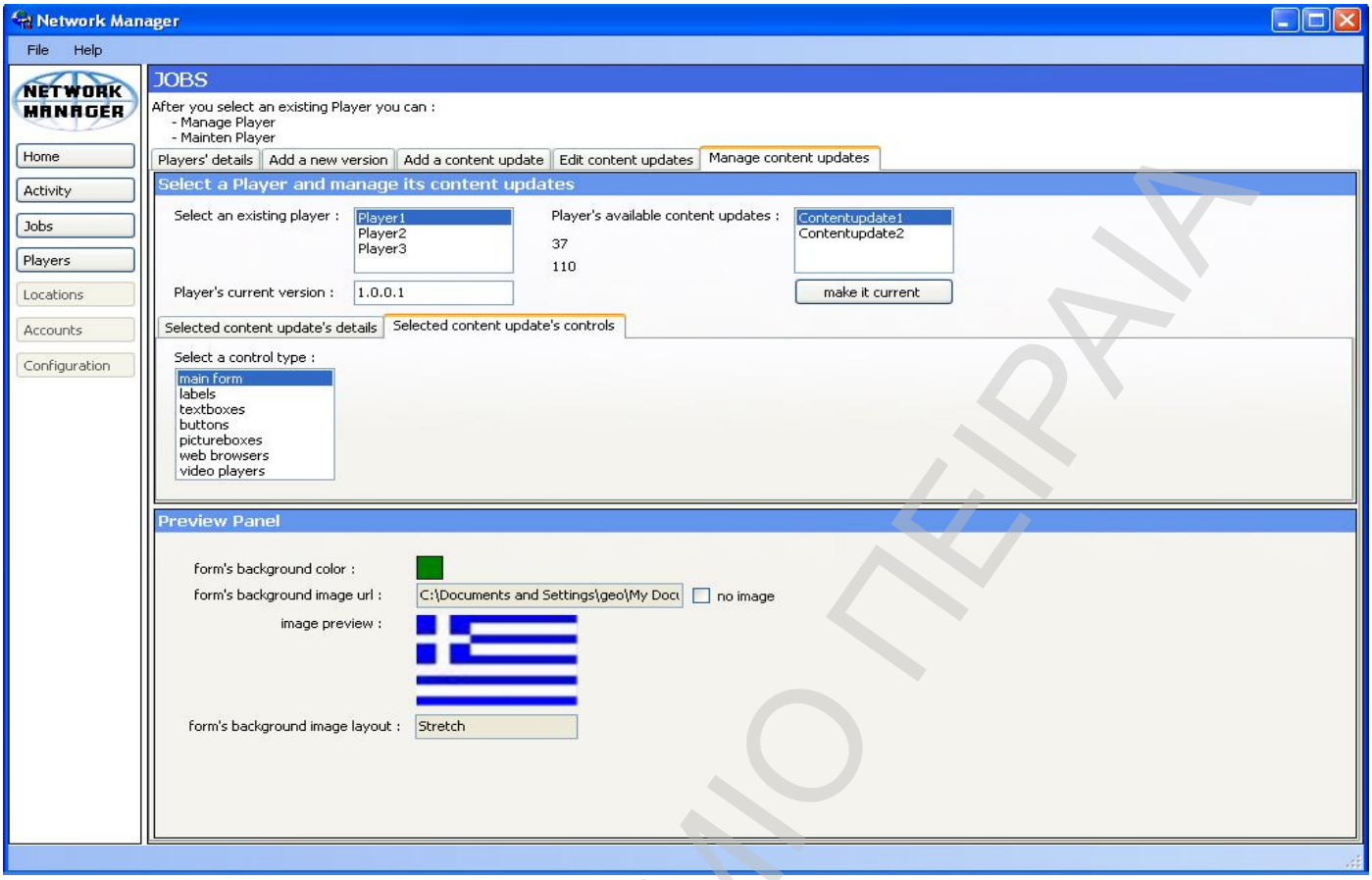


Figure 13-15 : *Network Manager - "Jobs" page's "Manage content updates" panel*

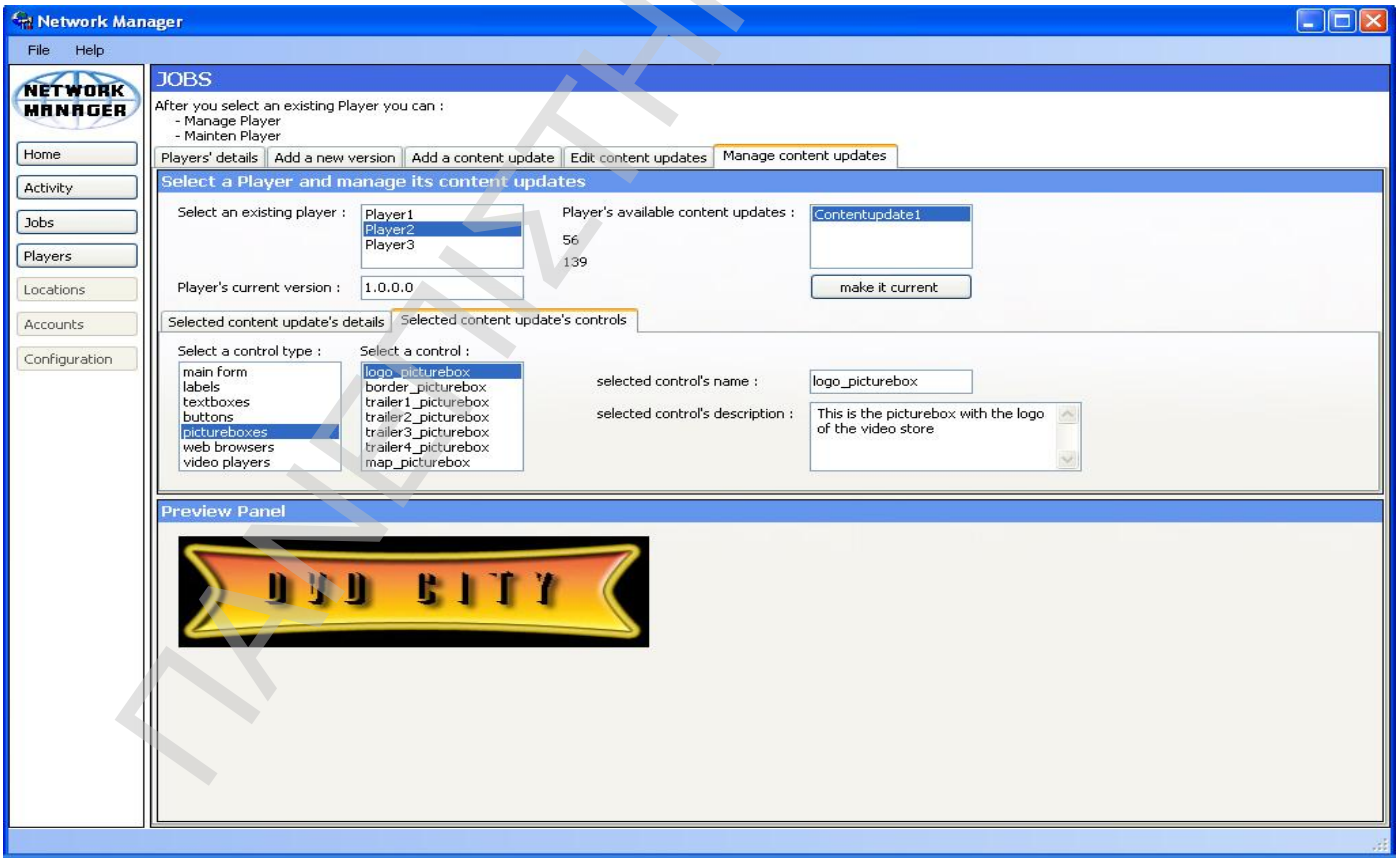


Figure 13-16 : *Network Manager - "Jobs" page's "Manage content updates" panel*

"Manage content updates" panel (Figure 13-14 – 13-16) enables user to manage efficiently Players' content updates. He can preview all the editable controls of all content updates and he can select a content update to make it current. Then he has to navigate to "Players" Page where he can connect with P2P Server to trigger the updates.

"Players" includes "Players details" panel (Figure 13-17) which is exactly the same as the one of the "Jobs" page. There is also "Add a new Player" panel (Figure 13-18) which enables user to add a new Player after filling its name and description.

"Remove a Player" panel (Figure 13-19) enables user to remove a player. When he removes one all the software versions, related content updates will be also removed.

Finally "Detect on-line Players and update" panel (Figure 13-20) enables user to connect with P2P Server, to get a list including active Players and if it is necessary to trigger a content's or a software version's update of a Player.

In addition, messages (errors, warnings,) accompany every action which a user can perform. By this manner user guidance is being succeeded.

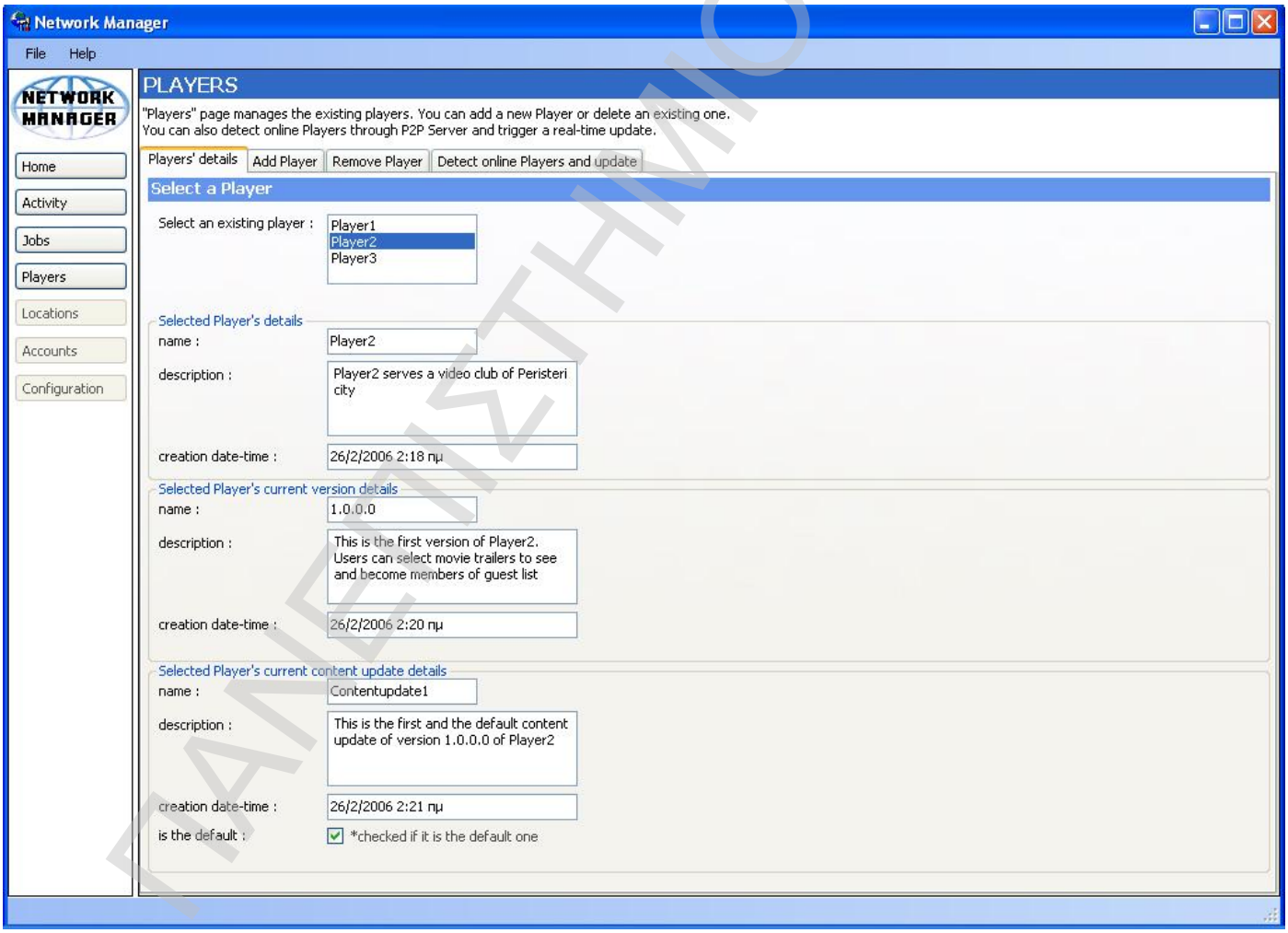


Figure 13-17: *Network Manager - "Players" page's "Players' details" panel*

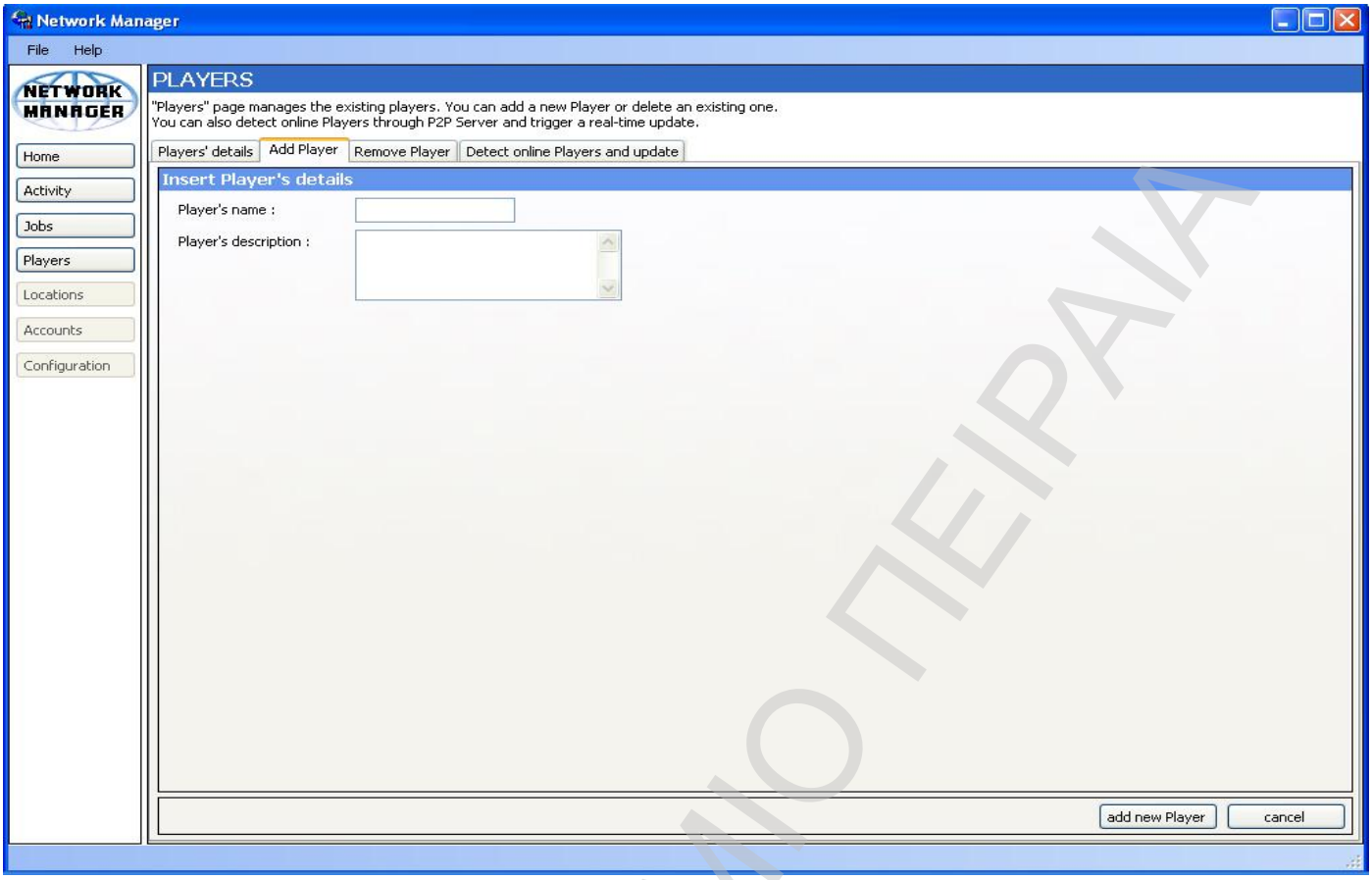


Figure 13-18 : *Network Manager - "Players" page's "Add a new Player" panel*

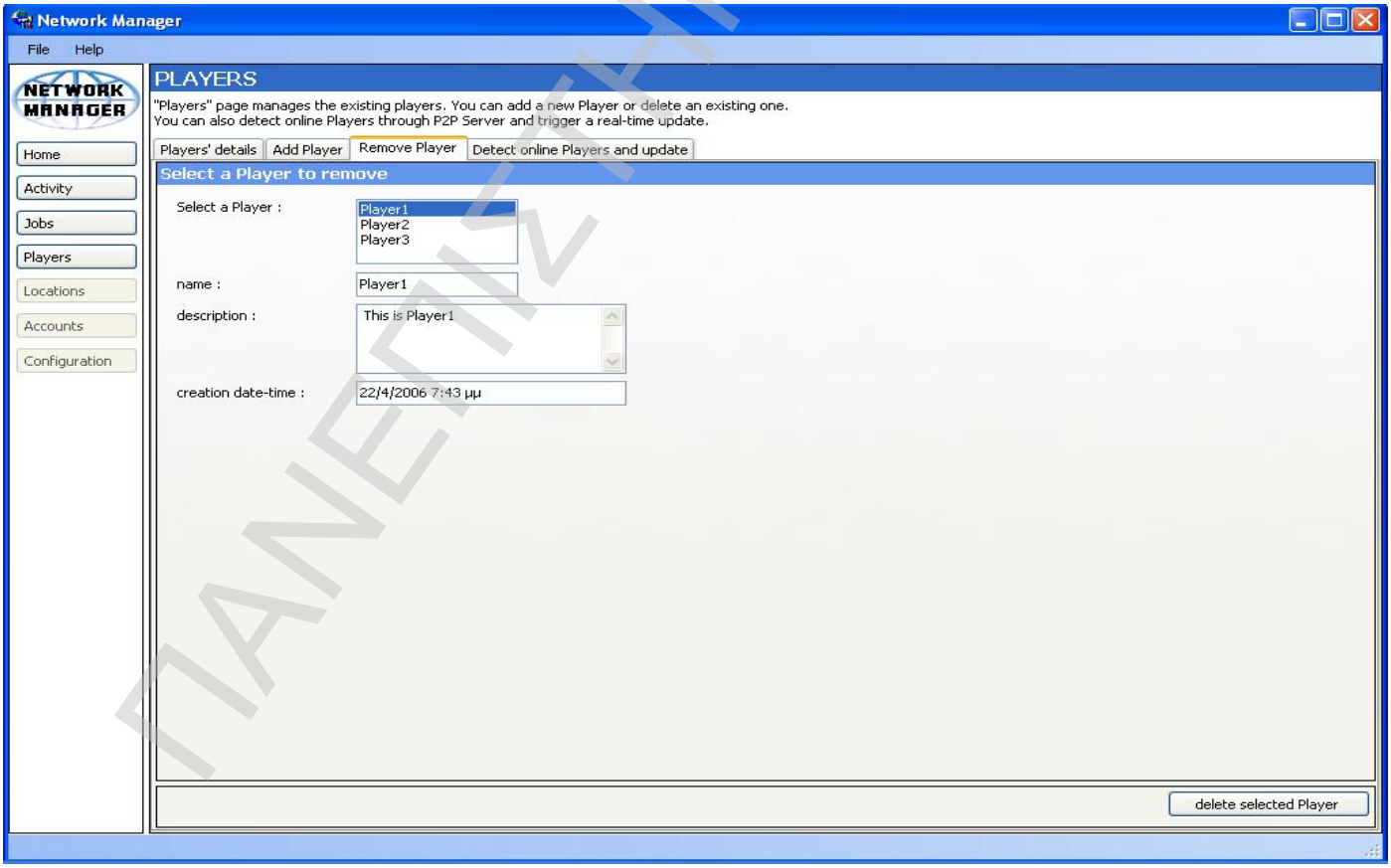


Figure 13-19 : *Network Manager - "Players" page's "Remove a Player" panel*

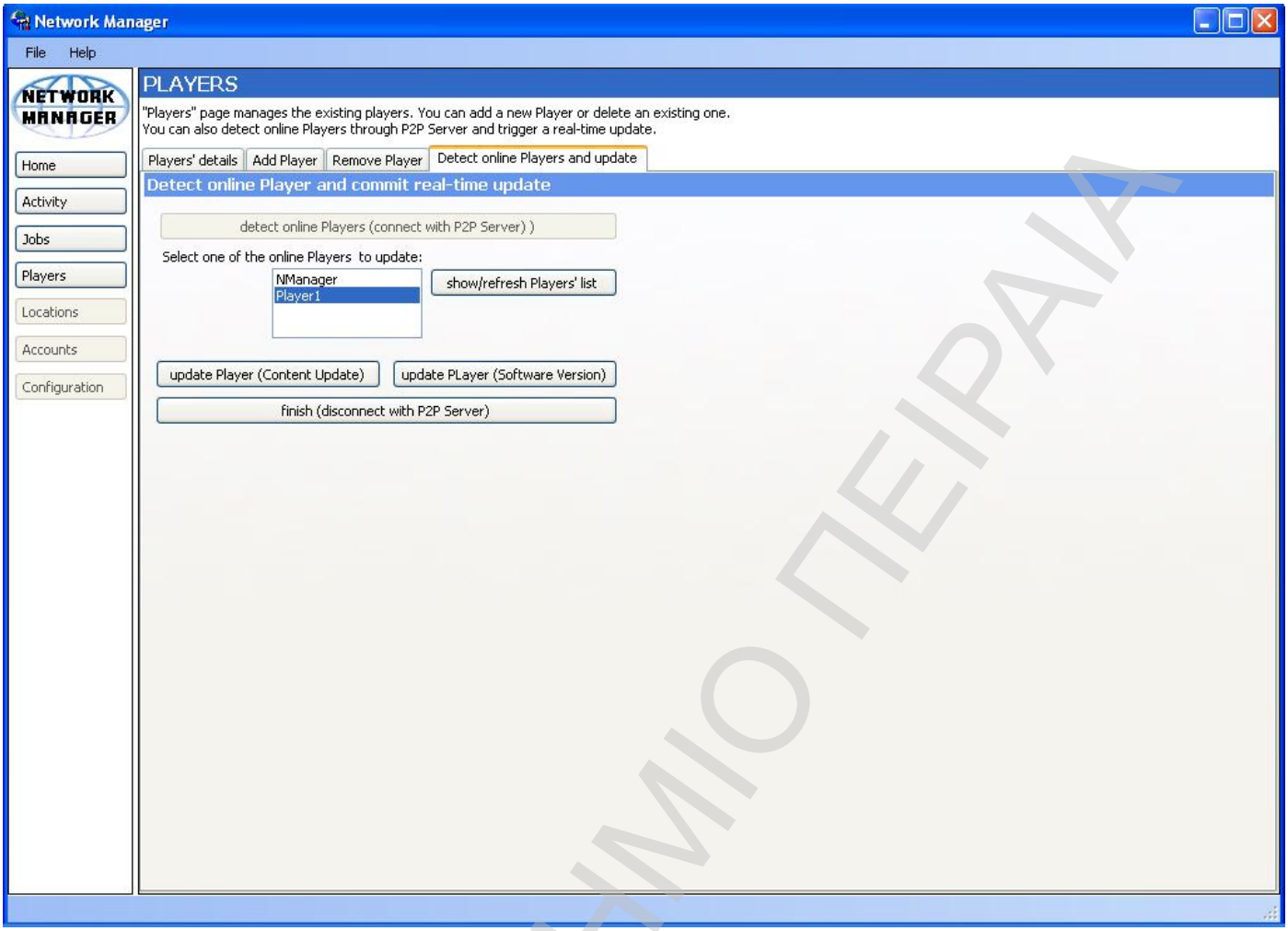


Figure 13-20 : *Network Manager - "Detect on-line Players and update" panel*

13.1.2 Calendar

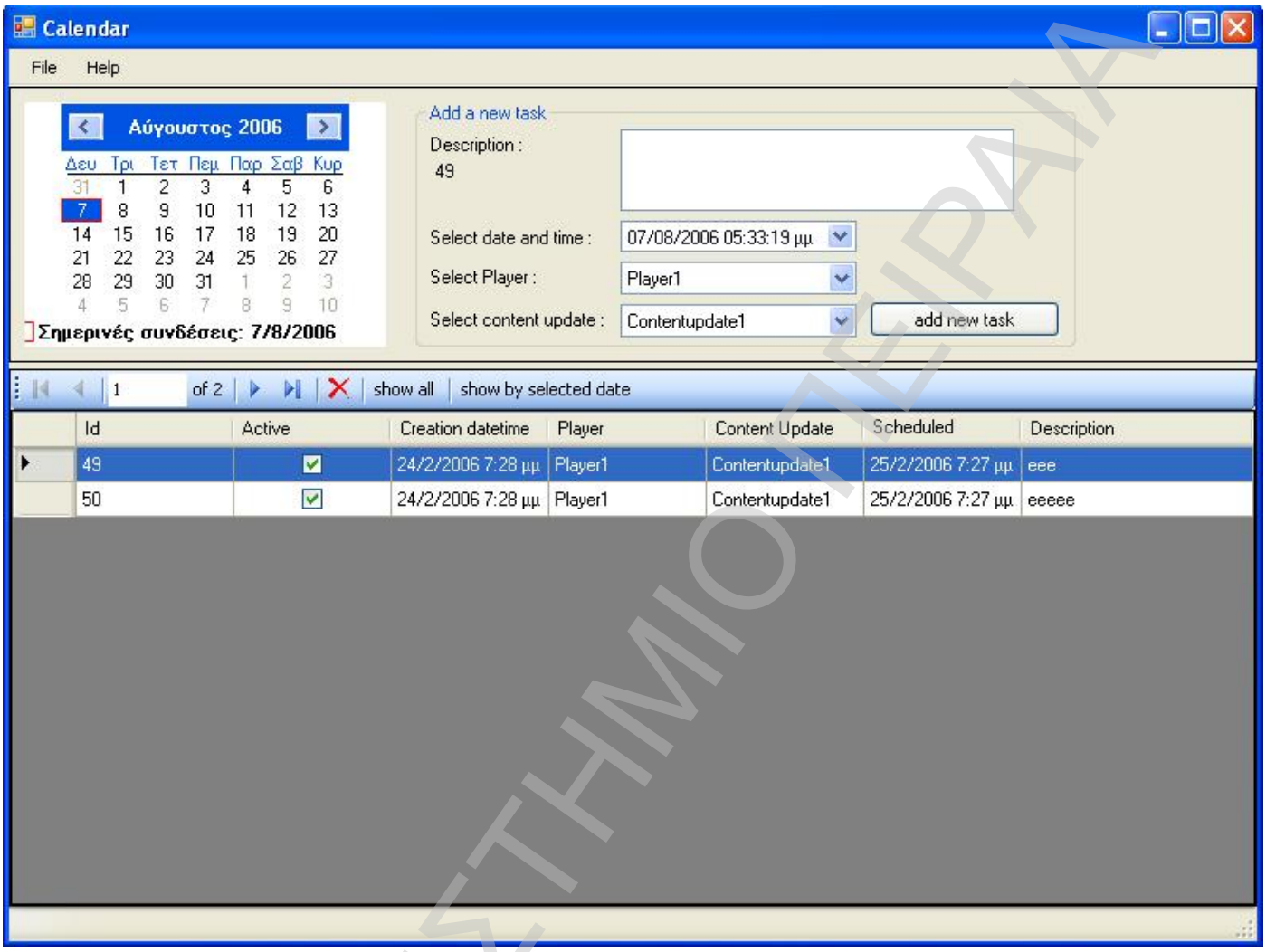


Figure 13-21: *Calendar - user interface*

Figure 13-21 presents the user interface of the calendar application. Calendar could have been a component of Network Manager application, but we preferred to develop it as a separated stand alone application. Calendar allows user to select an existing Player and one of its available Content Updates in order to schedule a future update on a desired future date. When user selects a date of the past, the application informs him to reselect another date (error control).

Programmatically the user entry is stored to the central database. Schedule Server then will be responsible to trigger the update at the right moment.

Beside the “add new task” capability, user can also monitor all previous scheduled tasks. He can check which of them are active or inactive for a specific day, period of days or from the whole of entries.

13.1.3 Schedule Server

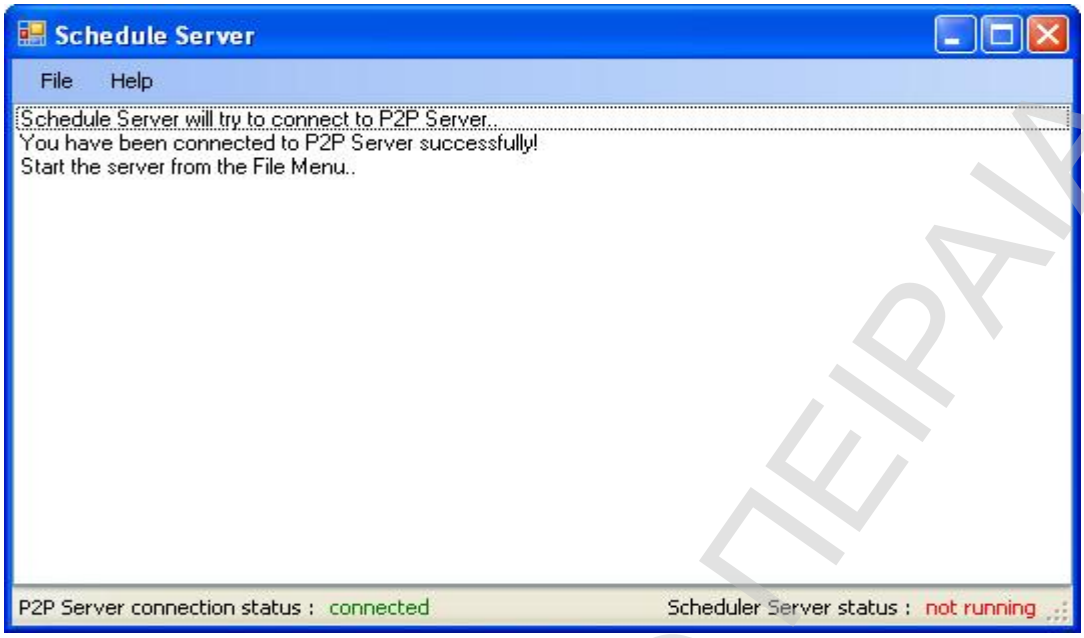


Figure 13-22: *Schedule Server – user interface (a)*

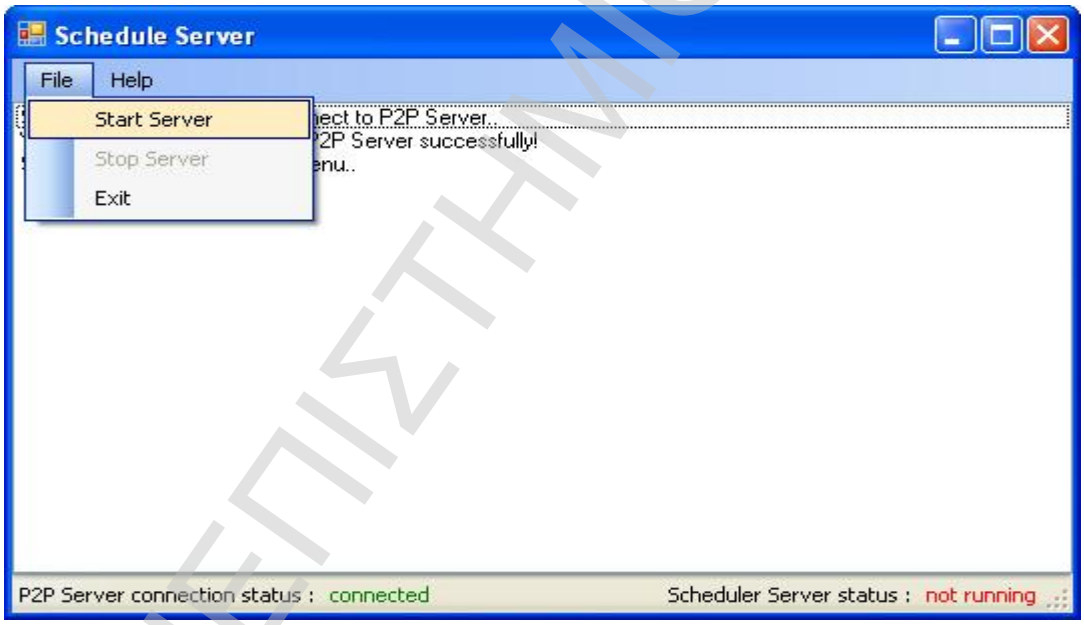


Figure 13-23: *Schedule Server – user interface (b)*

Figures 13-22 and 13-23 present the user interface of the Schedule Server. Schedule Server requires P2P Server. If P2P Server is not running, the user interface will not allow user to start Schedule Server. As you can see the status bar informs user about both the P2P Server connection status and the Schedule Server status. In case of failure on P2P Server connection, Schedule Server will automatically be stopped and will try to reconnect with P2P Server. All the activity is recorded in the status panel so that the user can understand what is going wrong and take action at the appropriate time.

13.1.4 P2P Server

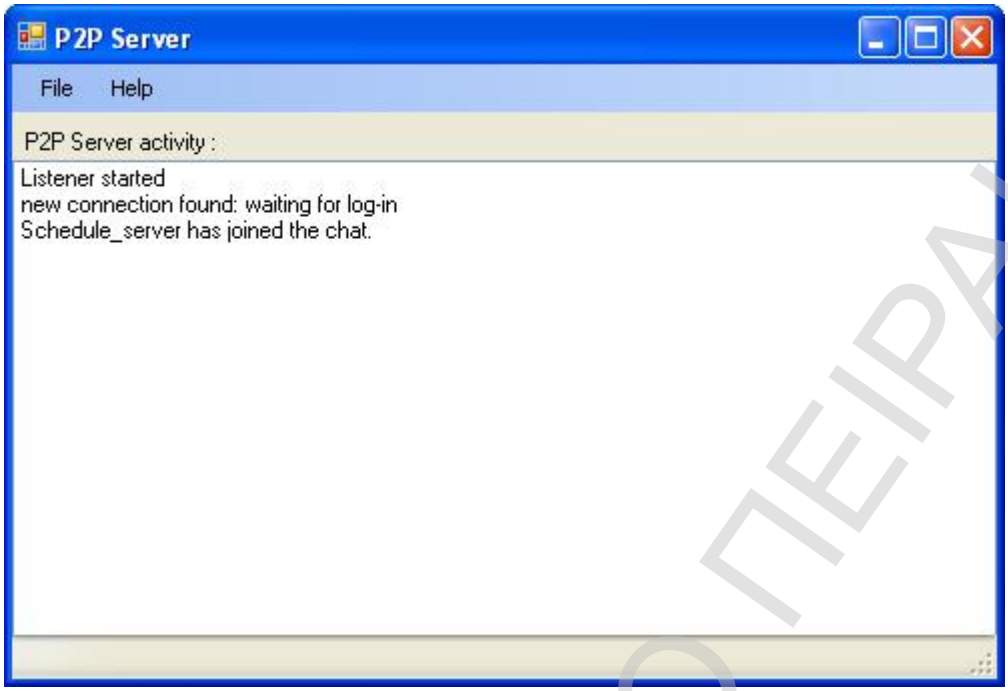


Figure 13-24: *P2P Server – user interface*

Figure 13-24 presents the minimal user interface of the P2P Server. P2P Server could have been implemented as window service. We implemented as window form so that we can monitor server's activity. P2P Server's activity panel records all the activity about status, connections, messages etc.

13.1.5 Players

A number of Players have been implemented during project's development. Some of them implemented with minimal functionality and others with more complex one, in order to test project's requirements and capabilities. Finally we decide to implement a Player that integrates a number of capabilities and a scalable functionality.

The scenario is based on a DVD club. Both the output and the input devise will be the glass window of the DVD shop. A USB membrane will stimulate the mouse and will turn the glass to a touch screen. A projector will turn the glass to a rear projection screen.

Client/user can step over the shop-window and browse the application. The service will be available 24 hours per day, 7 days per week (24/7).

The implemented Player (Player2.exe):

- On start will try to connect with P2P Server. If the connection could not be established will try again, till it succeeds.
- Can give useful information about store's history, services, timetable, items prices, location, contact information etc
- Allows user to preview trailers of films
- Allows user to become a member of the guest list so that the store can communicate with him (send him offers, news)
- Allows administrator remotely change the user interface (background, logos, images, text, buttons)
- Allows administrator remotely remove or add services (by setting the visibility of the buttons or by updating the software version)
- Allows administrator to change trailers that are available for previewing
- Allows administrator to change information content
- This is a demo Player and it includes a panel with detailed information about the Player, the current software version, the current content update. The messages that player receive are also monitored. Any trigger for software update or content update will be monitored too. Finally activity panel of the Player records every error and failure.
- Updates Network Manager's Activity Panel, so that the administrator can remotely understand if an action completed successfully or failed

In this section we will present just one screen shot of the Player (Figure 13-25). More screen shots that will show the whole functionality of the Player will be presented on the next chapter (13.2).

Figure 13-25: *Player – Player2 user interface*

14 Project in Action

In this chapter we will give an example of using the whole project. We will cover the functionality of all system components from the creation of Player application till the Player in action. Screen shots will be presented according to the logical timeline.

14.1 Designer/Developer implements the first software update of a Player (Player2)

After developing the application of the Player, developer has to prepare application for deployment. The developer will choose to publish Player's application via a web server. During this procedure he will be prompt to give a version name, to specify web server's URL, to declare the prerequiresite components (.NET Framework 2.0, SQL Server Express Edition), and to edit the properties that are available and determine the details for publishing.

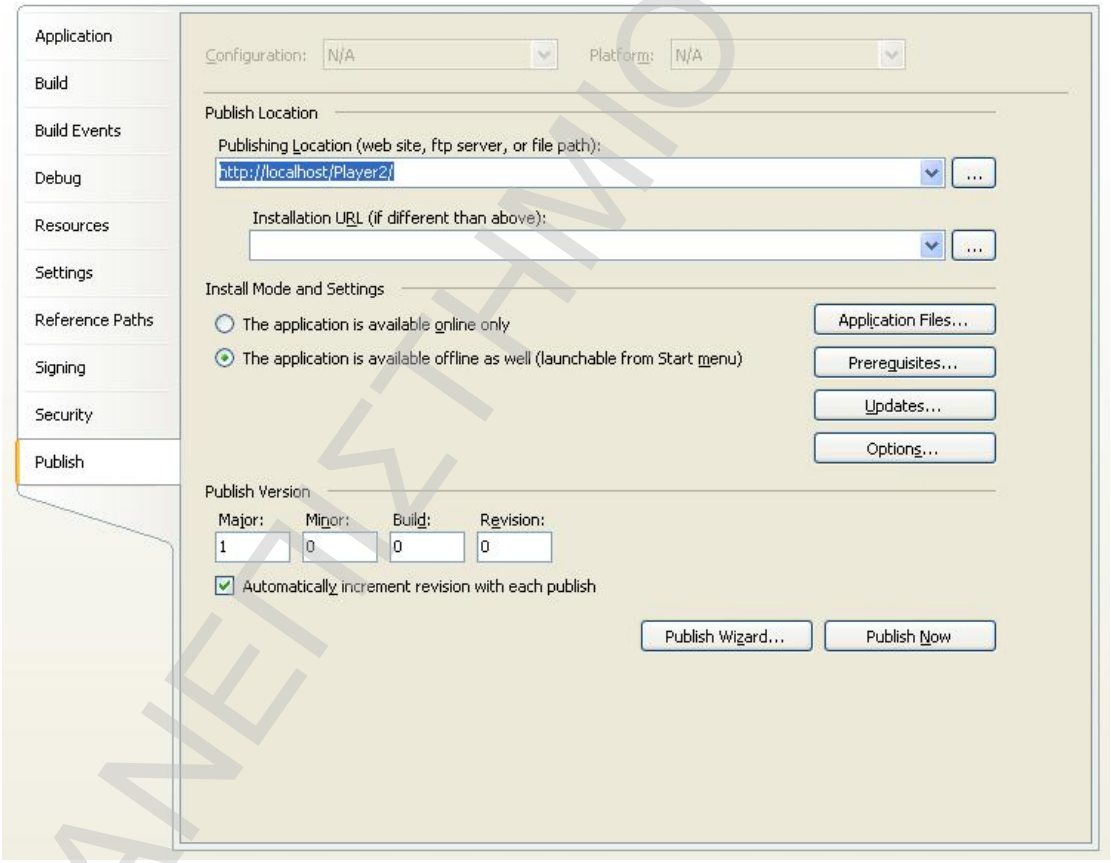


Figure 14-1: *Publishing Player's application (a)*

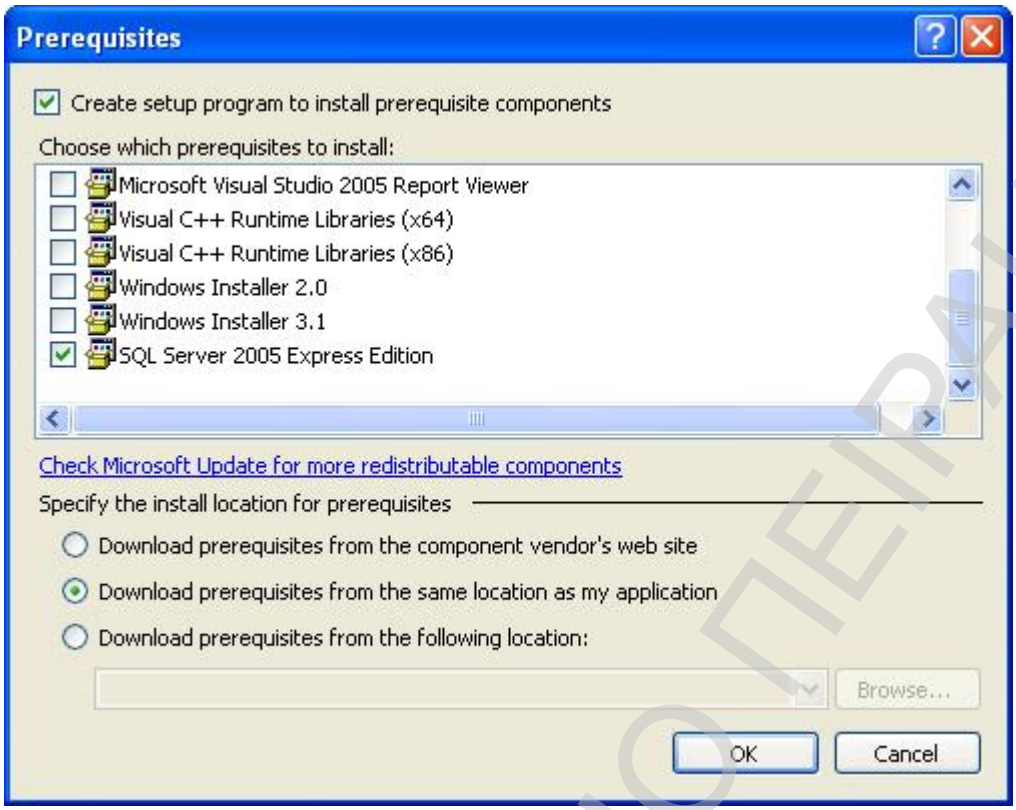


Figure 14-2: *Publishing Player's application (b)*

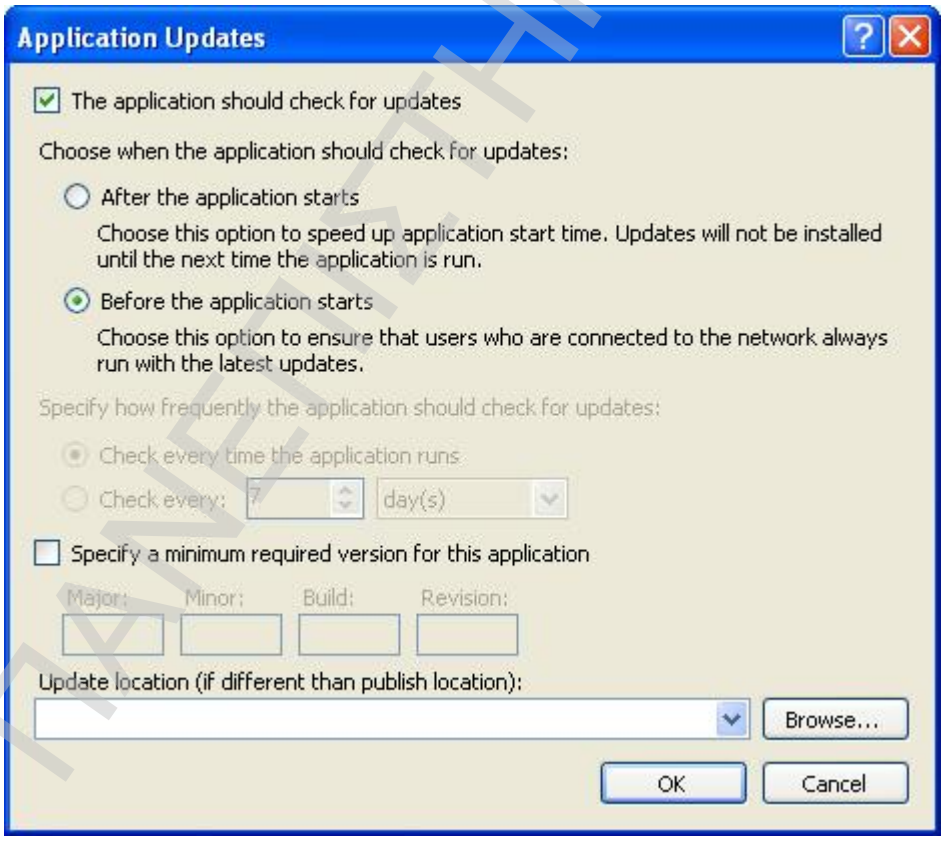


Figure 14-3: *Publishing Player's application (c)*

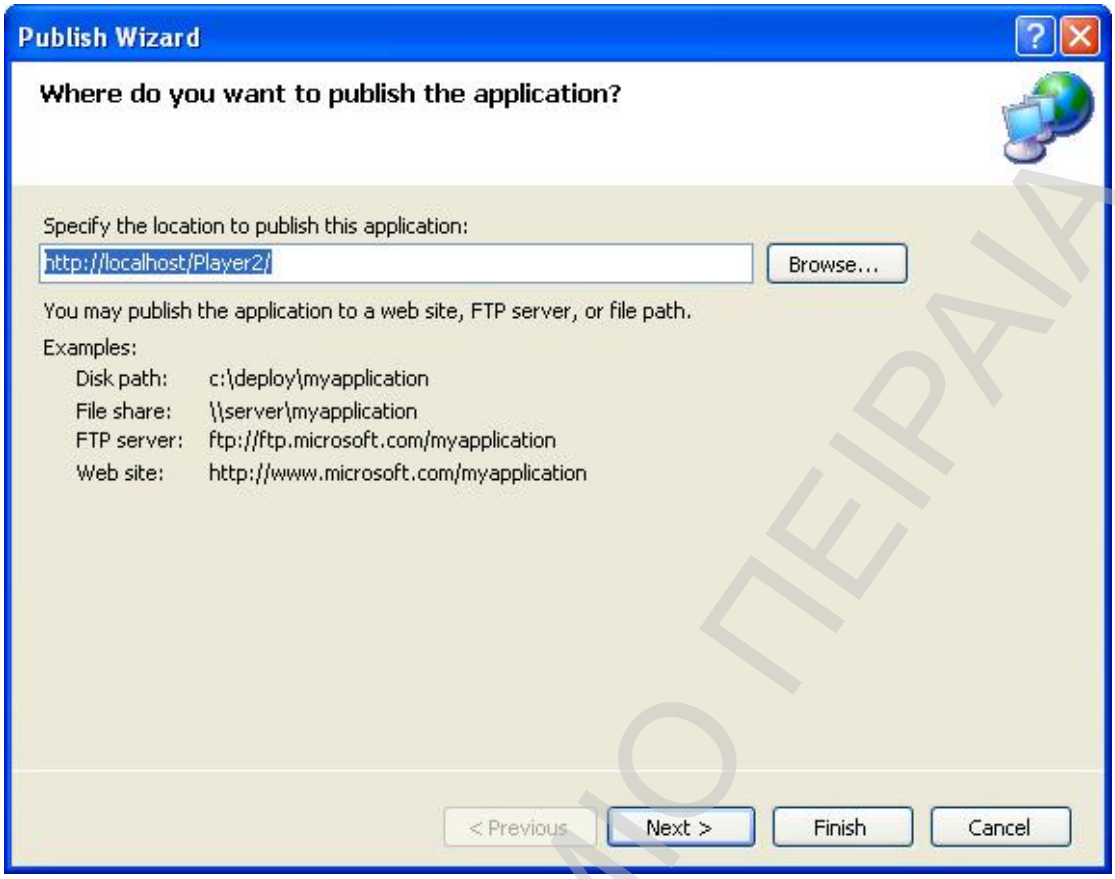


Figure 14-4: *Publishing Player's application (d)*

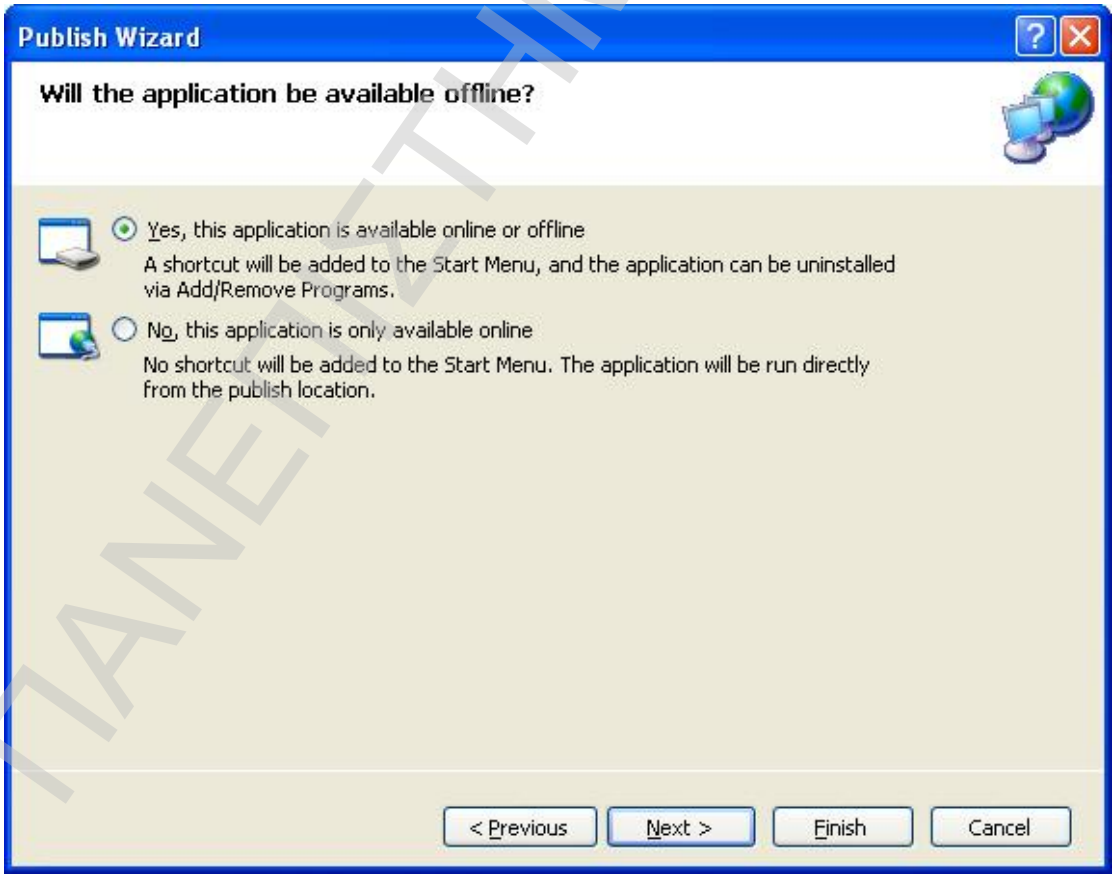


Figure 14-5: *Publishing Player's application (e)*

14.2 Network Manager's administrator adds the new Player (Player2)

Before publishing the software for the Player Network Manager's administrator should add the new Player (Player2) and the first software version. Then he can edit the default content update and add as much content updates he desires. For our example we will create Player2, add the first software version 1.0.0.0, then we will edit the default content update (Contentupdate1) that have automatically created during the procedure of adding the software version and finally we will create a second content update (Contentupdate2). All these actions are required the full demonstration of all system components.

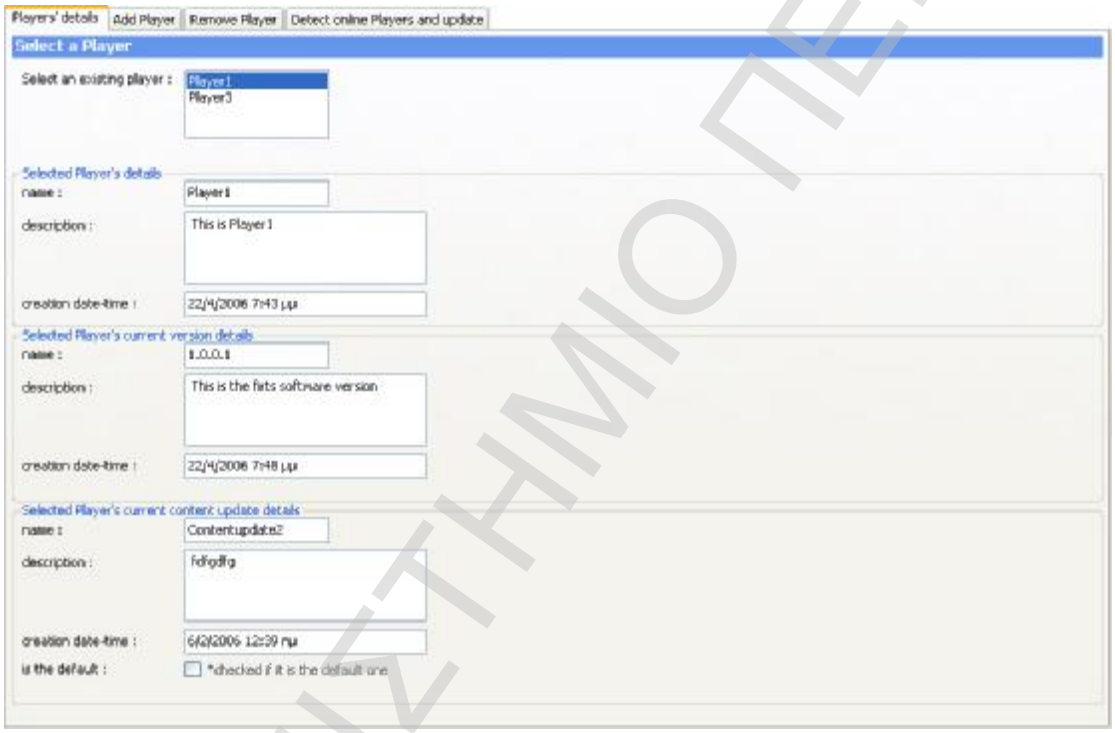


Figure 14-6: *Network Manager - "Players" page's "Players' details" panel before the creation of the new Player (Player2)*

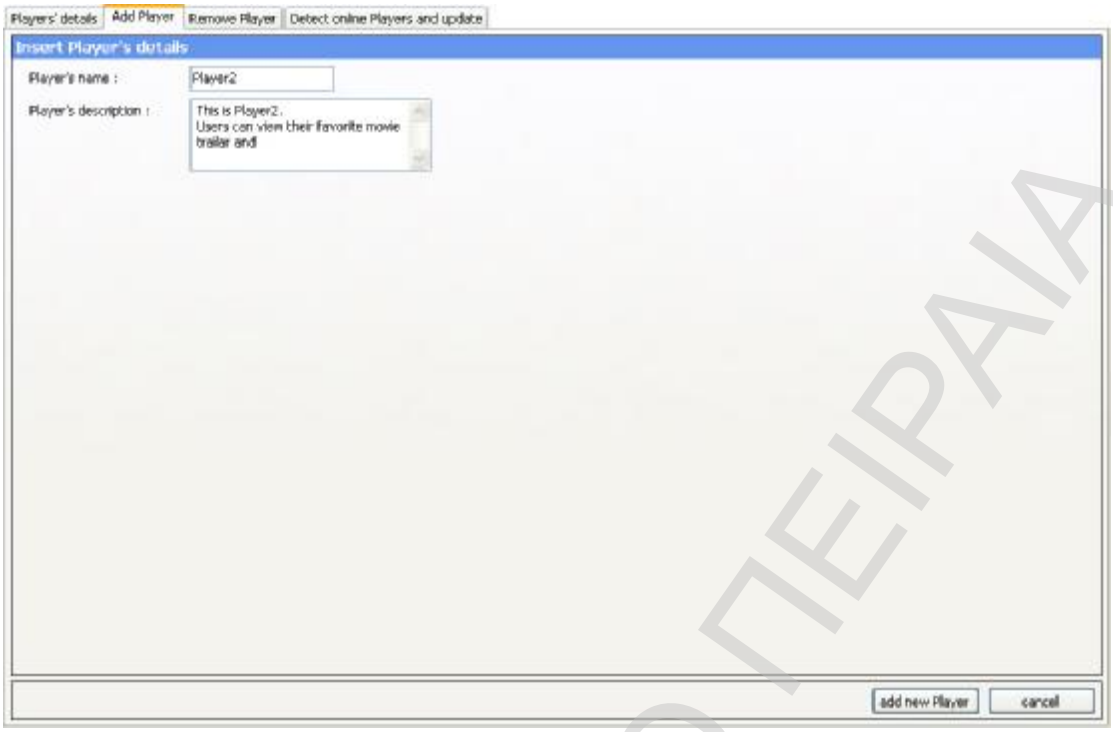


Figure 14-7: *Network Manager - "Players" page's "Add Player" panel. Adding Player2*

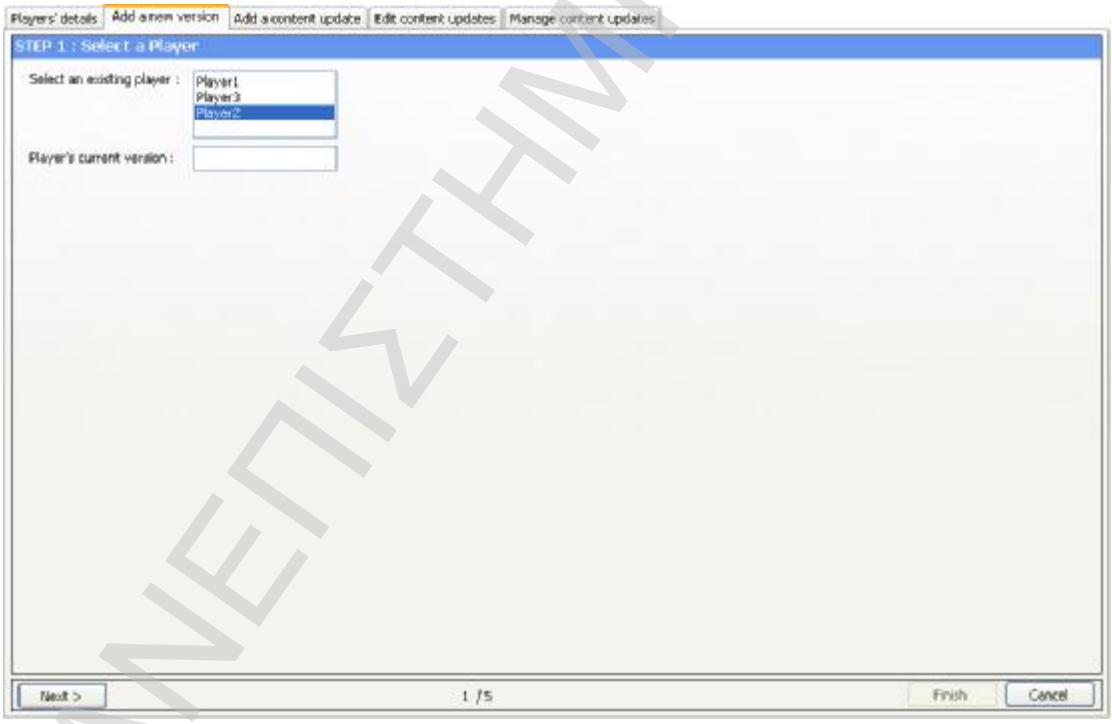


Figure 14-8: *Network Manager - "Jobs" page's "Add a new version" panel. Adding software version 1.0.0.0 of Player2 (1/5)*

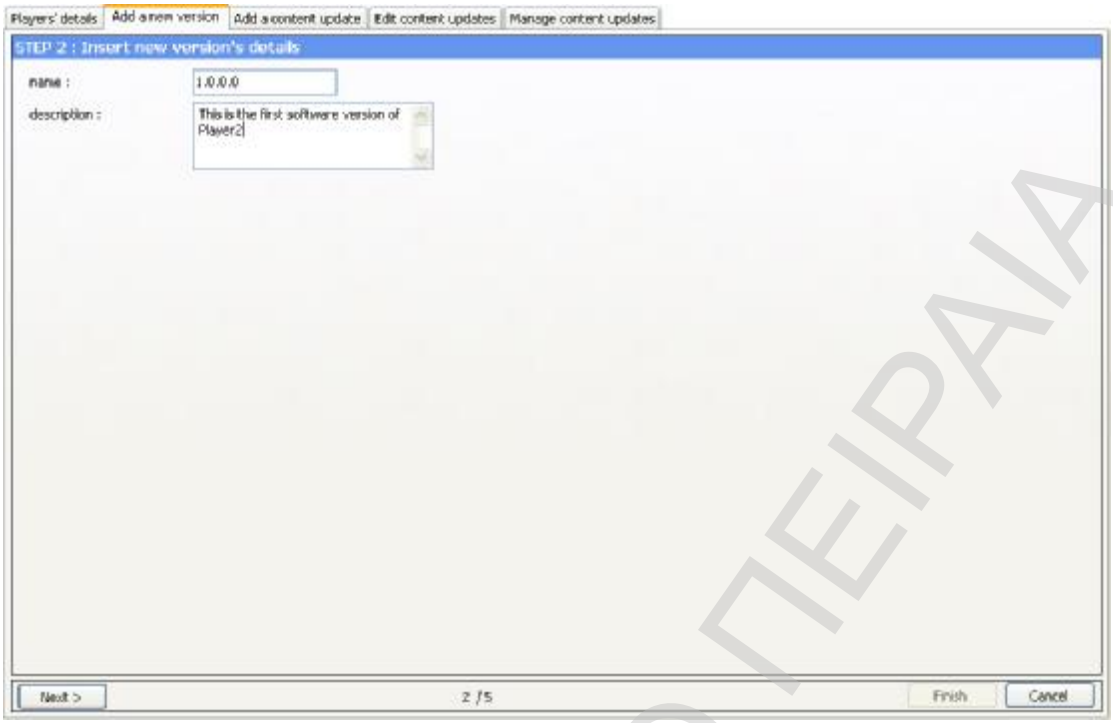


Figure 14-9: *Network Manager - "Jobs" page's "Add a new version" panel. Adding software version 1.0.0.0 of Player2 (2/5)*

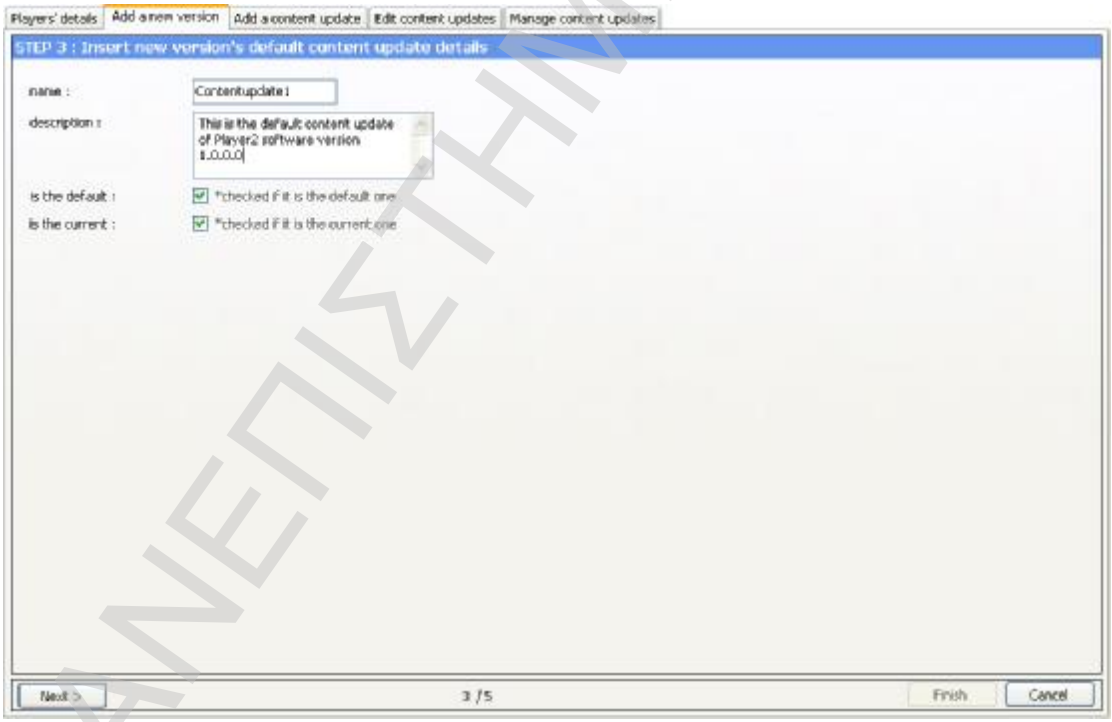


Figure 14-10: *Network Manager - "Jobs" page's "Add a new version" panel. Adding software version 1.0.0.0 of Player2 (3/5)*

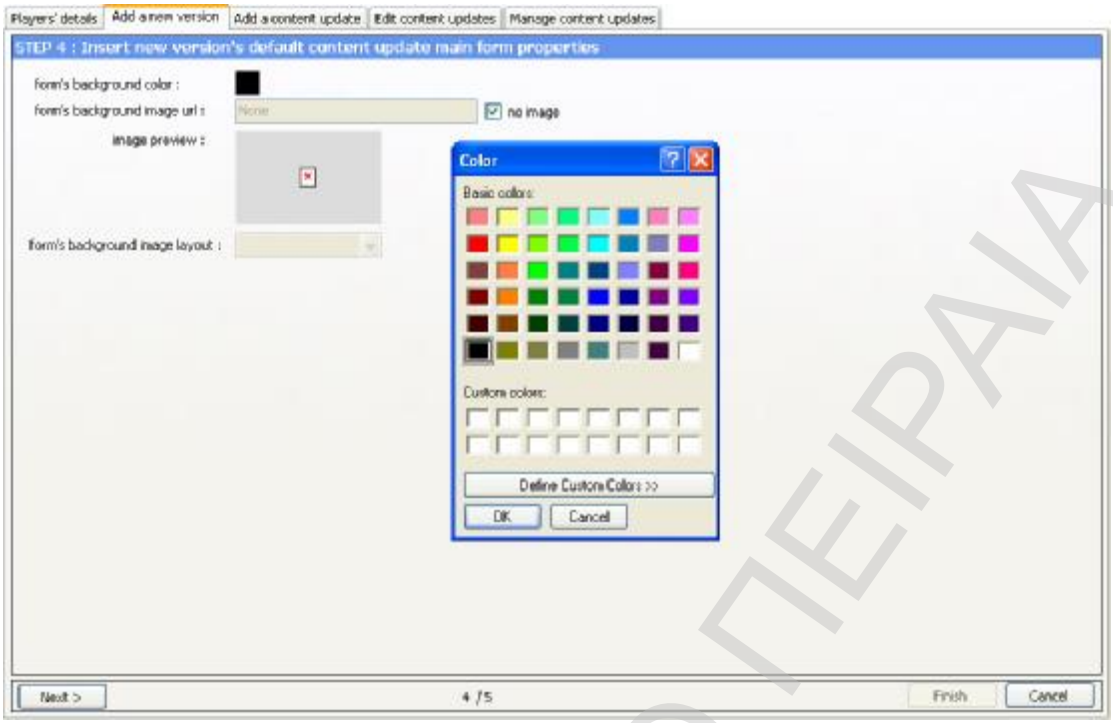


Figure 14-11: *Network Manager - "Jobs" page's "Add a new version" panel. Adding software version 1.0.0.0 of Player2 (4/5)*

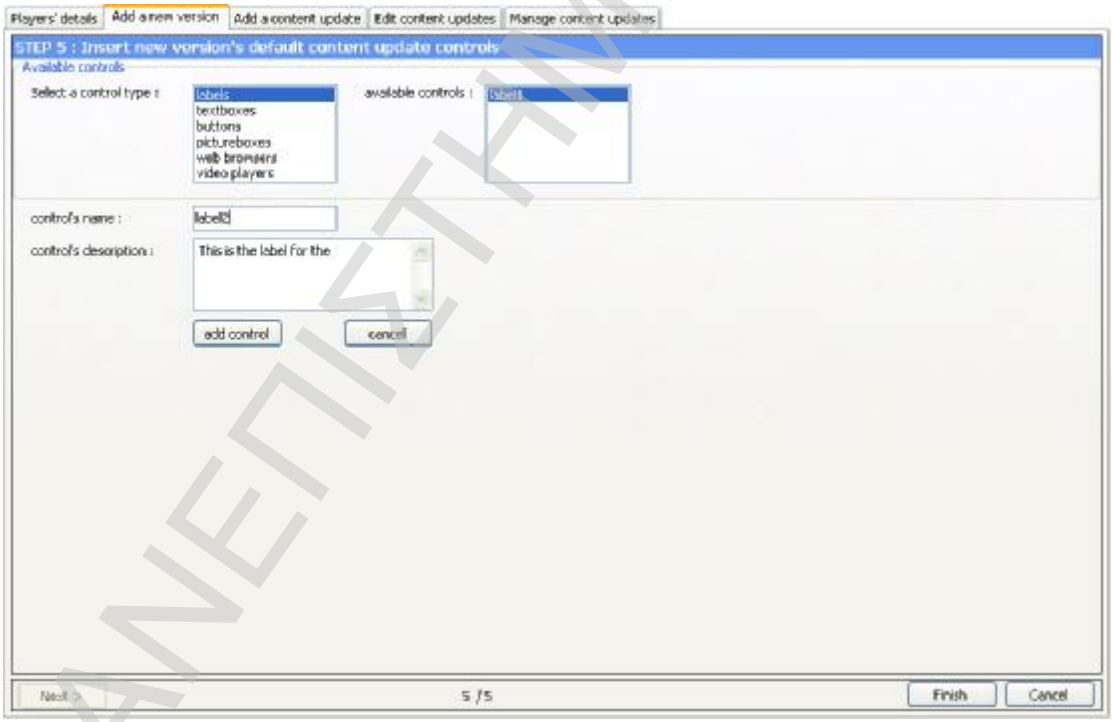


Figure 14-12: *Network Manager - "Jobs" page's "Add a new version" panel. Adding software version 1.0.0.0 of Player2 (5/5)*

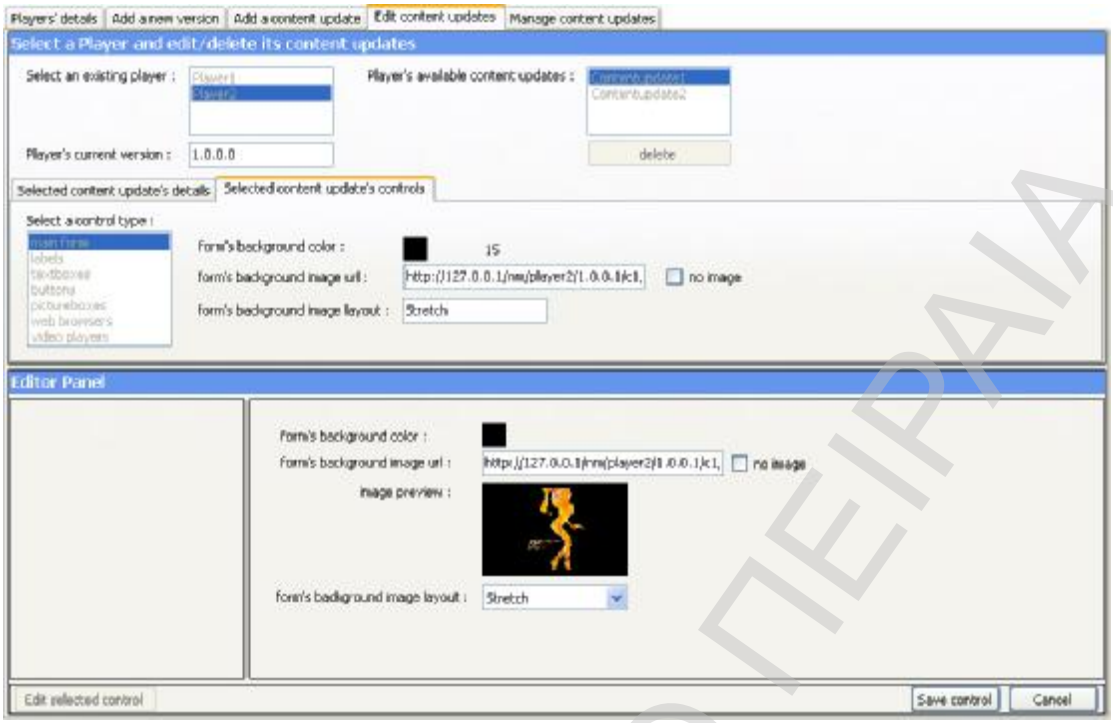


Figure 14-13: *Network Manager - "Jobs" page's "Edit content updates" panel. Editing the default content update (Contentupdate1) of Player2 software version 1.0.0.0 (1/2)*

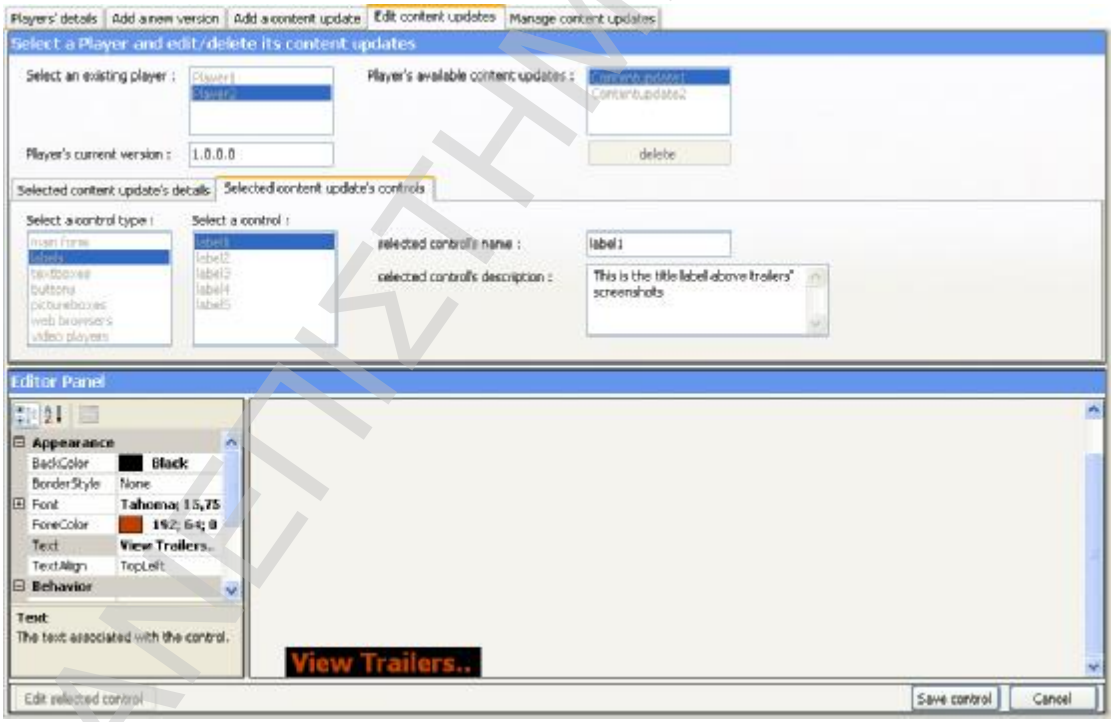


Figure 14-14: *Network Manager - "Jobs" page's "Edit content updates" panel. Editing the default content update (Contentupdate1) of Player2 software version 1.0.0.0 (2/2)*

Figure 14-15: *Network Manager - "Jobs" page's "Add a content update" panel. Adding the second content update (Contentupdate2) of Player2 software version 1.0.0.0*

After adding the second content update (Contentupdate2) we have to edit it. Adding a content update programmatically means the generation of a replica of the default content update.

14.3 Installation of Player2's software (software version: 1.0.0.0)

After publishing Player's application and registering it with Network Manager, The software is ready to play. In chapter 14.1 we saw that developer have chosen publication via a web server. This means if we want to install the software to a host, we have to browse to the specified URL. Figure 14-16 shows the user interface of the web site from which we are able to install Player2's software.

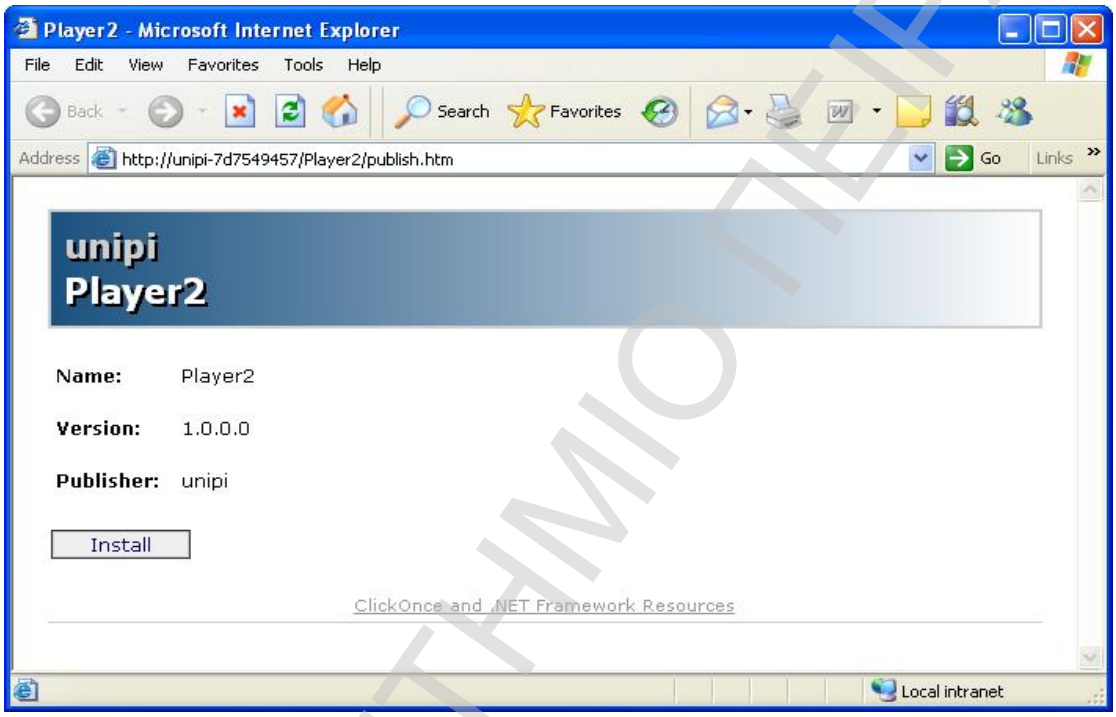


Figure 14-16: *User interface of the web site from which we are able to install Player2's software*

Figure 14-17: *Player2's software installation*

14.4 Starting / Using Player (Player2)

After installing the software we can run the application like any other program of Windows operating system (Figure 14-18).

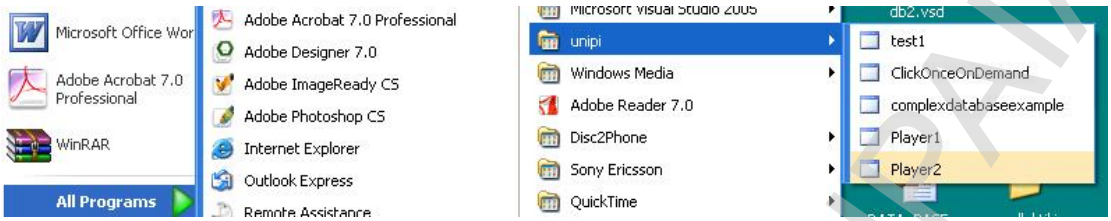


Figure 14-18: *Starting Player2 by browsing Windows' "Start" menu*

Figures 14-19 – 14-22 shows a part of the basic functionality of the Player (software version 1.0.0.0 content update: Contentupdate1). Among contents updates functionality remains as is, but this does not happen after an upgrade of software version.

We should remind that Player2 is a demo version of Player. Using black as background color will cause transparency if the output devise is a rear projection screen.

Figure 14-19: *Player2 1.0.0.0 Contentupdate1 : Basic user interface*

Figure 14-20: *Player2 1.0.0.0 Contentupdate1 : Pressing "Who" button*

Figure 14-21: *Player2 1.0.0.0 Contentupdate1 : Pressing "Where" button*

Figure 14-22: *Player2 1.0.0.0 Contentupdate1 : Pressing "Join guest list" button*

In the bottom – left corner we have add a button "details". This button should exist only in a demo Player. This button shows an activity panel which informs us about the activity. For example, when we start Player2, the application will try to connect to P2P Server. Figure 14-23 and 14-24 shows P2P Server before and after Player2 is connected. Figure 14-25 shows changes in the activity panel of the Player2.

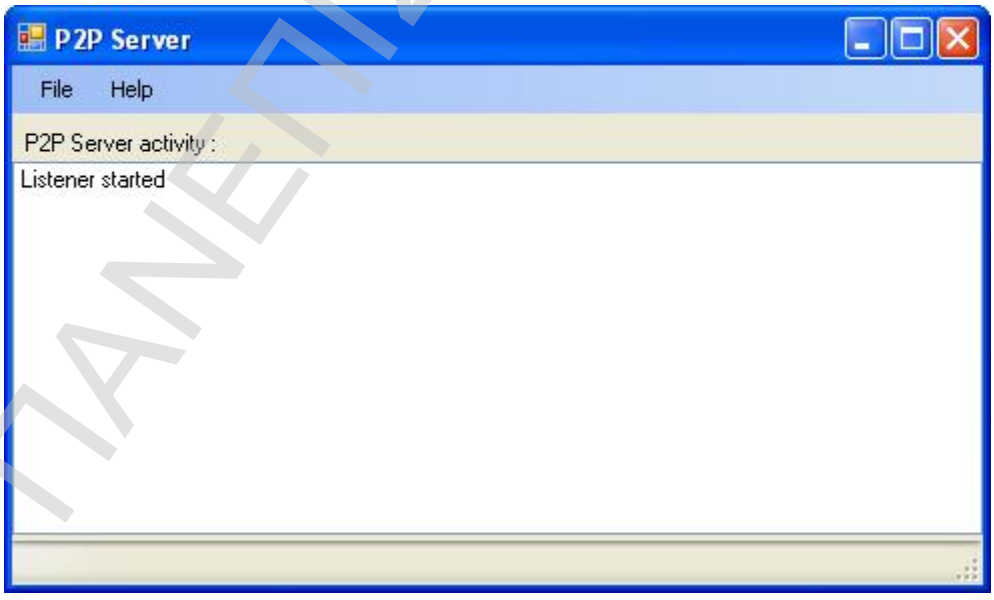


Figure14-23: *P2P Server : Player2 hasn't been connected yet*

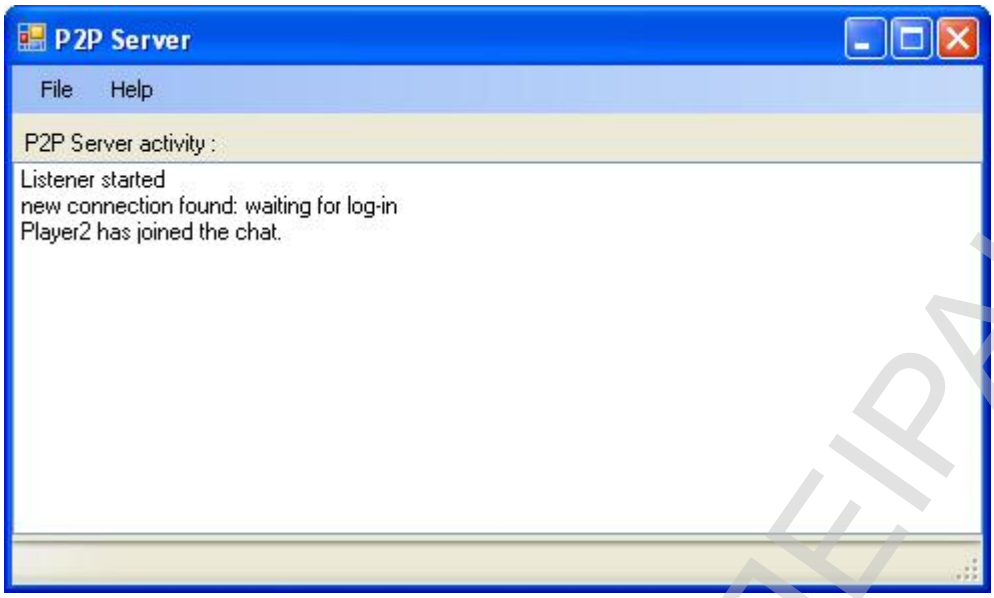


Figure 14-24: *P2P Server : Player2 is now connected*

Figure14-25: *Player2 1.0.0.0 Contentupdate1 : Activity panel after connecting to P2P Server*

14.5 On demand real-time update on content update on Player2 (from ContentUpdate1 to ContentUpdate2)

Changing Content Update of Player2 requires just two “clicks” in Network Manager’s application. These “clicks” trigger a number of actions in different components of the system. Figure 14-25 and 14-26 show the required actions of the administrator. First he should select the desired content update of Player2 as current and then he has just to connect with P2P Server and trigger the update. Figure 14-27 shows P2P Server’s activity panel, Figure 14-28 shows Player2’s activity panel and finally Figure 14-29 shows the result changes on the Player’s User Interface. Everything is now different, except from functionality. In addition, this activity has been recorded in “activity” page of Network Manager (Figure 14-30).

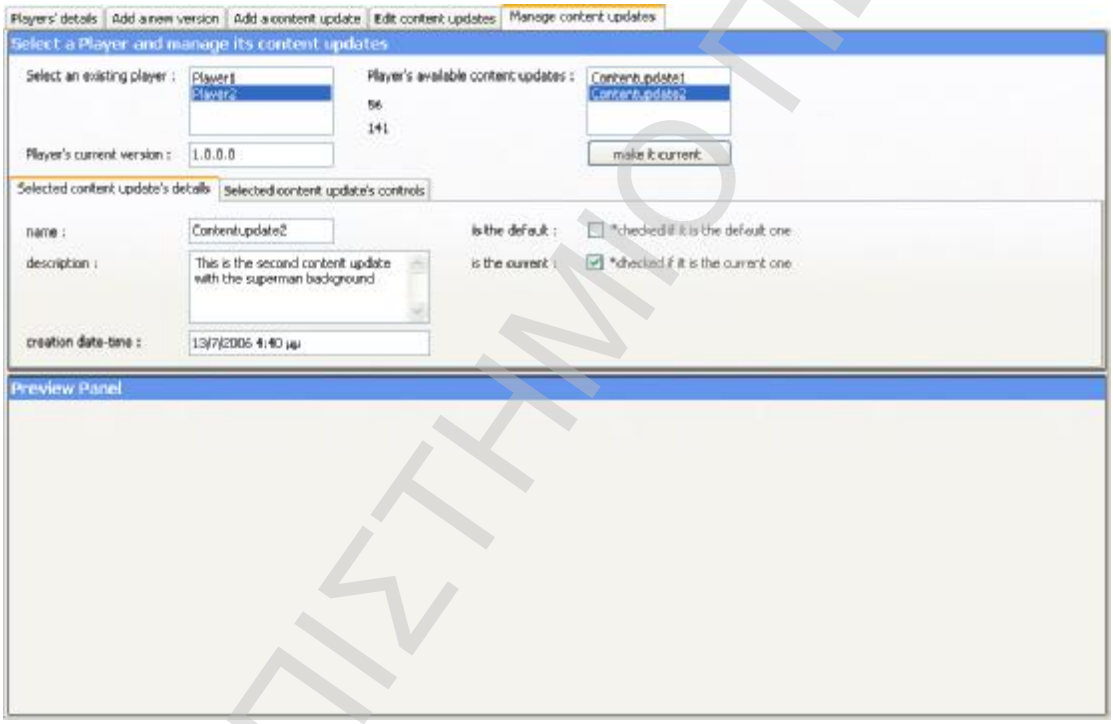


Figure 14-26: *Network Manager - "Jobs" page's "Manage content updates" panel*

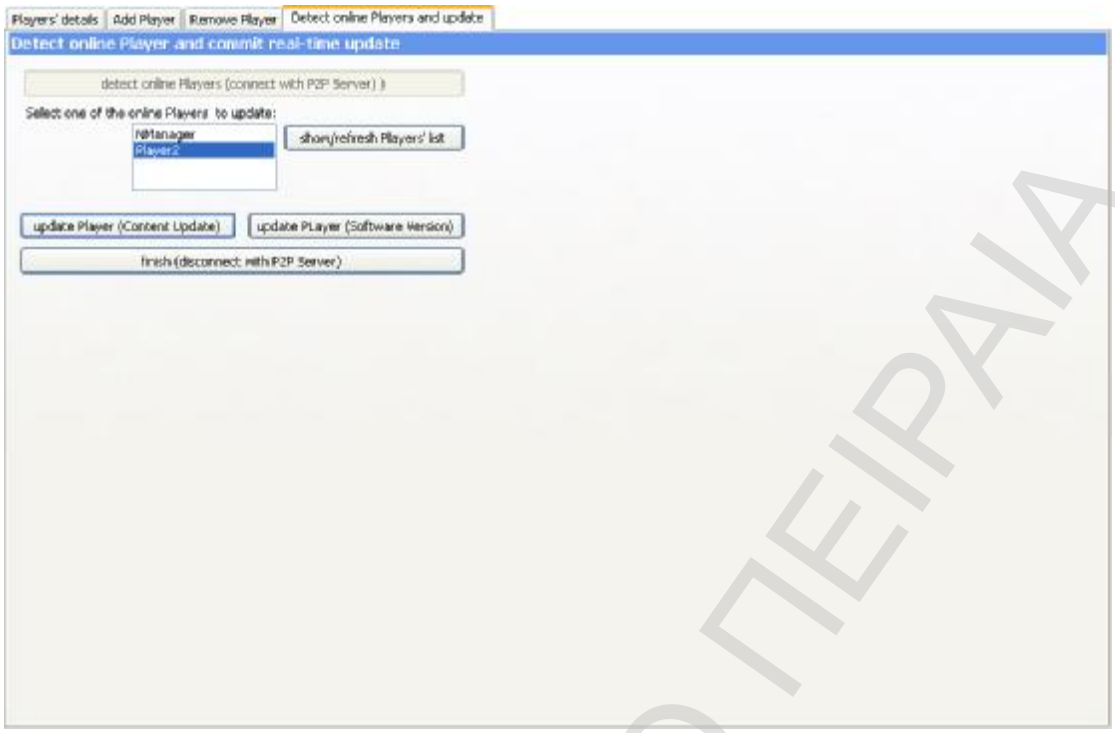


Figure 14-27: *Network Manager - "Players" page's "Detect online Players and update" panel*

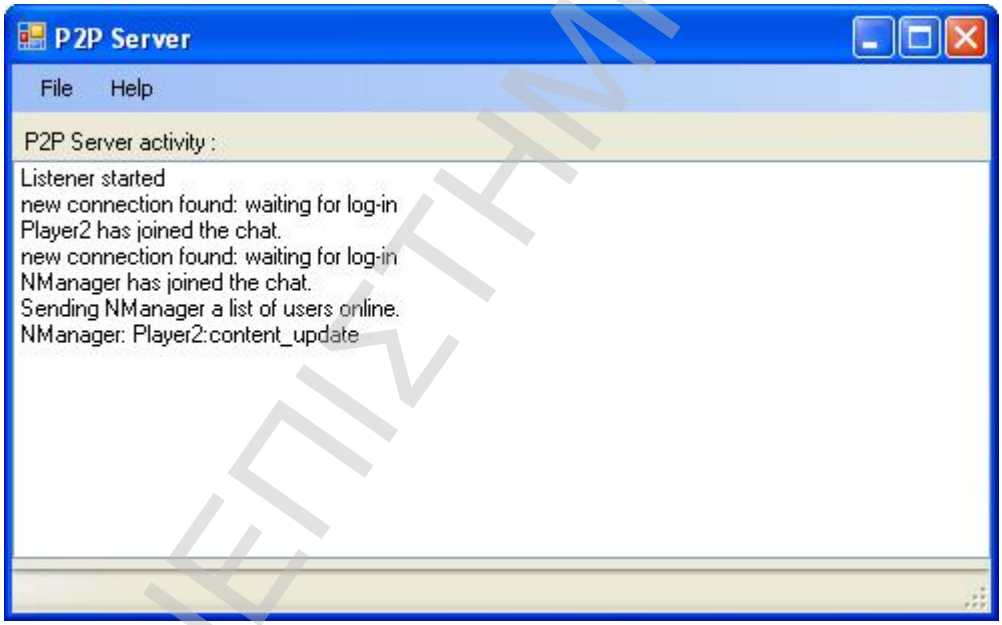


Figure 14-28: P2P Server – activity panel

Figure 14-29: Player2 – activity panel

Figure 14-30: *Player2 1.0.0.0 Contentupdate2: Basic user interface*

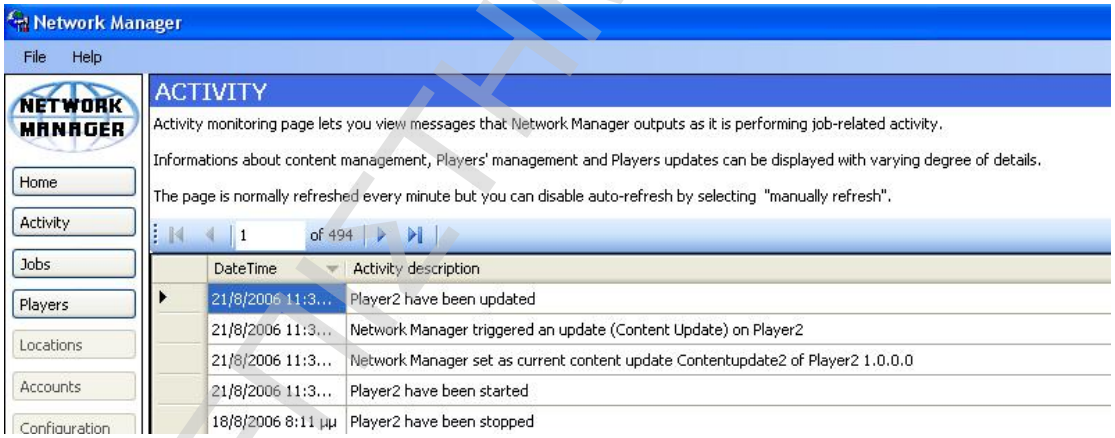


Figure 14-31: *Network Manager - "Activity" page*

14.6 Edit on content update (Contentupdate2) of Player (Player2) and real-time update

Supposing Player2 application is running (Figure 14 32) and Network Manager's administrator wants to change the label "View Trailers.." to "View Movie Trailers..". He should browse Network Manager's application to "Jobs" page and select "Edit content updates" panel. Then he has to select the label and make the appropriate changes (Figure 14-33). Finally he has to connect to P2P Server and trigger the update (Figure 14-27). Figure 14-34 shows the result on the user interface of the Player.

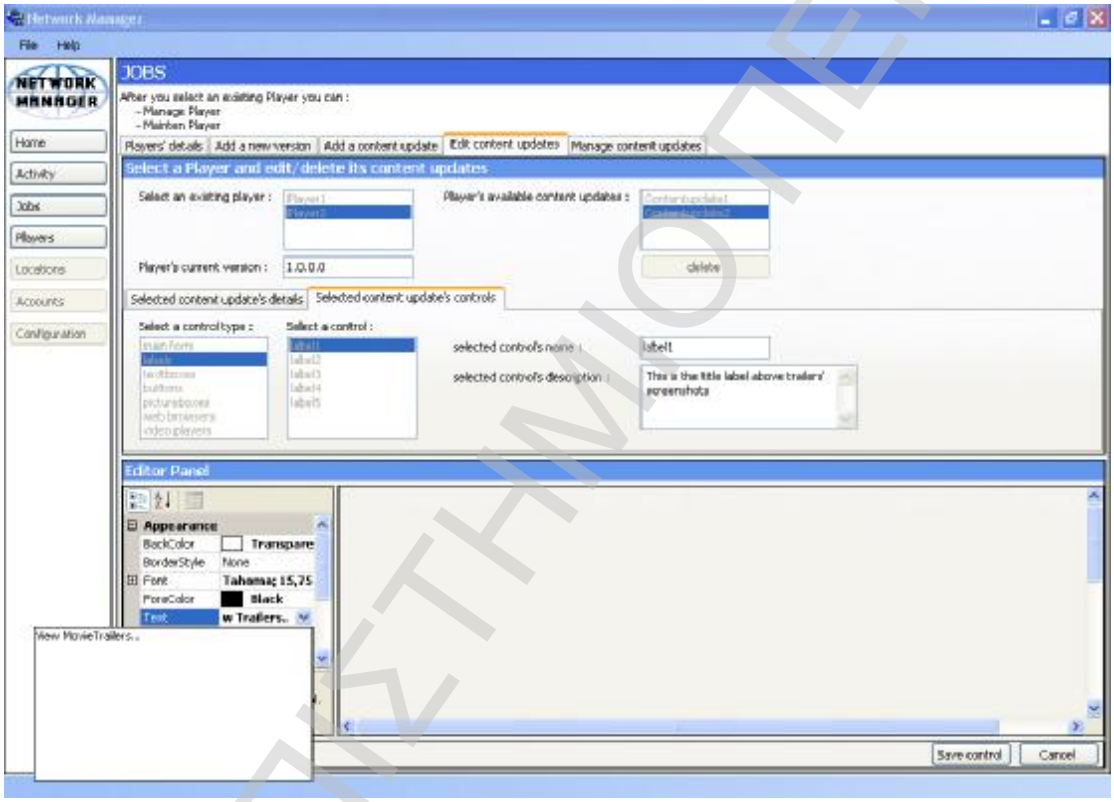


Figure 14-32: *Network Manager - "Jobs" page's "Edit content updates" panel: editing label1*

Figure 14-33: *Player2 1.0.0.0 Contentupdate2: Basic user interface after editing label1*

14.7 Scheduled real-time update on content update on Player2 (from Contentupdate1 to Contentupdate2)

Supposing Player2 is running with Contentupdate1. Network Manager's administrator wants to schedule an update on content update (from Contentupdate1 to Contentupdate2). First he has to ensure that both P2P Server and Schedule Server are running. Then he has to open the Calendar application. After he has to select Player2 among players, then he has to choose the desired content update and finally to select the date-time for the update. Figure 14-34 presents the user interface of the calendar. The scheduled task will be saved in the database and Schedule Server will commit the update at the right time.

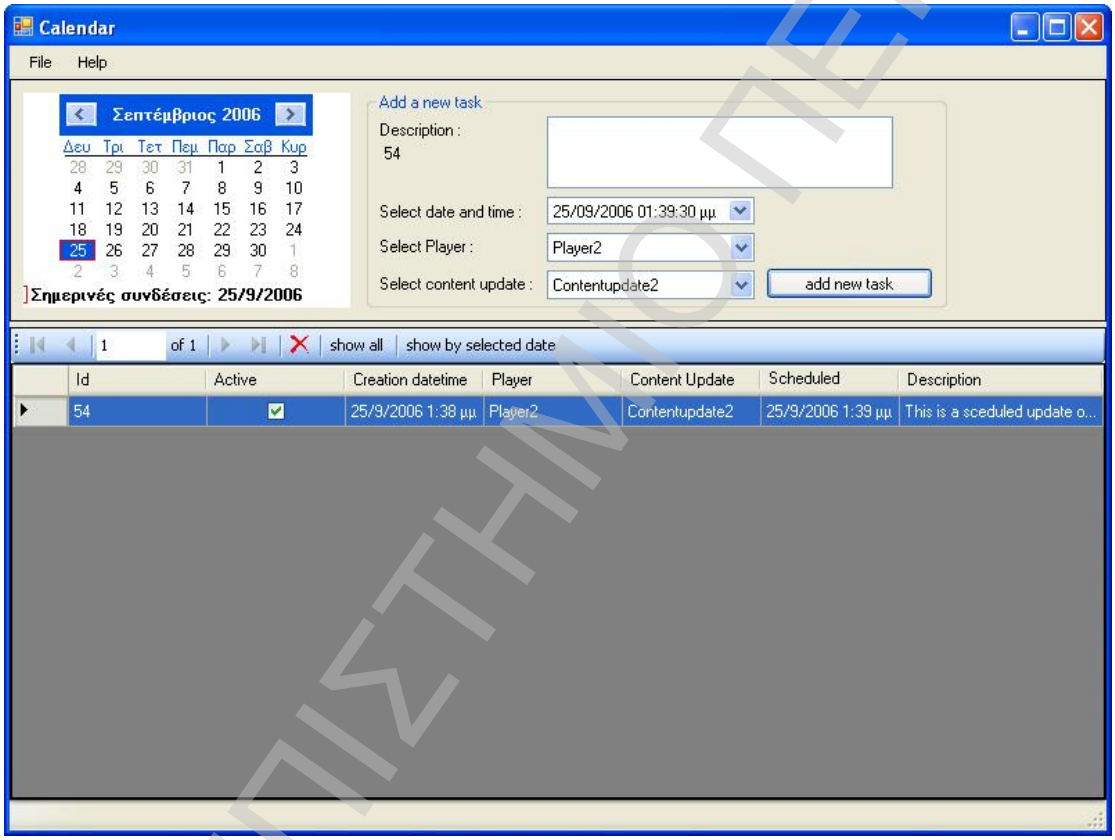


Figure 14-34: *Calendar application: The update is now scheduled*

15 Conclusion

After more that one year of research and development, a fully functional system was created. The system should not be considered as commercial product. The first intention was to create a prototype with basic functionality. The creativeness and novelty of the project were the inspiration point that leaded in an enrichment solution. The implementation will be useful for a starting point of a commercial product. Documentation, functionality, user interface and architectural components and technology tips can be adopted in a more sophisticated and accomplished solution.

16 References

[1] Mattern, F. and Naghshineh, M. (ed) (2002) Pervasive Computing. Berlin, Heidelberg: Springer Verlag.

[2] Pervasive and Ubiquitous Public Map Displays, Michael P. Peterson, Department of Geography/Geology, University of Nebraska at Omaha Omaha, 2004

[3] Grand Challenges in Computer Science and Engineering, U.S. Initiative. Computing Research Association; www.cra.org/reports/gc.systems.pdf.

[4] Information Society Technologies Advisory Group. Working Group Grand Challenges in the Evolution of the Information Society. July 6, 2004, (ftp://ftp.cordis.lu/pub/ist/docs/istag_draft_report_grand_challenges_wahlster_06_07_04.pdf)

[5] IST-FET: EU- funded proactive initiative The Disappearing Computer, (www.disappearing-computer.net)

[6] ISTAG Report on Scenarios for Ambient Intelligence in 2010, (ftp.cordis.lu/pub/ist/docs/istagscenarios2010.pdf)

[7] Weiser, M. The computer for the 21st century. *Scientific American* (Sept. 1991), 94–104

[8] Norbert Streitz AND Paddy Nixon, March 2005/Vol. 48, No. 3 COMMUNICATIONS OF THE ACM

[9] Rui José, Beyond Application-Led Research in Pervasive Display Systems, 2005

[10] Scala White Paper: How InfoChannel Works – A bridged By: Richard F. Trask, Marketing Director, Scala Inc. 2006

[11] WireSpring, An Introduction to Digital Signage, 2005

[12] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja1, Jim Pruyne, Bruno Richard, Sami Rollins 2 ,Zhichen Xu, Peer to Peer Computing, 2002 (HP)

[13] Microsoft Patterns and Practices : Smart Clients architecture and design goals

[14] MSDN Library for visual studio 2005

[15] http://www.theserverside.net

[16] http://www.cimtec.dk/dot-net

[17] http://www.windowsforms.net

[18] http://dotnetjunkies.com

[19] http://www.ondotnet.com

[20] http://www.codeproject.com

[21] http://msdn.microsoft.com/smartclient/

[22] .NET Show: Episode: NETShow45_300k.wmv (Video about Smart Clients' implementations and techniques)

[23] http://www.scala.com

[24] Microsoft application Blocks for .NET:

- Ø Data Access V2
- Ø Offline
- Ø User Interface process 2.0

[25] Microsoft Patterns & Practices :

- Enterprise Library
- Enterprise Library June 2005
- Updater Application Block V2
- Composite UI Application Block December 2005 C#

[26] Scala Infochannel Designer 3 Trial (Software for Digital Signage)

[27] http://www.elotouch.com