AAL Joint Programme

D.2.1 Technical specification of WeCare 2.0 Service

Due date of deliverable:	December 11th 2010
Actual submission date:	November 2nd 2010
Date of revised edition:	March 10th 2012,
Editorial update:	September 31st 2012

Start date of project:	11 February 2010
Duration:	30 months
Lead contractor for this deliverable:	Paul Kiernan, Skytek
Name of the lead coordinator person:	Stefan Burgers, Ericsson
Contact details of the coordinator:	Stefan.Burgers@ericsson.com +31161242654

Confidentiality status:	Restricted to bodies determined by the AAL WeCare project

Technical specification of WeCare 2.0 Service

Abstract:

The scope of this document is to present the architectural design and component description for the WeCare system.

The project's key goal is to prevent elderly people from becoming lonely through the provision of a support structure by participating in social networks. The goal is to support and empower senior people to participate in social networks as early as possible, before they start to suffer from ill health, isolation or loneliness. When or if people start to suffer from ill health, these social networks are already 'in place' and can function beneficially. The objective is to empower senior people to engage in active, two-way relationships of receiving help and offering help (inter-dependency), as much as possible and for as long as possible. Family, friends and neighbours create 'connected groups' of mutual care: receiving care and providing care to each other.

Scope:

Technical

Impressum

Full project title: AAL WeCare

Title of the workpackage: Technology development

Document title: D.2.1 Technical specification of WeCare 2.0 Service

Editor: <P. Kiernan, Skytek>

Work package Leader: < S. Burgers, Ericsson>

Project Co-ordinator: <Sharon Prins, TNO>

This project is co-funded by AAL through the AAL programme.

List of Authors

Paul Kiernan (Skytek)

Rob Vermeulen (Sharecare)

Ilkka Ketola (Videra)

Arjan de Vos (Sharecare)

Mario Goorden (Ericsson)

Stefan Burgers (Ericsson)

Table of Content:

Section 1: Scope and Applicability

This document is the software design document for the WeCare project. It defines the overall architecture and the design of major components of the WeCare software. A high-level architectural overview is provided and additionally detailed documentation for single components as provided by the project.

Section 2: References

Section 2.1: Applicable Documents

The following documents contain provisions to be applied to produce the software design document or to be applied within the scope of the activities for which the software design document is relevant. For dated references, subsequent amendments to, or revisions of, any of these apply to this document only when incorporated in it by amendment or revision. For undated references, the latest edition of the publication referred applies.

Ref	Title
PROD-OVERVIEW	Overview of products and pilots, 25/03/2010
VGU-ARCH	Videra Video Graphics Unit System Architecture Simple, 07/04/2010

Section 2.2: Reference Documents

The following documents, although not a part of this software design document, clarify its contents:

Ref	Title
WECARE-PROPOSAL	AAL Proposal WeCare Final Version, 11/02/2009
UML standard	UML 2.0 Standard at : http://www.omg.org/cgi-bin/apps/doc?formal/05-07-04.pdf
Java Code Conventions	Code Conventions for the Java Programming Language at: http://java.sun.com/docs/codeconv/

Section 3: Terms, Definition and Abbreviated Items

Term	Description
Action Item	An action item is a documented event, task, activity, or action that needs to take place. Action items are discrete units that can be handled by a single person
Blogs	A blog (a contraction of the term "Web log") is a Web site, usually maintained by an individual, with regular entries of commentary, descriptions of events, or other material such as graphics or video. Entries are commonly displayed in reverse-chronological order. Blogs provide commentary or news on a specific subject. A typical blog combines text, images, and links to other blogs, web pages, and other media related to its topic. The ability for readers to leave comments in an interactive format is an important part of many blogs
Content Management System	A content management system (CMS) is a computer system that allows publishing, editing and modifying content from a central location. The systems are usually accessible via web interfaces and provide a collaborative environment for multiple users to view and update content.
Document Management System	A document management system (DMS) is a computer system (or set of computer programs) used to track and store electronic documents and/or images of paper documents.
Forum	An Internet forum is a web application for holding discussions and posting user-generated content. Internet forums are also commonly referred to as web forums, message boards, discussion boards, (electronic) discussion groups, discussion forums, bulletin boards. The terms "forum" and "board" may refer to the entire community or to a specific sub-forum dealing with a distinct topic. Messages within these sub-forums are then displayed either in chronological order or as threaded discussions
Instant Messaging	Instant messaging (IM) is a form of real-time communication between two or more people based on typed text. The text is conveyed via computers connected over a network such as the Internet.
Revision Control	Revision control (also known as version control (system) (VCS), source control or (source) code management (SCM)) is the management of multiple revisions of the same unit of information. Changes to these documents are usually identified by incrementing an associated number or letter code, termed the "revision number", "revision level", or simply "revision" and associated historically with the person making the change.
RSS Feeds	RSS is a family of Web feed formats used to publish frequently updated works such as blog entries, news headlines, audio, and video in a standardized format
Social Networking	A social network service focuses on building online communities of people who share interests and activities, or who are interested in exploring the interests and activities of others
Web 2.0	*Web 2.0* is a term describing the changing trends in the use of World

	Wide Web technology and web design that aim to enhance creativity, information sharing, collaboration and functionality of the web. Web2.0 concepts have led to the development and evolution of web-based communities and hosted services, such as social-networking sites, video sharing sites, wikis, blogs, and folksonomies.
Wiki	A wiki is a page or collection of Web pages designed to enable anyone who accesses it to contribute or modify content, using a simplified markup language. A defining characteristic of wiki technology is the ease with which pages can be created and updated. Generally, there is no review before modifications are accepted. Many wikis are open to the general public without the need to register any user account. Sometimes session log-in is requested to acquire a "wiki-signature" cookie for auto signing edits. Many edits, however, can be made in real-time, and appear almost instantaneously online.

Abbreviation/Acronym	Description
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CMS	Content Management System
CPU	Central Processing Unit
DMS	Document Management System
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IF	Interface
J2SE	Java 2 Platform, Standard Edition
JDBC	Java Database Connectivity
JRE	Java Runtime Environment
OS	Operating System
PDF	Portable Data Format
PHP	Personal Home Page (Scripting Programming Language)
QWG	Quality Working Group
RSS	Really Simple Syndication
SCORM	Shareable Content Object Reference Model
SMS	Short Message Service
SOAP	Simple Object Access Model
SOW	Statement of Work
SQL	Structured Query Language
SSL	Secure Socket Layer
TLS	Transport Layer Security
UI	User Interface
UML	Unified Modeling Language

URL	Unified Resource Locator
VoIP	Voice Over IP
VGU	Video Graphics Unit
VM	Virtual Machine
WYSIWYG	What You See Is What You Get
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

Section 4: Software Design Overview

This section describes the high-level software components, system boundaries and the architectural principles of the WeCare project. The UML Standard 2.0 is used for design documentation.

The overall architecture of the WeCare software is driven by some general design rules:

1. Manage all major functional components of the software within a single application framework to allow overall layout control and component interaction. Although the individual components are self contained in terms of later software re-use, testability and maintainability the integrated working environment for the pilot study end users is a major usability driver. Furthermore the provision of a standardised communication architecture for those components allows the development of functions which are based on the interaction of multiple components, improving the information flow and data exchange between components as well as providing additional valuable information to the end user and improving workflow.
2. Provides standardised interfaces, based upon SOAP, URL queries or XML message exchanges and a predefined set of implemented method interfaces, for later enhancements for the addition of further Web 2.0 collaborative components or for the exchange of a selected open source technology for a different product which may evolve better and faster than the selected technology in the future. The selected architecture provides this ability via allows close integration of new collaborative components in terms of component communication and integration of user interfaces. The interfaces which are used within the WeCare system are documented in the separate Interface Control Documentation.

The following diagram gives a high level overview of how different servers located at geographically separated locations throughout Europe where integrated to provide a single WeCare system and set of services to the end users. The diagram also shows how multiple groups of end users participated in the trials from different regions throughout Europe accessing these servers and related WeCare services.

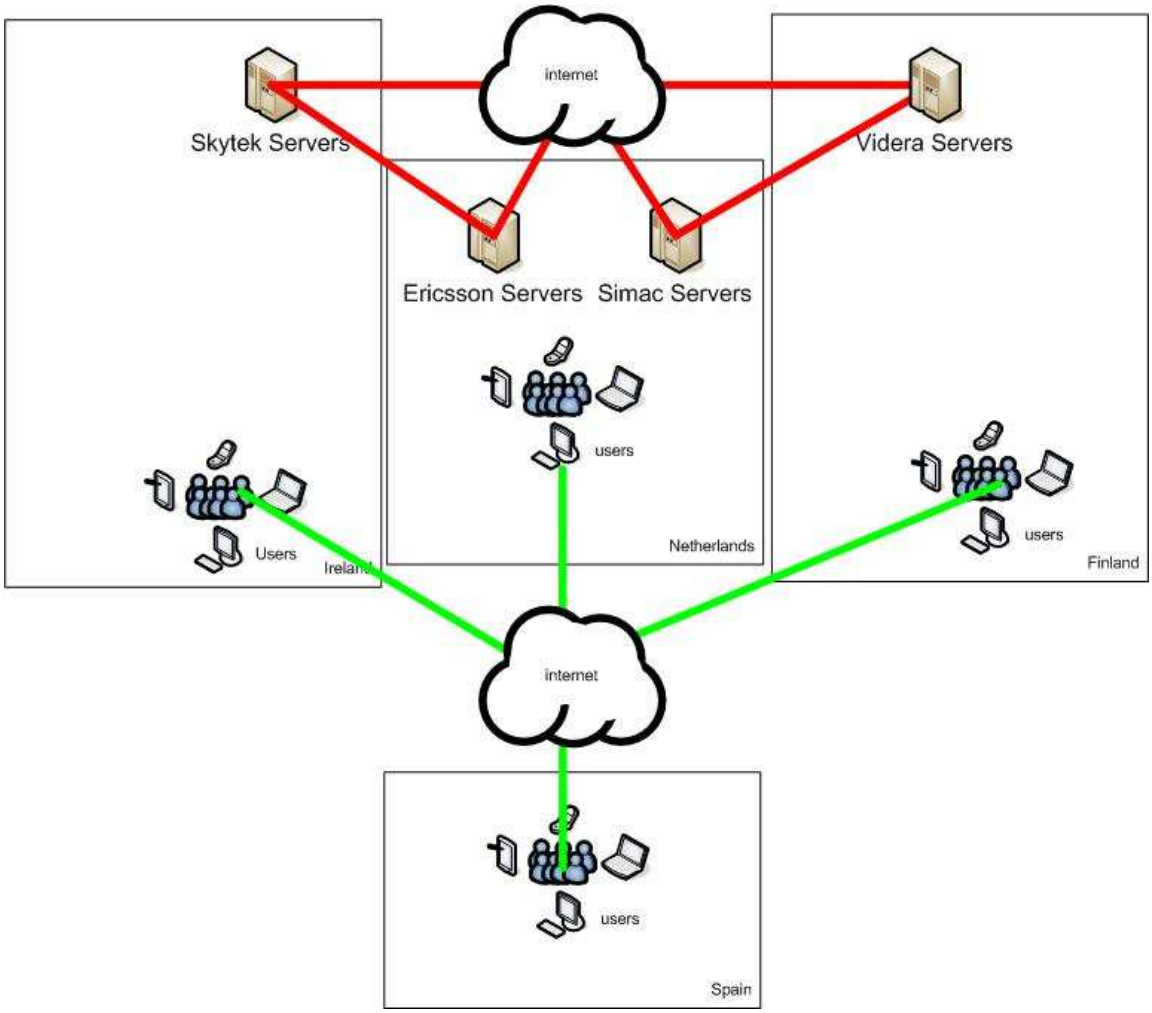


Figure 1 WeCare services provides by interconnected geographical distributed servers all used by different trail user groups

Section 4.1: Software Architecture

WeCare comprises different software components for the building of a social networking and collaborative framework for the easy exchange of information and data between groups of end users.

Section 4.1.1: WeCare Entities

The WeCare software consists of two major entity classes:

- Executable software, components and libraries residing on the Web Server
- Embedded web applications and the systems graphical front end that is executing in the end user's web browser

Being a web based application the WeCare system consists of a set of libraries and components which are deployed via several physical servers located in different places throughout Europe. The components are deployed by a standard web application server software application (Apache HTTP Server), via the Apache Tomcat web application server or

by a dedicated server component listening on pre-assigned port numbers i.e. Instant messaging. Also part of the application's execution files are resources (icons, fonts, themes, templates etc.) that are directly related to the application. The last group of files in this context are the start scripts for each component which are standard shell-scripts and batch-files. To support the functionality of the web application server an underlying MySQL database is used to store information and the web server is configured to support the execution of PHP authored pages. The system also contains an interface to IPX for sending SMS messages and an SMTP server to send emails.

The WeCare system components which are deployed via the Apache Tomcat web server depend on a Java Runtime Environment (JRE) which is installed on the web server as provided by the vendor (Oracle) without any adaptations.

The final class of files of the WeCare Application are the configuration files.

Section 4.1.2: Multiple End Users

WeCare is designed to support multiple end users simultaneously. Every end user uses a standard web browser executing on an individual machine or mobile device which is connected to the WeCare central server system.

End users web browser systems do not directly synchronize among each other using peer to peer technology. Every end user transfers information and any common files, information via the central WeCare web application server. The end users device do not share parts of their file system with others nor do they provide other access to their local storage.

Section 4.1.3: Mass Storage

The WeCare application provides mass storage for the end users via the hard disks that are attached to the deployment web servers. It is foreseen that there will be different levels of hard disks each providing different speeds and capacity of access. For components of the applications and files which are commonly accessed these shall be provided on the server file system which has the fastest access time. Archived files will be on a slower but larger system. The mass storage device of the application will also provide a RAID backup and redundancy.

Section 4.1.4: Events

The WeCare system provides a mechanism for logging information via the logging mechanism of each of the subcomponents functionality. The component based logging information is sent to a dedicated WeCare logging server which contains specialized analysis application to assist in the visualization and interpretation of the captured logging data. This mechanism is extended through the use of the logging system provided by both the underlying HTTP web server and Apache Tomcat web application server. Configuration changes specifying additional logging requests do not require re-compilation of the software but just a restart of the underlying web servers.

Section 4.1.5: Security

The general aim for WeCare is to be providing ease of access to the system but be as secure as possible for the safe storage of the information for end users and which it is managing. To enable a secure WeCare system the following approach is taken.

All WeCare components functionality can only be accessed through the provision of a username and password which has been assigned by the administrator of the system. A

single-sign on process is provided by the system which allows access to the functionality of all components once the user has logged into the system.

All interactions with the system will be logged via two separate mechanisms; firstly through the logging facility that is provided by the web application servers (i.e. Apache HTTPD and Apache Tomcat) used by WeCare and secondly through a dedicated Open Web Analytics (OWA) that is implemented within each of the major components of the WeCare system. These mechanisms are described in more details in the individual trial documentation reports. Both these generated logs can be fully analysed as part of the audit trail.

Finally the central WeCare web application will be deployed one of the most popular and advanced version of Linux (CentOS). The system shall be configured so that the latest security patches both for the operating system, for the web and application servers and for the system components are automatically downloaded and applied as they are made available. Standard security monitoring software, tripwire and virus protection software will be deployed on the WeCare central system server for verifying that the server hasn't been compromised.

Within WeCare trials single sign on was provided for each of the participating countries. This was a single URL location where the users could access the system through a pre-assigned username and password. Once the user accessed the system, session information is stored on the server for the individual. This session information is deleted once the user logs out of the system, after a predefined period of inactivity or the user's browser window is closed. To implement the single sign on process and access to WeCare components, the session information is used and passed to individual components when launched by the user. This information is checked by each component and if not valid then access is denied. This information also allows for the tailoring of the content that individual components display for the particular needs of the user. Currently the user details are passed as parameters within an encoded URL string to each of the individual components. While this approach is acceptable for the trials scenarios of WeCare it is recognised that for a complete commercial role out of WeCare that the URL approach for user information sharing between components should be enhanced so that the information is fully encrypted or a more general web service based approach is used for information sharing.

During the trial for the creation of accounts a certain amount of manual user creation and administration was required. Due to the distributed nature of the provision of the WeCare services and components, I.e. Several servers located in different countries and administrated by different organizations that are partners in the WeCare project were used, information on trial users needed to be shared between partners when accounts were required to be created or updated. The approach taken in WeCare was to use a single password protected Excel chart that contained a summary of all user information. This was shared on the common WeCare Sharepoint server and referenced by each organization that was involved in account creation and component access. Each organization was responsible for the update of user accounts for access to components that they developed or managed on their servers. Additional information such as trial membership, group membership emails and phone numbers of the users were also stored in the shared password protected Excel file for the duration of the trials. For the full commercial system, user account management would be fully automated via web services so that from a centralized administration point all changes to user account details would be automatically pushed out to all registered components and servers.

Section 4.1.6: Single Sign On

The WeCare system supports a single sign on for users to access the functionality provided by the system. This means that the users only requires a single logon details which they use on

the home page from which each component is presented. They can then access each of the WeCare subcomponents without having to re-enter their logon details.

The administrator will specify for an end users which components they have the rights to access and use within the system.

The sign on process of WeCare is handled by a dedicated component, once an end user is created or the details of an end users is modified these details are distributed to each of the components within WeCare using the defined standard interface between components of WeCare. The registration details of an end user are then processed internally within each component based on how the component internally handles the management of end users.

The single sign on model that will be used within the WeCare system is illustrated in the diagram below. This shows that secondary domains, in this case the series of WeCare components trust the primary domain login for supplying access credentials to the WeCare system.

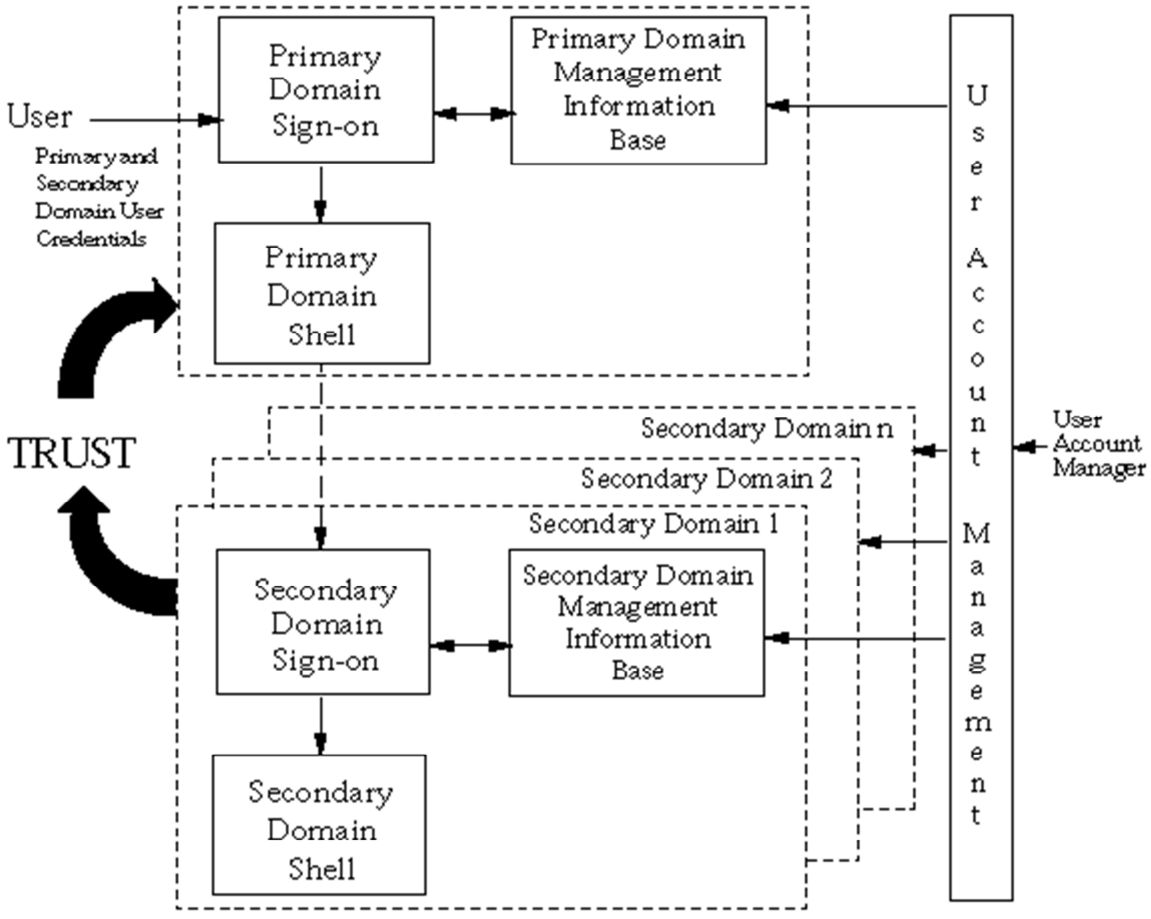


Figure 2 WeCare Single Sign On security model

Section 4.2: Interfaces Context

The UML Deployment System Architecture Overview Diagram in Figure 2 shows the overall context in which the WeCare components are running.

Section 4.2.1: Web Calendar IF

The Web Calendar is the system that allows users share events and date information between users and groups of the system. The calendar component interfaces to the Apache HTTP web server for the delivery of information to the web client of the user. The web calendar interfaces to the underlying MySQL database through standard PHP to database connections.

Section 4.2.2: Web Forum IF

The Web Forum allows a user to access and add to various forums that is managed by this system. Forums can be secured by groups. The system provides a web interface to its contents via a standard web browser to the user. The system interfaces to the underlying Apache HTTP web server for delivery of information and interfaces to an underlying SQL database through standard PHP connections for storage and retrieval of forum contents.

Section 4.2.3: Content Management System IF

The Content Management System is the components that allow user share documents, articles and information via a common web interface. Information can be edited online and through a WYSIWYG interface. The content management system interfaces to the underlying Apache HTTP web server for the delivery of information to the web client of the user. The web calendar interfaces to the underlying SQL database through standard PHP to database connections. The underlying database stores the content of articles, edit information, news entries and user preferences.

Section 4.2.4: Document Version Control IF

The Version control system provides the ability to share multiple versions of documents between multiple end users. The underlying version control system resides on the web server and interfaces to the user via a custom component which is deployed as a web client application. The client through the underlying web application server communicates with the version control application which resides on the server.

The interface is standardised for access to the version control system and the web interface can be replaced with a different version control client which can be integrated into their development environment. For example the end user could swap the web client of the WeCare system for a component directly integrated into Word. These alternative interfaces will provide the same features of document version control provided by the WeCare system.

The version control system interfaces to the underlying SQL database through the use of Java database interfaces.

Section 4.2.5: Action and Task Tracking IF

The Action and Task Tracking system is the components that allow users create action and tasks which can be tracked by multiple users via the web interface. The GUI is provided through a standard web browser interface. The action and task tracking system interfaces to the underlying HTTP web server for the provision of the information to the end user. The system interfaces to the underlying SQL database for the storage of information relating to action and task tracking through the user of Perl based modules for database connectivity.

Section 4.2.6: Wiki IF

The Wiki system is the components that allow users create edit and manage Wiki pages. The GUI is provided through a standard web browser interface. The web client provides a rich client interface for the provision of a WYSIWYG editing suite and this interfaces directly with the underlying editing component for the exchange of the underlying Wiki formatted text. The component communicates directly with the underlying HTTP web server for the provision of the Wiki pages and functionality. Communication to the underlying SQL database for the storage of Wiki entries are through standard PHP to SQL communication interfaces.

Section 4.2.7: User Management IF

The user management component provides a single point of user management for the end users registered on the WeCare centralised server. Therefore through this component the user has access to all the components of the system without having to log into each one individually. The component communicates directly with the underlying web application server for the provision user management functionality. Communication to the underlying SQL database for the manipulation of information for each component to allow user access is provided by standard Java to SQL database (JDBC) interfaces. The user management system also provides advanced group management definition and storage where both global groups that a user belongs to along with user created and owned group details are stored.

Section 4.2.8: General Information Services IF

The general information services were used within the Irish trials and offered access for the end user to summary information on wide range of sports and news articles that were identified as of interest to the Irish trial users. Both these systems interfaces directly via server side components to RSS feeds provided by the Irish government's main broadcasting and news authority titled **Raidió Teilifís Éireann (RTE).** The RSS feeds were interpreted and visualised in a user friendly manner within these components where the articles title, summary information and links to further information was provided. If a user clicked on a 'more details' link then the original article on the web was obtained and displayed within the WeCare system framework.

Section 4.2.9: Weather Services IF

The Irish weather services component provides a range of information from summary weather information per Irish region to a range of satellite images within the visible and infra-red range to the user. Within WeCare the component interfaces to the Irish National Meteorological Service which is a division of the Department of the Environment, Community and Local Government. The organisation is also known as Met Eireann. Through an interface that the component has with the Met Eireann web site (www.met.ie) weather details are obtained and displayed within the WeCare component.

Section 4.2.10: Phone/Video Calls

This component provides access to phone and video calling functionality within the WeCare system. The component interfaces with the Skype component that is installed on the end user system and the interface passes the username details and automatically launches the Skype

application from the end users WeCare web interface when they wish to make a phone or video call.

Section 4.3: Memory, CPU and other budgets

CPU budget and hard disk space are limited resources especially when located on a web server where multiple users are accessing the system in parallel. In order to ensure responsiveness of the system each of these limiting factors needs special consideration. The selection of the architecture for the WeCare does allow a 'virtual' web deployment machine to be built up where the load can be distributed over several machines. However each machine should be optimised for maximum usage of its resources. An additional architecture advantage of the WeCare solution is that the services provided are distributed over several different physical servers each of which are located in a different country. This allows the system to remain responsive even under a heavy user load as the use of components by users is distributed over the different servers.

Section 4.3.1: Memory

To minimize the memory usage, each component running on a single hardware server are configured to utilise a single instance of underlying web server and application server instead of a separate instance of each one. This greater reduces the memory load of the WeCare system.

Secondly a single instance of the underlying SQL database for data storage within the WeCare centralised server has been selected and implemented instead of a separate database and instance for each component running on the server. Within the selected database each component is managed by a separate table.

Finally, common libraries for provision of functionality are integrated as modules into the WeCare centralised web and application servers. Through common usage of these modules the memory load is reduced for the central server. Modules which are used by multiple components included the PHP execution module of the web server and the JRE of the web application server.

In addition standard Java VM features for limiting the maximum total amount of memory are in use to avoid degradation of the WeCare central server system functionality due to memory consumption of the application. The Java Virtual Machine provides parameters to constrain the maximum amount of memory which the VM may use.

Section 4.3.2: Hard Disk Usage

The amount of hard disk space that is required for the WeCare system for the storage of end user generated content is expected to be large due to the number of different information management components that are provided.

Limiting hard disk budget for the system can be accomplished by using disk quota on OS level but this is currently not foreseen. During deployment, usage statistics shall be prepared and will drive the deployment of additional hot pluggable hard disks for storage.

Section 4.3.2.1: CPU

The available CPU power for the WeCare centralised server needs to be shared among all components that are deployed on the server. A dual processor system has been selected to

enhance performance on the main central single sign on server for WeCare and it usage statistics will be monitored to calculate when an additional physical server would need to be integrated for continuing performance.

Section 4.3.2.2: Network

The network for the provision of the WeCare services needs to be of high quality, reliability and speed for the delivery of the solution to multiple locations throughout Europe in an efficient and fast mechanism. This will be achieved through the deployment of the WeCare central single sign on web server in a dedicated web farm with fibre optic connections directly to the backbone of the Internet. This in addition to each of the support servers being located on high bandwidth network connections throughout Europe provides for a fast response and low latency response in connection to the WeCare system.

Section 4.4: Design Standards, Conventions and Procedures

The WeCare application is a Web 2.0 based system which is implemented in both J2SE and PHP. J2SE is the main programming language for the implementation of dedicated components and features which are not provided by the integrated open source components. With Java being an object oriented language the analysis and design methodology for the dedicated component follows the object oriented approach as well. The following sections will provide more detailed information about design and coding relevant standards.

Section 4.4.1: Software Design Method

For documenting and illustrating the software design the Unified Modeling Language (UML) is used. Being a specification language UML is not a method itself but supports the object oriented approach with a wide variety of supported diagram types, objects and interactions.

The UML is a non-proprietary, object modeling and specification language used in software engineering. Providing various diagram types the UML supports to illustrate the following three prominent parts of a software system:

- Functional model
- Object model
- Dynamic model

While use cases are maintained in text/table format in this project instead of use case diagrams all other available UML diagram types are used for documentation where appropriate.

Section 4.4.2: Code Documentation Standards

The Java programming language provides specific language structures for documenting source code. Together with the specification of the documentation format a tool (javadoc) is provided to generate HTML based documentation of a software application.

In WeCare the use of javadoc documentation is enforced by an automated tool (checkstyle) that makes a software build fail if no according documentation is available. The configured rules of this tool require text documentation for all public elements.

Section 4.4.3: Naming Conventions

The naming conventions in WeCare are derived from the Sun Code Conventions for the Java Programming Language (see reference documentation). The compliance with the defined rules are forced by an automated tool (checkstyle) that makes a software build fail in case they are not followed.

Section 4.4.4: Programming Standards

A consistent coding style is ensured by an automated check on all source code. A set of rules – derived from the Sun Code Conventions for the Java Programming Language – has been defined and is enforced using the tool checkstyle that makes a software build fail in case the rules are not followed. The detailed list of checks is documented in annex Section 10.2: .

While a consistent coding style is important for readability, maintenance and review, it is not enough for ensuring good code quality. Due to the complexity and mere quantity of user interface centric software applications the quality of the design and implementation is measured utilizing tools for calculating common object oriented software metrics. Examples of such metrics include 'fan-out' (the number of classes called by a class) and 'fan-in' (the number of classes used by a class).

Section 4.4.5: Overall Software Configuration Management

During the development process each partner was clearly assigned modules, components and features of the WeCare system that they would be responsible for in developing and integrating into the overall solution. Due to the loosely coupled nature of each of the WeCare components and the clearly defined interfaces between components this approach worked well for the complete WeCare development phase.

Since each organisation was over the development of their own clearly defined WeCare component and features it allowed each organisation retain their dedicated development processes, methodology and software development infrastructure.

Skytek use the Atlassian Wiki system for the statement of details on system requirements and individual WeCare issues are defined using a Jira based tracking system. Development is done through the Eclipse framework and code is checked into a Subversion based software management system. Automatically continuous integration, code building and execution of developed JUnit tests against the developed code is performed and information made available to the development team. Checkstyle rules for quality of code are applied to the code base. Test servers are used in house within Skytek to validate the developed WeCare components functionality and verification against the defined system requirements.

Ericsson uses subversion as their code revision system for the offered modules. Customisations and trial specific adaptations were stored in a WeCare code branch.

ShareCare uses SVN as their version management system

Section 5: Software Design

This section describes in detail the major software components and their relations.

Section 5.1: Overall Architecture

The description of the overall architecture is split into an overview of the WeCare application software layer. It identifies for each part the major components and their relation to each other. The following diagram provides an overview of how the different partner components were integrated with a centralised server from which the end users started to access the WeCare 2.0 system through the login service.

Once the end user connected to the system the main functions of the system are available to them. These functions presented within the diagram can be geographical distributed on several physical servers and this was the case with the WeCare demonstrators where Skytek components were located on the primary WeCare access server while ShareCare, Videra and Ericsson components where provided through separate servers physically located throughout different countries in Europe. For a complete description of all components and functionality provided by each of the consortium partners please refer to chapter 5 and chapter 6 of this document.

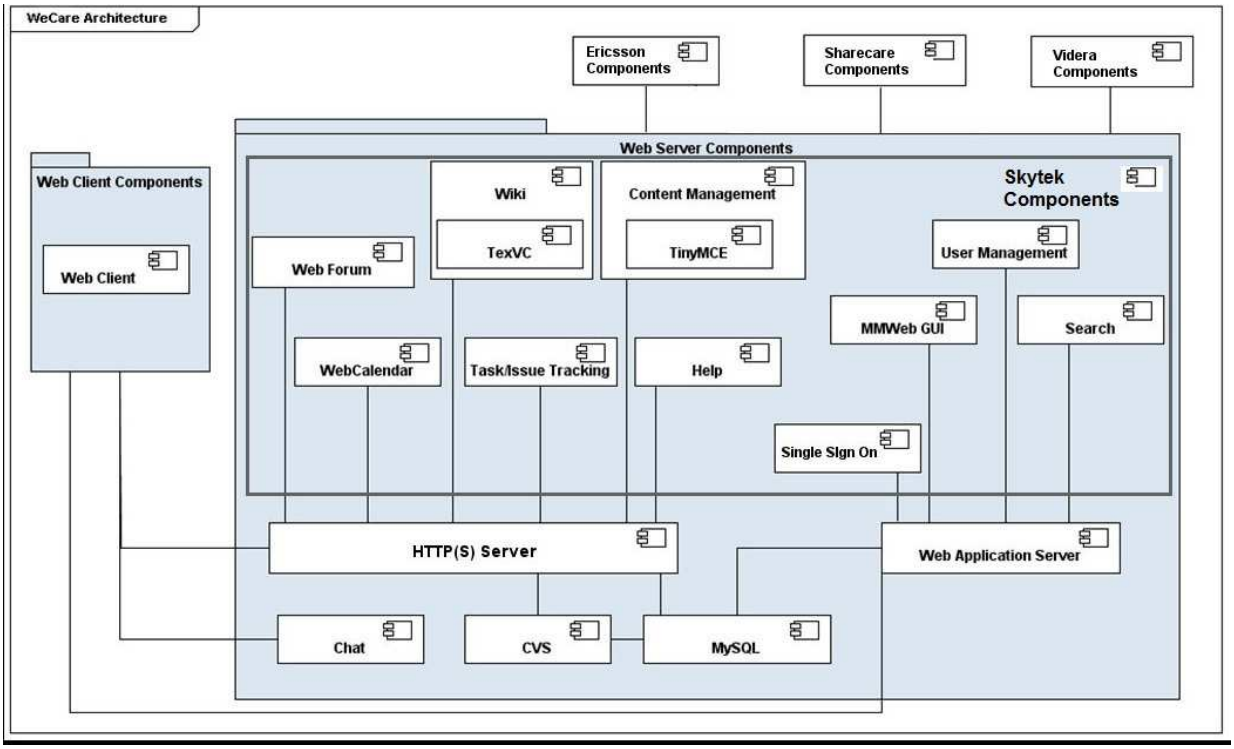


Figure 3 WeCare System Architecture providing a functional overview and showing how different partners components were integrated.

Section 5.1.1: Web Server Component Overview

Web Server Component in the context of WeCare means all software which is to be installed on the web server platform and form the basis of the WeCare application.

Section 5.1.1.1: Application Technical Platform

The WeCare system software is a pure web deployed application which is delivered via a standard web interface. The systems functionality is based upon both standard Java and PHP technology and just requires a Java Run time and PHP interpreter to be available on the

deployment platform. Both Windows, Linux and Mac OS platforms provide such JRE and PHP interpreters for their operating systems therefore the application platform is very flexible for deployment of the WeCare system.

By the use of Java the application has no direct dependencies on the executing hardware or underlying operating system. File and network access is provided by the Java VM and the application does not use any operating system specific functions.

The system is accessed through the use of two separate web interfaces, firstly a HTTP server for delivery of web pages and web applications based upon the PHP language. Secondly, the system use a web application server for the deployment and access to applications which are built upon JSP, Java Servlets and Java Bean technologies. These two separate web servers will be integrated to provide a common interface for end users to access. The selected web servers (Apache HTTP Server and Apache Tomcat) can be deployed on multiple OS platforms therefore provide flexibility in selection of the technical platform.

Finally, the system requires an underlying SQL database. Standardised interfaces to the SQL database are used thereby providing flexibility in the selection of the technical platform for the database to be deployed on. For the WeCare system the MySQL database has been selected as the underlying database. This database can natively be deployed on a number of operating systems.

Section 5.1.1.2: Graphical User Interface

All interaction to the WeCare system is made available to the end user via standard web browsers. All user interface elements such as buttons, menus and fields are standard web browser components. These standard components provide a consistent appearance.

A major functional component of WeCare is the portal home screen that is displayed to the end user once they log into the system through the single sign on page. This home page is customised to the needs of each of the WeCare trials that were performed. Through the portal screen the user has access to the major components of the system available for that trial. In addition the portal screen can show additional component information. Details such as summary content extracted from connected components can be display on the page. The portal page can be personalised for each of the end users. This information can include the components that the user has been allowed access through to the display of all ‘closed’ groups, as in the Dutch trial. These closed groups are groups that the individual can create and invite other WeCare users to join.

The interface is designed using Web 2.0 technologies to provide a highly interactive interface which could not be provide using traditional web site design approaches. The Web 2.0 based approach brings together a mix of Dynamic HTML, Javascript and AJAX technology. Ajax (asynchronous JavaScript and XML) is a group of interrelated web development techniques used for creating interactive web applications or rich Internet applications. With Ajax, web applications can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page. Data is retrieved using the XMLHttpRequest object or through the use of Remote Scripting in browsers that do not support it

Section 5.1.1.3: WeCare 2.0 System deployment

The WeCare system architecture allows for the flexibility to deploy the individual components over different hardware platforms that are located in different physical locations, in addition

individual components can be distributed over a single 'virtual' server which can be in reality several physical servers where automatic load balancing is applied by the virtual server configuration. For the WeCare trials no virtual server configuration was required due to the limited number of end users however several physical servers each with different components and located in different locations throughout Europe were used to provide the infrastructure for the WeCare trials.

In summary the four physical locations used for the WeCare servers and system deployment were:

- Dublin, Ireland (Skytek managed servers)
- Netherlands (Sharecare managed servers)
- Netherlands (Ericsson managed servers)
- Finland (Videra managed servers)

To ease the monitoring of uptime and availability of the different physical WeCare servers during the trials different website monitoring services were used. The two major services deployed were:

- Nagios, an open source IT infrastructure monitoring solution
- Montastic (http://www.montastic.com) a website monitoring cloud service.

Each of these services constantly checked the availability of each of the hardware servers for the WeCare solution and informed all technical contacts of the WeCare partners if a server became unavailable during the trial period.

Section 5.1.1.4: Data Security

The security of data within the WeCare system is of key importance both to the WeCare consortium and the end users that will be involved in the trials. The system will provide security at two distinct levels:

1> At the application level, a single point of user authentication is provided in the system. The user authentication is based on an assigned username/password combination.
This level of authentication can be extended to use a hardware token, as described in http://en.wikipedia.org/wiki/Security_token where for example a digital display on a key fob that the end user has and where the displayed number has to be entered along with their username and password.
Upon user login secure session identifiers are assigned to each user. This is used to confirm each requested page and if not valid redirects the user to a login page. These sessions automatically time out if the user doesn't use the system for 20 minutes (can be changed) or the user logs out of the system. All session identified are sent to a web client in encrypted format.

2> At the server security level. The server OS is constantly updated with latest security patches and has a number of monitoring software such as tripwire to inform administrator of any unauthorised access attempt. The data stored on the WeCare server will be encrypted at the OS level through the use of TrueCrypt technologies. More details are available at http://www.truecrypt.org/

Section 5.2: Available Software Components Design

The System Requirements Documents identifies high-level design objects in the context of existing products and organizational boundaries. These objects form layers that are reflected in the structure of this chapter.

Section 5.2.1: Web Server Application Components

The web server WeCare components are all described on the same level without reflecting dependencies in this document's hierarchy. In case a component is a compound of several other components this information is given in the description of the component itself.

Section 5.2.2: Web Forum

Type

Component

Purpose

Provide a web forum within the WeCare system.

Function

The web forums main functions and features can be broken down into a number of categories. Each category is stated below with the main features that the component provides.

General functions provided are:

- Comprehensive template system.
- Advanced permission and user management.
- Tracking of new and old unread topics
- Multi-media output. (XHTML, XML, RSS, WAP)
- Multi-language support
- Package manager that automatically installs or uninstalls mods
- Search the entire forum, a category/board or within a topic.
- Personalised search within your personal messages

Security related features are:

- Actions require a session based authorization code
- Administrative actions require the user's password
- Major actions are time and IP locked, preventing 'hammering'
- The number of login attempts from a certain IP can be limited and time locked

Forum Settings functions are:

- Ability to display page creation time and query count per page
- Put a board into maintenance mode, allowing only admins to login.
- Word censoring, either full word or partial
- Ability to break up long words.

Boards and Categories functions are:

- Group boards into collapsible categories.
- Set categories as non-collapsible.
- Reorder boards within categories, or reorder categories.
- Create child boards under other boards. (sub boards)
- Assign moderators to boards.
- Allow certain membergroups to access a board, including guests *only*.
- Configure permissions for each membergroup on the board level.
- Ability to indicate new posts to child boards but nothing new in parent.

Member Registration Functions are:

- Require registration before forum entrance
- Require a user to agree to terms before they register
- Disable member registration completely (allowing only moderators to register people).
- Require email authentication by sending an authentication link.
- Require a moderator to approve registration.
- Register new members from the admin center.

Navigation and Authentication Functions are:

- Several security checks during navigation.
- Password reminder option, by email with confirmation.
- Both cookie and session based authentication
- Cookies can be set local to a path, global to all subdomains, or normally
- Adjustable expiration time for authentication cookies

Member Tracking and Tracing Functions are:

- Sortable and searchable public
- Powerful sortable and searchable admin memberlist
- Show all (error) messages and IPs made by a member. (track user)
- Show all (error) messages from an IP address or range. (track IP)
- See who's doing what (accessible by permission.)

Statistics functions are:

- Several board statistics (accessible by permission.)
- Tracking of member's online time in seconds.

- Tracking of topics, messages, new members, and hits per day.
- Individual member statistics accessible from their profile.

News and announcements functions are:

- Ability to create announcement boards (members receive a notification of topics automatically.)
- Member option to disable receiving announcements.
- Email or private message your members by membergroup.
- Show a news ticker or news fader.

Communication functions are:

- Ability to choose sendmail or SMTP (with or without authentication.)
- Ability to send a topic to another member of a quality working group.
- Ability to view a "printer friendly" version of topics.

Membergroups functions are:

- Create membergroups to group members on permissions, access rights, and/or appearance.
- Assign several membergroups to a single member, with one membergroup as the primary group.
- Define membergroups that are auto-assigned based on the amount of posts a user has.
- Determine the maximum number of personal messages a membergroup is allowed to have by group.
- Assign graphical symbol(s) to a membergroup - by primary group.
- Determine which membergroups are allowed to access a board.
- Assign graphical symbol(s) to a membergroup - by primary group.
- Determine which membergroups are allowed to access a board.
- Determine which membergroups are allowed to access a board.

User access restriction functions are:

- Ban members based on their username, email address, IP address or hostname.
- Support of wildcards for email address, IP address, and hostname.
- Include a ban reason (viewable for the banned user).
- Include a ban note (only viewable by the admins).
- Chose between full ban, 'no post' ban, or registration ban.

News Feed Support functions are:

- Export forum data using XML/RSS
    - Latest members.
    - News.
    - Recent posts.

Theming and Templating functions are:

- Ability to allow or disallow your users to select their own theme.
- Ability to reset all of your members to a certain theme.
- Ability to install a new theme via your administration center.
- Default templates are XHTML 1.0 Transitional and CSS 2.0 compliant.
- Admin can add smileys and smiley sets.
- Members can choose which smiley set they wish to use (or none.)
- Themes can be installed by way of the "latest and greatest themes" panel.
- SSI can have and show layers and the like from the template system

Forum Posting Functions are:

- Spell Check.
- Quick Reply
    - Members can disable it or collapse it.
    - Can be used with "Quote".
    - Also contains "Spell Check".
- Vast number of "bulletin board codes" to use (including rtl, acronym, and others.)
- Optional editing grace period before a post is shown as modified.
- 'Insert Quote' feature on posting screen to quickly quote previous replies.

Subordinates

The web forum component has the Apache HTTP web server as its subordinate to execute the system.

Dependencies

The web forum component has the Apache HTTP web server to deployment of the system.

The web forum uses PHP as its implementation language. It depends on a PHP interpreter for execution.

The web forum uses MySQL as the underlying data storage system.

References

The web forum component is built upon the open source 'Simple Machine Forum' http://www.simplemachines.org/ system.

Section 5.2.3: Wiki Component

The Wiki component provides the end users with the ability to create, edit and manage 'Wiki' pages using the Wiki representation language for online editing of the content. A Wiki is a piece of server software that allows users to freely create and edit Web page content using any Web browser. Wiki supports hyperlinks and has a simple text syntax for creating new pages and crosslinks between internal pages on the fly.

Wiki allows the organization of contributions to be edited in addition to the content itself.

Type

Component

Purpose

Provide a complete wiki management and editing environment within the WeCare system.

Function

The Wiki's main functions and features can be broken down into a number of categories. Each category is stated below with the main features that the component provides.

Look and feel functions provided are:

- Links with shortcuts
- Skins
- User styles: Users can adapt the look and feel of the site through custom CSS on their user pages.
- "Stub" threshold: Users can see links to articles below a certain size rendered in a different color
- Printable versions of articles can be generated
- Auto-number headings in an article (optional)
- Intra-page Anchors (automatically generated for headings, and also with ...)
- Generate a table of contents for long articles (optional)
- Automatically turn ISBN numbers into links to an editable list of booksellers
- XHTML-compatible output (or darn close to it), tidy integration

Multimedia and extensions functions are:

- File upload feature allows to upload graphics or sound files
- Automatic resizing of images using ImageMagick or libgd

Editing tracking functions are:

- Watchlist. Every page has a link "Watch this article for me".
- User contributions in the sidebar of each user page list all articles the user has worked on.
- Extended recent changes with dynamic collapsing of edits to the same article and quick links to diff the edit, show the article history, show the user page, show the user talk page, or block the user (for sysops)
- "Related changes": View a filtered version of Recent Changes to the pages linked from the current page.
- Side-by-side diffs - the diffs are shown side-by-side, and changed portions of lines are highlighted, making it much easier to see what's what. Additionally, a diff is shown during an edit conflict so you can see exactly what you need to reintegrate.
- Real names. Users can (optionally) specify a "real name" they want to use for author credits.
- On-page credits. Administrators can enable an on-page paragraph giving credit to editors who've worked on a page.

Structures and syntax features are:

- Editing syntax based on UseMod, with support for mixing wiki-syntax and HTML. Only free links are supported for linking, not CamelCase.
- Namespaces allow content separation
- Transclusion of arbitrary pages in any namespace: {{:Page Name}}.
- Word-extension linking:
- Parenthetical hiding If you include a link of the form [[kernel (mathematics)|]], the parenthetical portion will be hidden in the link.
- Link to individual sections of an article
- Support for subpages
- Special keywords for inserting dynamic data such as the name of the current page, the current date, the number of articles, etc.
- Multiple Categories can be assigned to any page.

Editing functions are:

- Section editing.
- Edit toolbar (JavaScript-based), extended by a rich editor for non experienced quality working group end users to have a more word like WYSIWYG environment.
- Edit summary which is shown in "Recent changes".

- Double click editing: Users can enable an option that allows them to edit articles by double clicking them.
- Edit preview
- Handle edit conflicts (page being saved by a user while still being edited by another one, then saved again). MediaWiki will merge changes automatically if possible and otherwise require the user to do a manual merge.
- Mark edits as minor
- Anti-spam features

Discussion functions are:

- Talk pages: Each user (including every anonymous user) and every article has an associated page where messages can be left.
- Message notification (user gets a "You have new messages" notice if someone else has edited their user discussion page); this also works for anonymous users
- Automatic signature: Just type three tildes (~) when you edit, and on saving the page, it will be replaced with your user name and a link to your user page. If you use four tildes, the current date will be added as well. Mainly intended for Talk pages.
- Support for emailing users through the wiki

Subordinates

The Wiki system uses the **texvc** system to render mathematical formulae's. The program texvc (TeX validator and converter) validates LaTeX mathematical expressions and converts them to HTML, MathML or PNG graphics.

The HTML-code generated by texvc is based on three classes.

1. Conservative: the code should look good and work well in most browsers.
2. Moderate: the code should work and look good in reasonably modern browsers.
3. Liberal: the code is HTML, but it is designed for very recent browsers.

Dependencies

The Wiki component has the Apache HTTP web server to deployment of the system.

It uses PHP as its implementation language. It depends on a PHP interpreter for execution.

The web forum uses MySQL as the underlying data storage system.

The libraries ImageMagick and dvips are used for rendering of the Wiki entries.

References

The Wiki component is built upon the open source 'MediaWiki' http://www.mediawiki.org/wiki/MediaWiki system.

Section 5.2.4: Content Management Component

The content management component provides the end users with the ability to create, edit and manage articles and information within the WeCare system. Content can be simple text, photos, music, video, documents, news or just about anything you can think of. It requires almost no technical skill or knowledge to manage since the CMS manages all the WeCare end user groups content. Some CMS are quick complex and technical to use where users have to be experienced in concepts such as automated revision numbers, document check-in and check-out. The selected component chosen for integration into WeCare provides a highly graphical front end which hides the complexity of content management from end users. The interface uses Web 2.0 technologies to provide the ease of use to the end users.

Type

Component

Purpose

Provide a complete content management system within the WeCare system with powerful editing and administration tools for the non expert to use.

Function

The Content Management components main functions and features can be broken down into a number of categories. Each category is stated below with the main features that the component provides.

Core functions provided are:

- Banners allows the user to manage banners by Categories and Clients
- The “Contacts” component allows the user to manage a Contacts directory with regard to the Contacts within the web site.
- The “Contacts” tab shows a detailed list of active (published and unpublished) contacts. It also provides the ability to add new or edit existing contacts and their details.
- The “Categories” tab allows whole categories of contacts to be organised and “Published” status changed

- The “Newsfeeds” component enables the most recent articles from external web sites feeds (e.g. RSS) to be linked to for further reading. The “Feeds” tab contains a detailed list of all active (published and unpublished) feeds within the website. Important information is instantly visible such as “Category,” “# Articles,” and “Cache Time.”
-
- The “Polls” component displays a list of all active (published or unpublished) polls on the MM Web 2.0 site
- The “Search” component provides statistics on searches
- The “Weblink” component lists and provides management controls for controlling Web Links displayed in the Front-end of the web site.
- User editing of contents.
- Easy creating of new content and association within a branch of the system.
- Template selection for the ‘look and feel’ of the interface.

Additional functions are:

- Archiving of content within the system.
- “Breadcrumb” navigation of content.
- “Footer” ability to be displayed on each page of the CMS.
- User management and support.
- Most read and latest news.
- “Newsflash” ability.
- “Syndicate” support of content to end users favorite reader.
- “Who’s Online”

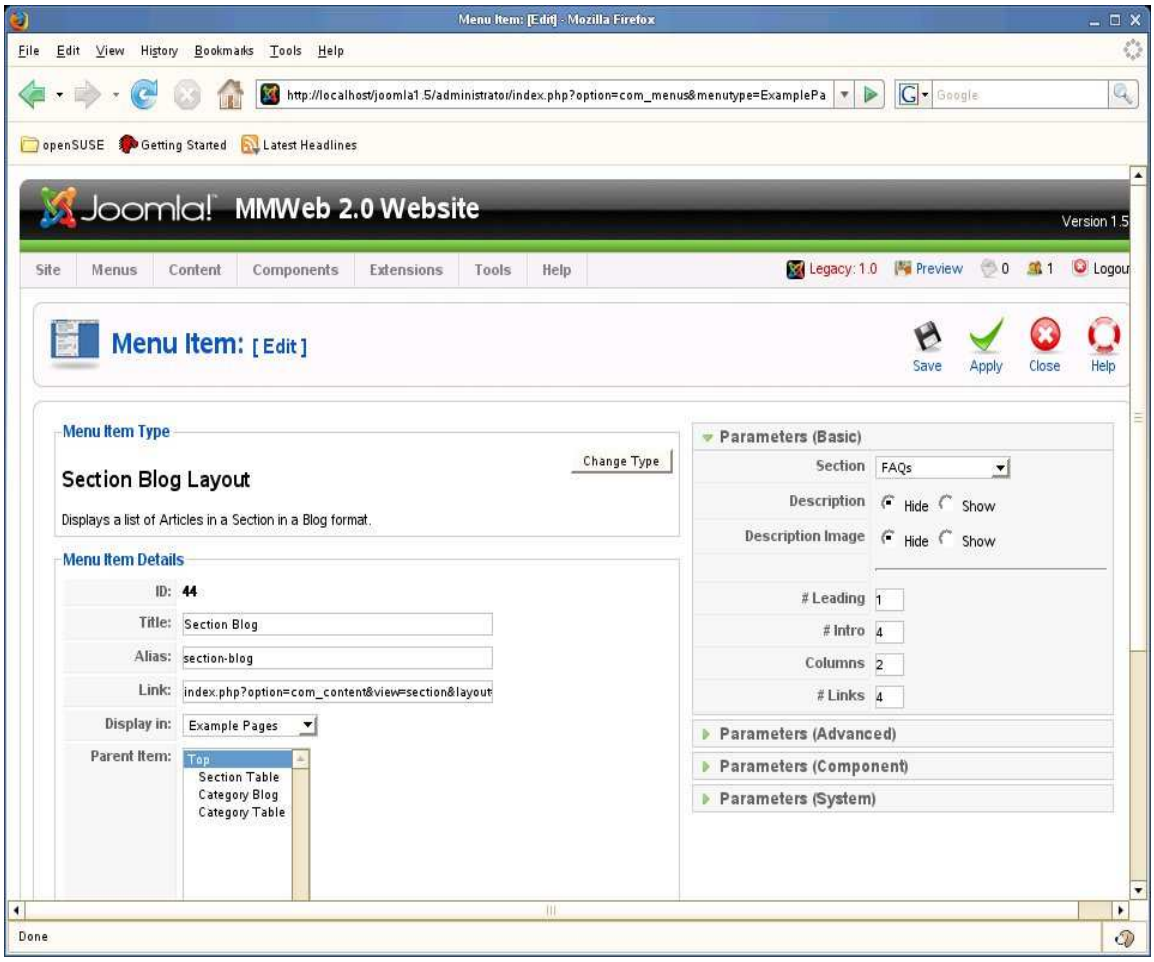


Figure 4 Blog support in CMS

Subordinates

N/A

Dependencies

The Content Management system component has the Apache HTTP web server to deployment of the system. It uses PHP as its implementation language. It depends on a PHP interpreter for execution.

The system uses MySQL as the underlying data storage system.

The Content Management component uses the TinyMCE 2.1 component to WYSIWYG editors. This component provides a standard set of editing icons and features which users would be used to through the use of a document editing software system.

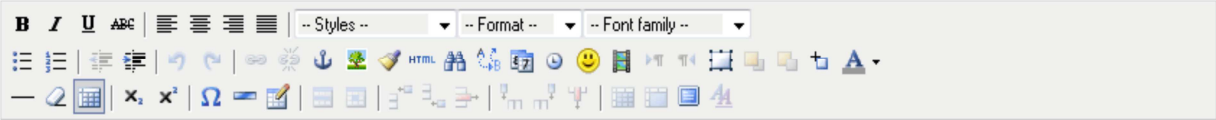


Figure 5 Interface of rich editing suite for CMS

References

The Wiki component is built upon the open source 'Joomla' http://www.joomla.org/ system.

Section 5.2.5: Concurrent Version Control Component

The concurrent version control component provides the end users with the ability to upload, share, and exchange document of any format between multiple users of the WeCare system. The key strength of the version control system is that all revisions of a document are stored and accessible by the end user so a previous version can be recalled if the user wishes. Additionally, the system allows several users edit a document in parallel and merge the differences back together upon completion. This provides major productivity compared to a traditional approach of a single user locking the file from edits by all other users.

Type

Component

Purpose

Provide a complete version control system for all documentation within the WeCare and the ability to share documents through a web based system rather than using FTP server mechanisms.

Function

The concurrent version control component's main functions and features can be broken down into a number of categories.

Core functions provided are:

- Most standard CVS features supported.
- Directories are versioned.
- Copying, deleting, and renaming are versioned.
- Free-form versioned metadata ("properties") to be attached to any file or directory. These properties are key/value pairs, and are versioned just like the objects they are

attached to. Subversion also provides a way to attach arbitrary key/value properties to a revision (that is, to a committed changeset). These properties are not versioned, since they attach metadata to the version-space itself, but they can be changed at any time

- Atomic commits. No part of a commit takes effect until the entire commit has succeeded. Revision numbers are per-commit, not per-file, and commit's log message is attached to its revision, not stored redundantly in all the files affected by that commit.
- Branching and tagging are cheap (constant time) operations.
- Merge tracking.
- File locking.
- Symbolic links can be versioned.
- Executable flag is preserved.
- Apache network server option, with WebDAV/DeltaV protocol.
- Parseable output.
- Localized messages.
- Interactive conflict resolution.
- Repository read-only mirroring.
- Write-through proxy over WebDAV.
- Natively client/server, layered library design with clean APIs.
- Binary files handled efficiently.
- Costs are proportional to change size, not data size.
- Bindings to programming languages.
- Changelists.

Subordinates

The concurrent version control system uses the a web client as a mechanism for access to the version control system. This is describes in the following section.

Dependencies

The concurrent version control component has the Apache HTTP web server to deployment of the system and integrates the version control system service directly into the HTTP web server for access. The system uses MySQL as the underlying data storage system for the documents and information on version revisions.

References

The concurrent version control component is built upon the open source ‘Subversion’ http://subversion.tigris.org/ system.

Section 5.2.6: WebCalendar Component

The Webcalendar component provides a system for the sharing of events, meeting schedule between groups of users on the WeCare system.

Type

Component

Purpose

Provide a web based interface to shared calendars of events, meetings and schedules. Advanced graphical interface to the system appears as if it is a locally installed application. Ability to export to the user own calendar system provides ease of integration into their normal calendar application.

Function

The web calendar has a number of functions available to the end users.

Core functions provided are:

- XHTML/CSS compliance
- Multi-user support
- 30 supported languages
- Auto-detect user's language preference from browser settings
- View calendars by day, week, month or year
- View another user's calendar
- View one or more users' calendar via layers on top of your own calendar
- Add/Edit/Delete users
- Add/Edit/Delete events
- Repeating events including support for overriding or deleting (exceptions)
- Configurable custom event fields
- User-configurable preferences for colors, 12/24 time format, Sun/Mon week start
- Online help
- Checks for scheduling conflicts
- Email reminders for upcoming events
- Email notifications for new/updated/deleted events
- Export events to iCalendar, vCalendar or Palm
- Import from iCalendar, vCalendar or Palm
- Optional general access (no login required) to allow calendar to be viewed by people without a login (useful for event calendars)
- Users can make their calendar available publicly to anyone with an iCalendar-compliant calendar program (such as Apple's iCal, Mozilla Calendar or Sunbird)
- Publishing of free/busy schedules (part of the iCalendar standard)
- RSS support that puts a user's calendar into RSS (WebCalendar 1.1+)
- Subscribe to "remote" calendars (hosted elsewhere on the net) in either iCalendar or hCalendar formats (WebCalendar 1.1+)

- User authentication: Web-based, HTTP, LDAP or NIS

Subordinates

N/A

Dependencies

The webCalendar has the Apache HTTP web server for deployment of the system. The system is developed using PHP and uses the server based PHP interpreter for execution of the code. The component uses advanced GUI components, which depends on the web client features for provision of the functionality. Finally, the system uses the underlying MySQL database for storage of calendar entries and user access details.

References

The Web Calendar component is built upon the open source 'WebCalendar' http://www.k5n.us/webcalendar.php/ system.

Section 5.2.7: Task/Issue tracking Component

The task and issue tracking and ticketing component which enables a group of people to intelligently and efficiently manage tasks, issues, and requests submitted by a community of users of the WeCare system. The issue tracking system will be used both for internal tracking of issues by managers of the pilot system and for end users who wish to track actions assigned within groups of users.

Type

Component

Purpose

Provide the users the ability to track issues and perform an assigned workflow process for rectification of issues through a web based interface where information can be shared between different groups of the WeCare system.

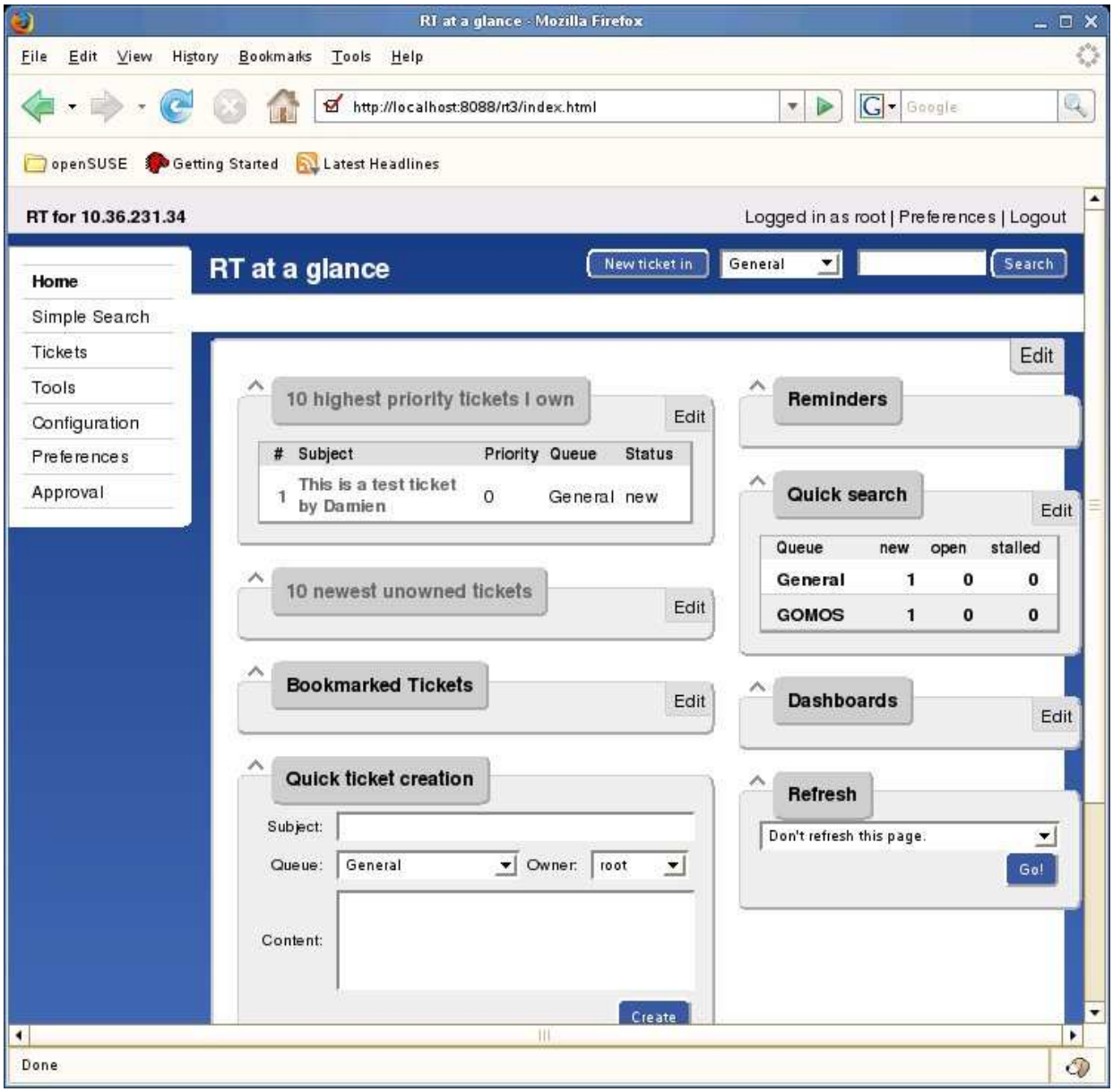


Figure 6 Task/Action item main interface

Function

The issue and task tracking component has a number of functions available to the end users and these can be broken down into several subcategories.

Critical data tracking functions are:

- Single focal point for tracking tasks, issues, knowledge, and collaboration.
- Easy to submit, assign, prioritize, search, escalate, and report on issues.
- Keeps track of each ticket's full history and metadata to help your organization better retain knowledge and analyse trends.
- Track multiple projects for multiple teams within a single installation.

- Tracks critical system metadata, including time spent per action, due dates, and estimated time to completion.
- Record private comments that are not available to end-users.
- Web interface comes complete with an intuitive "iterative" search interface that allows end users to construct complex queries by pointing and clicking within their web browsers.
- Users can save and edit queries later, using their browser's "bookmarks" feature.

End user functions are:

- Provides a simple, self-service interface that allows end users to view their own active and resolved tickets online.
- Web interface is designed to be easy to use from any browser.
- Works with Windows, MacOS or Unix.
- Fully compliant with Section 508 accessibility requirements.
- Web interface internationalised support.

Customisation functions are:

- Lets user define list-based and freeform custom fields to help track of tickets.
- Flexible templating system

Security functions are:

- Built-in access control system that lets administrators grant user, group, and role-based rights to users.
- Users can be made members of groups for the purpose of access control. These groups are hierarchical, making it easy for administrators to manage access to the system.
- Rights can be granted globally or only to specific parts of system.
- Users can be allowed the right to delegate rights that they have been granted.

Subordinates

N/A

Dependencies

The issue and task tracking system has the Apache HTTP web server for deployment of the system. The system is implemented using Perl and has multiple Perl module dependencies which must be configured on the web server. These dependencies can be automatically configured and downloaded from the CPAN web site. A Perl interpreter is required on the Web server for execution of the component.

References

The issue and task tracking system component is built upon the open source ‘RT’ http://bestpractical.com/rt/ system.

Section 5.2.8: Chat Component

The chat component provides real time chat (instant messaging) between the community of users of the WeCare system. The chat component of WeCare uses the widely adopted open standard and protocol for instant messaging which is known as XMPP (also called Jabber).

XMPP is an open, XML-based protocol originally aimed at near-real-time, extensible instant messaging (IM) and presence information (e.g., buddy lists), but now expanded into the broader realm of message-oriented middleware. It was developed by the Jabber open-source community in 1999. Built to be extensible, the protocol has been extended with features such as Voice over Internet Protocol and file transfer signalling.

Type

Component

Purpose

Provide chat functionality between multiple users in parallel.

Function

The chat component has a number of functions:

Critical data tracking functions are:

- Web-based administration panel
- Plugin interface
- Customizable
- SSL/TLS support
- User-friendly web interface and guided installation
- Database connectivity (i.e. embedded Apache Derby or other DBMS with JDBC 3 driver) for storing messages and user details
- LDAP connectivity
- Platform independent, pure Java
- Full integration with Spark Jabber client

Subordinates

N/A

Dependencies

The chat component runs as a separate web service to the HTTP web server to the Tomcat application server. However it depends on the common underlying MySQL database for

storage of user details and messages. The system also requires a JRE for execution of the components server.

References

The chat system component is built upon the open source 'OpenFire' http://www.igniterealtime.org/projects/openfire/ system.

Section 5.2.9: Search Component

The search component provides a complete search facility of all the content stored within the WeCare system independent of the component that the content was created with. To enable text content within each of the WeCare subcomponents to be searchable, the defined WeCare interface for searching must be implemented by each subcomponent. Each subcomponent will implement triggers within their associated tables in the underlying database. Once a trigger is invoked the implemented web service passes the details to the search component for addition to the overall WeCare text index. Details on the search interfaces definitions are supplied in the supporting WeCare ICD.

Type

Component

Purpose

Provide search of all content within the system.

Function

The chat component has a number of functions which can be broken down into several subcategories.

Scalable functions are:

- over 20MB/minute on Pentium M 1.5GHz
- small RAM requirements -- only 1MB heap
- incremental indexing as fast as batch indexing
- index size roughly 20-30% the size of text indexed

Searching type functions are:

- Ranked searching -- best results returned first
- Many powerful query types: phrase queries, wildcard queries, proximity queries, range queries and more
- Fielded searching (e.g., title, author, contents)

- Date-range searching
- Sorting by any field
- Multiple-index searching with merged results
- Allows simultaneous update and searching

Subordinates

The system search and indexes each of the content generated by all of the integrated component in the WeCare system. Each component uses a different set of tables and fields in the underlying database which the search system communicates with.

Dependencies

The search system uses the Apache Tomcat web server for execution of the application. The system is developed using Java Servlet and JavaBean technology where specially formatted file system for indexing storage. The system also requires a JRE for execution of the components server.

References

The search system component is built upon the open source 'Lucene' http://lucene.apache.org/core/ system.

Section 5.2.10: Help Component

The help component provides a complete help system within the WeCare system.

Type

Component

Purpose

Provide help on all content within the system.

Function

The help system provides an overview of all features of the component and access to a list of 'how to' most common features.

Help functions are:

- Navigation of a list of 'how to's.
- Standard HTML interface of contents.

- Accessible from all components.
- User can search help contents for specific entry.

Subordinates

Docbook is the system that is used to develop the help manual. The system provides a custom XML format for the definition of the online administration and user help. The XML Schema format of DocBook is an internationally recognised standard. The developed help documentation can be transformed into multiple format including online HTML and PDF formatted manuals. The use of a common definition for all manuals guaranteeing up to date manual in all major formats.

Dependencies

The help system uses the Apache HTTP web server for deployment. Standard HTML formatted pages are delivered to the users web browser for displaying help.

References

The help system component is built upon standard web technologies for displaying HTML formatted pages.

Section 5.2.11: Single Sign On User Management Component

The user management component provides a single mechanism for user access to features within the WeCare system independent of subcomponents. It manages each of the security systems of each of the components from a single component thereby providing a single login capability for the WeCare system.

Type

Component

Purpose

Manage users and access to sub-components.

Function

The user management system has a number of functions:

- Add/Remove Users to the system.
- Add/Removes Groups in the system.
- Assign Users to Groups.
- Define which components the user has the rights to use.
- Define search restrictions on content.
- Password assignment and management.
- User details, external emails etc.
- General configuration required for each sub component in the system.
- Communicate with security systems of each of the components within the system.
- Logging of users access to the system.

Subordinates

N/A

Dependencies

The user management system interfaces with each of the components user management system to provide a common method for user management. User addition and modification details are communication via the standardized WeCare web service interface to each of the subcomponents for user registration and updates. The system is deployed using the Apache Tomcat Application Server and is developed using Java technologies. It therefore relies on a JRE on the web server for execution of the system. The system relies on an underlying MySQL database for the storage of user details, access rights and passwords.

References

N/A

Section 5.2.12: WeCare Portal GUI Component

The WeCare GUI component provides a single mechanism for access to all the components of the WeCare system. It is the main interface that is presented to the user when the user logs into the system. It also provides features such as access to general information on the system and the ability to contact administrators.

Type

Component

Purpose

Main interface to the WeCare system.

Function

The WeCare component has a number of functions:

- Web 2.0 interface for access to components.
- Interactive add/removing of components.
- Customised interface on a per user basis.
- List of favourite documents for each user.
- Look and feel in ESA style.
- Provision of search text box for searching of content entered via components.

Subordinates

N/A

Dependencies

The WeCare GUI component uses the Apache Tomcat Application Server and is developed using Java technologies. It therefore relies on a JRE on the web server for execution of the system. The system relies on an underlying MySQL database for the storage of user preferences and favourites. The interface depends on the functionality provided by web browsers including HTML interpretation, Javascript execution and XMLHttpRequest processing for updating the GUI in real time.

References

N/A

Examples

Below are the customized portals used for the Spanish Irish and Dutch trail, please note that the underlying technology is the same.

Figure 7 Irish portal

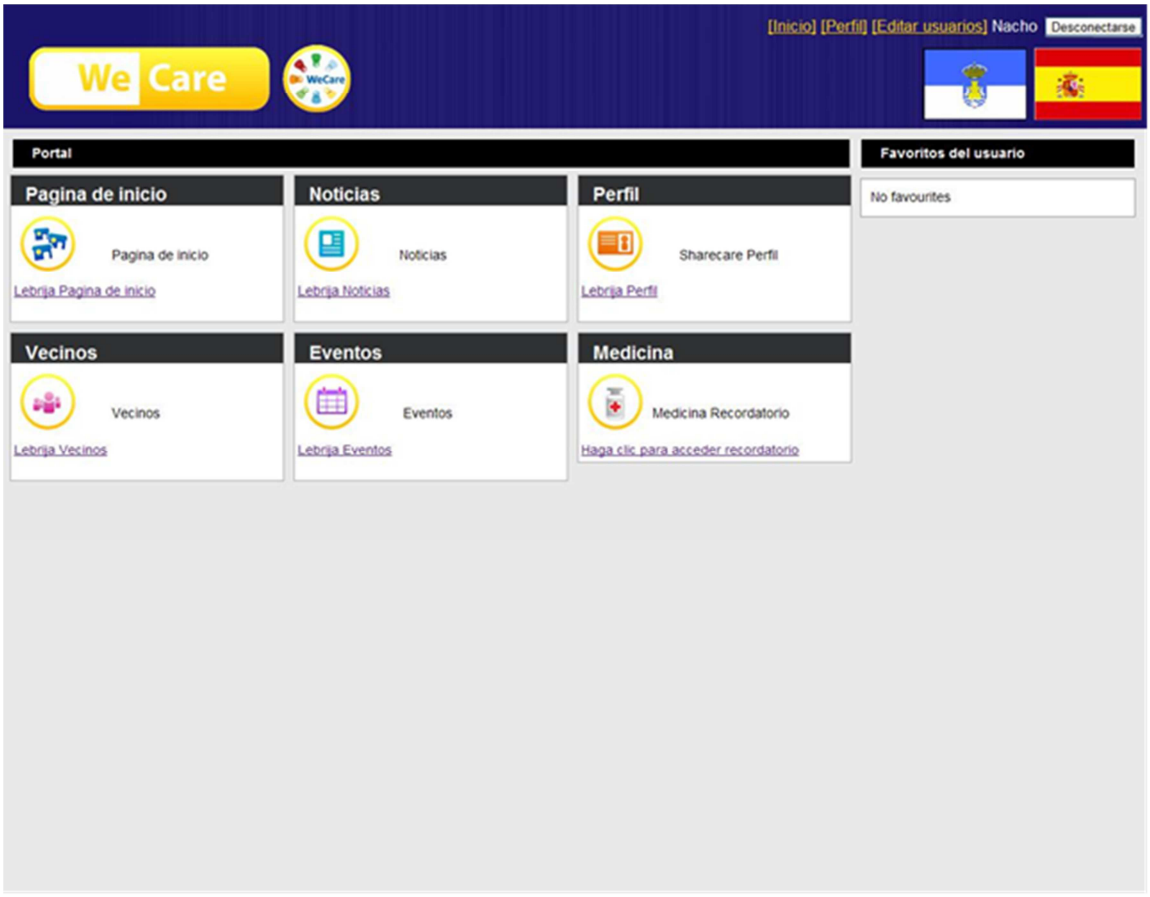


Figure 8 Spanish portal

Figure 9 Dutch Portal

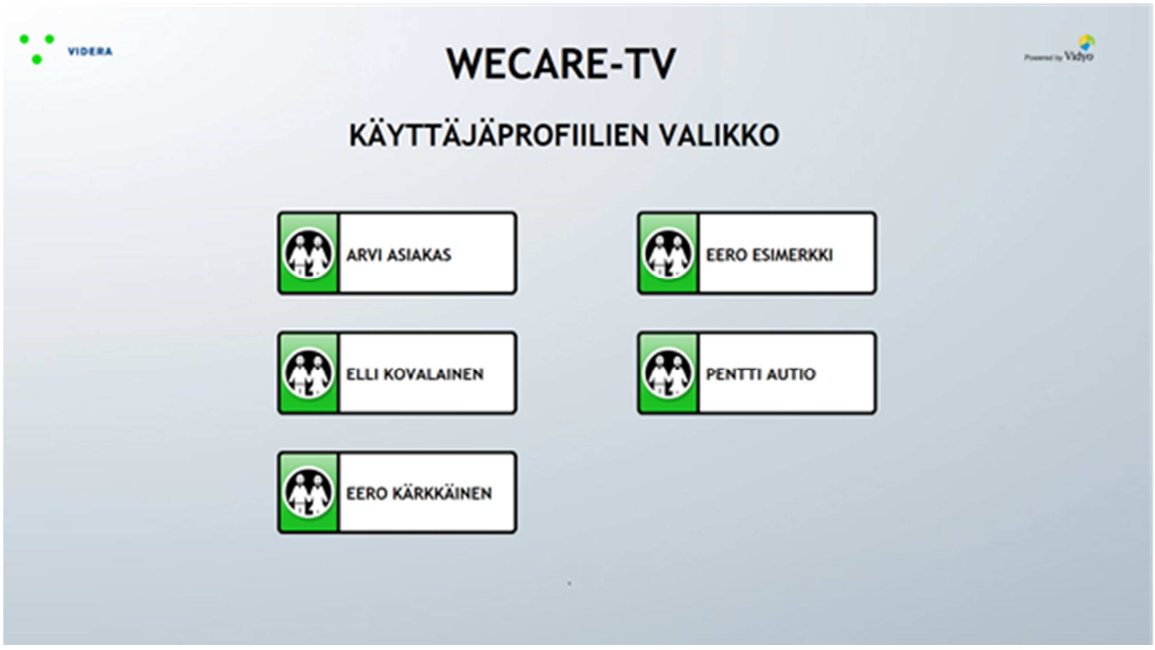
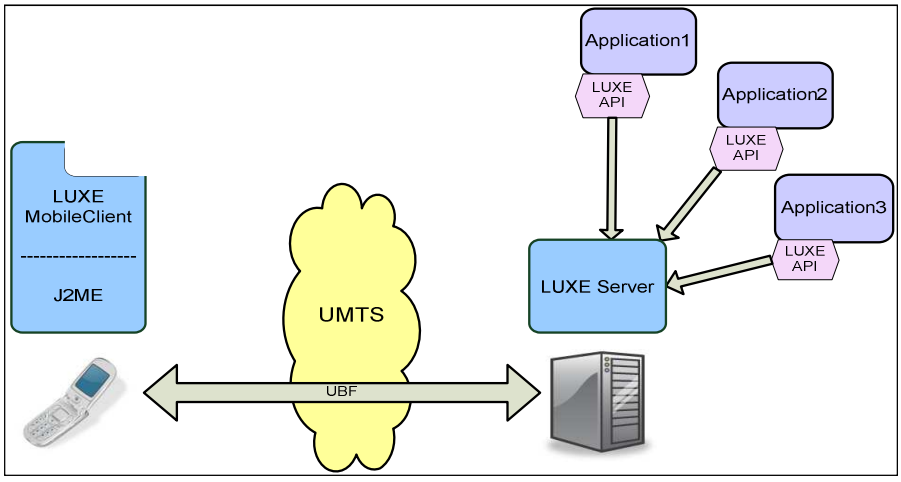


Figure 10 Finland portal – WECARE-TV

Section 5.2.13: Mobile GUI Component LUXE

The mobile GUI component 'LUXE" provides mobile access to the components of the WeCare system. It is the interface that is presented to the user on his mobile. It provides the WeCare system a way to alert users (wherever they are). It also provides the user access to the system via his mobile.

The LUXE mobile client can provide access (service) to multiple LUXE applications. An Application can be a single WeCare component or a set of WeCare components.



Type

Component

Purpose

Mobile interface to the WeCare system.

Alerting a user.and provide access to the Wecare system.

Function

The Mobile Gui Component LUXE

- A J2SE library provides the API to build on or more LUXE applications (services) which can be handled by one mobile client per user.
- LUXE applications (services) can push screens, edit-boxes and alarms to the LUXE mobile client
- LUXE applications (services) can get access to mobile functionality like initiating a phone call.
- LUXE application receives events of all user interaction on the mobile LUXE client.
- LUXE application can start the mobile client remotely via SMS when needed.

Section 5.2.14: Neighbourhood - Calendar

Type

Component

Purpose

In this component the users can start activities based on calendar dates. In the events calendar there's an overview of all the activities in the neighbourhood. In the standard view there's an overview of the events in the current month.

There is a navigational system available for the months and the days

Activities become visible by clicking the chosen day

The creator of an event or an admin of the pilot group can also edit or delete an event on this page.

Function

Mainly connects all active components within the neighbourhood, forms the central switching board. Control of content by users for reasons of liability.

Dependencies

- Connected, controlled and fed by the Simac/ShareCare Neighbourhood engine
- The level of authority of the user
- The member's list
- Relevant activities

Example

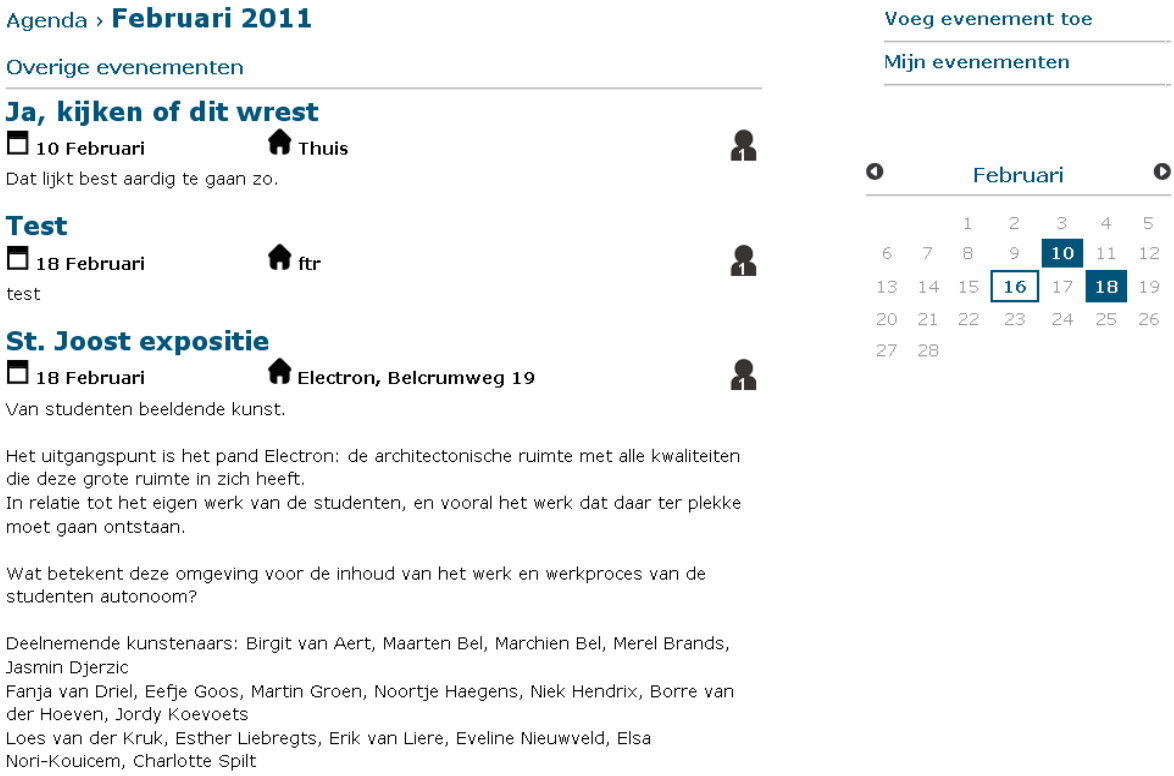


Figure 11 example

Section 5.2.15: Neighbourhood Forum

Type

Component

Purpose

To enable the discussion among the members of neighbourhood site

Function

Each member can start a discussion via the forum module

Each member can edit or delete only his of her own messages

The administrator can manage all posted messages

Dependencies

Connected, controlled and fed by the Simac/ShareCare Neighbourhood engine

The member's list

The level of authority of the user

Example

Figure 12 example

Figure 13 Example new forum entry

Section 5.2.16: Neighbourhood Starting page

Type

Component

Purpose

Main interface to the Neighbourhood system.

Function

Welcome to users

To present the main settings and activities in one glance

Dependencies

Connected, controlled and fed by the Simac/ShareCare Neighbourhood engine

All other components at or below the Neighbourhood level

Examples

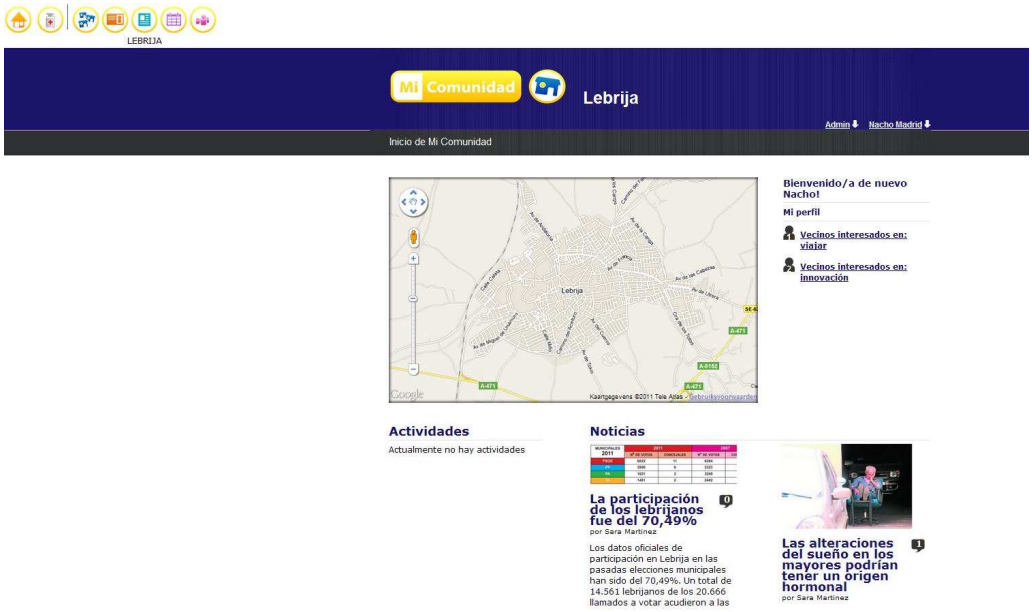


Figure 14 The neighbourhood start page offers the users an all-in-a-glance overview of their vicinity.

Section 5.2.17: Care Site Calendar

Type

Component

Purpose

Calendar system organized around one person.

Function

To present an overview of activities

To initiate activities

To modify activities

To present an overview in different time scales (week, day, month)

Dependencies

Connected, controlled and fed by the Simac/ShareCare Care Site engine

The level of authority of the user

The member's list

Examples

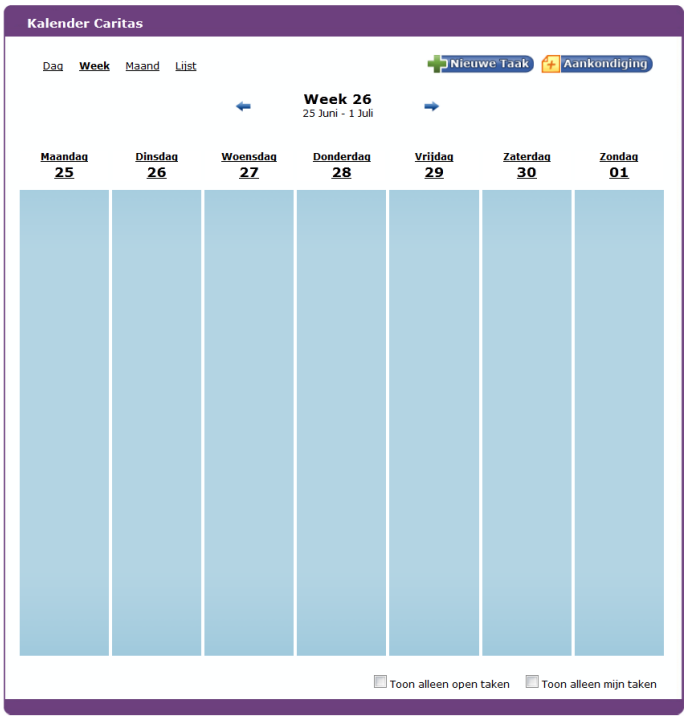


Figure 15 [week page of calendar Care Site]

Section 5.2.18: Care Site - Forum

Type

Component

Purpose

To organise a Forum system around one person

Function

Communicate open and accessible the situation of the central user

Offer an overview (notice board) of topics

Dependencies

Connected, controlled and fed by the Simac/ShareCare Care Site engine

The member's list

Topics

Reactions of users

Examples

Figure 16 [notice board]

Section 5.2.19: Care Site - Profile

Type

Component

Purpose

To organise a Profile system around all users individually

Function

Show and edit the main data around each person

Show a picture

Use data for messaging and communication

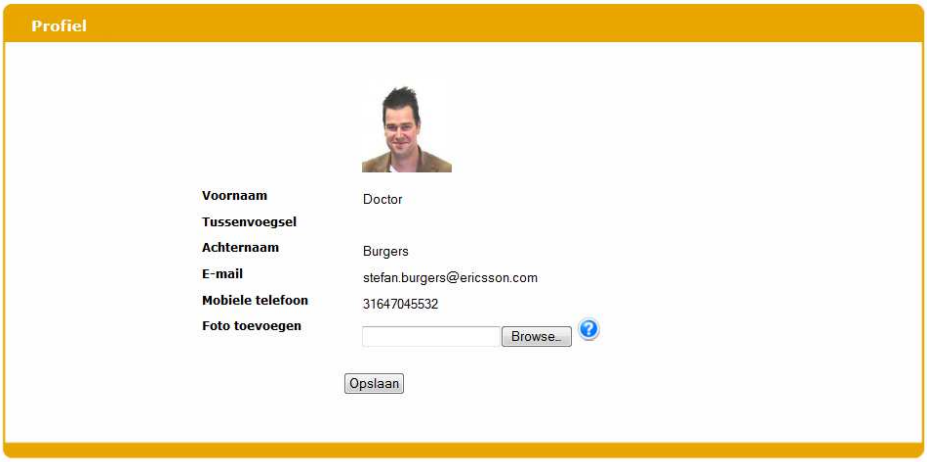
Dependencies

Connected, controlled and fed by the Simac/ShareCare Care Site engine

The member's list

Administrator's selection of data to be used for messaging and communication

Examples

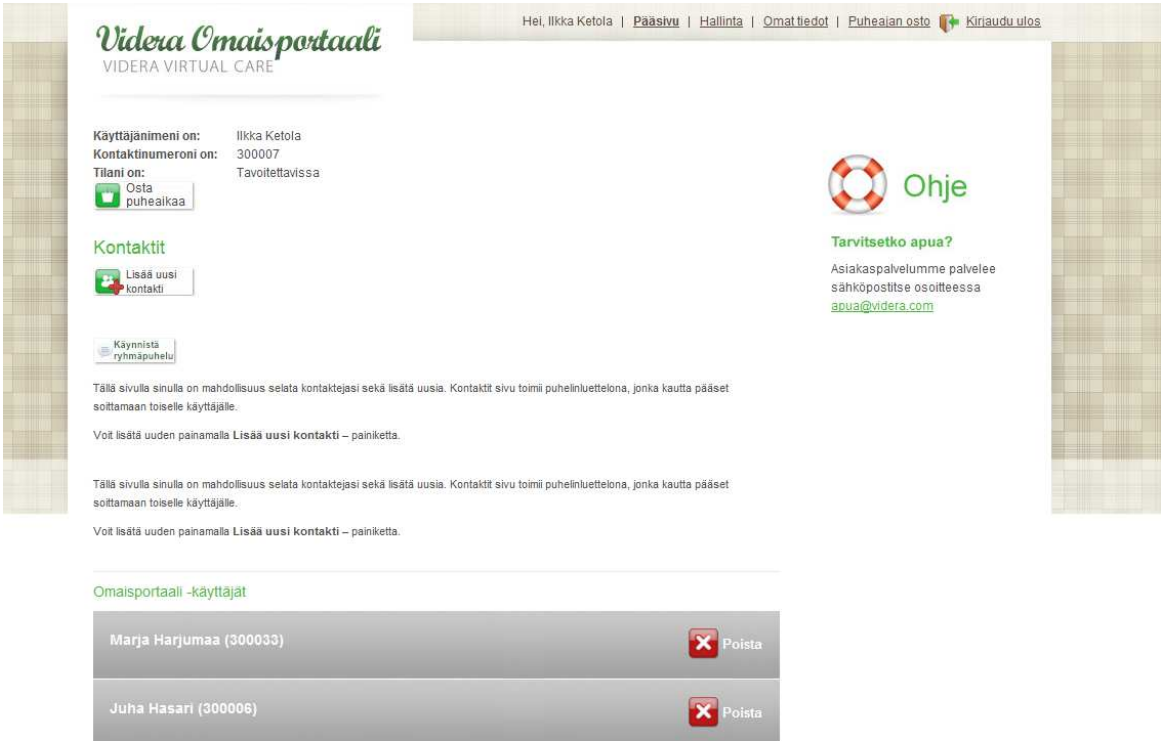


Section 5.2.20: Family Portal (End user Component)

Family Portal Video offers the WeCare network users a secure HD Videoconferencing connection between two or more participants. Users meet in a virtual meeting room where they can see and hear each other.

The login page prevents unauthorized access to the WeCare Family Portal system. Users that have a valid account will be transferred to the family portal component once logged into the system and validated.

The family portal is designed to allow one user contact other users with a click of the button. The portal is always personalised to show only the defined (and wanted) contacts.



The user has the possibility to directly add and delete contacts without the need of a separate administrator.

Section 5.2.21: Videoconferencing view

Figure 17 Videoconferencing view

During the videoconference, both (or all) parties hear and see each other and have the option see own image in symmetrical view (as above), as a mini image or none at all. The image quality can be up to 720p at 25 fps. Beside the image captured by an USB camera or other video capture device,

Section 5.2.22: Videra Touch Screen Video (End user Component)

Touch Screen video offers HD Videoconferencing connection between two or more participants. Users meet in a virtual meeting room where they can both see and hear each other. The user interface is tailored for touch screens.

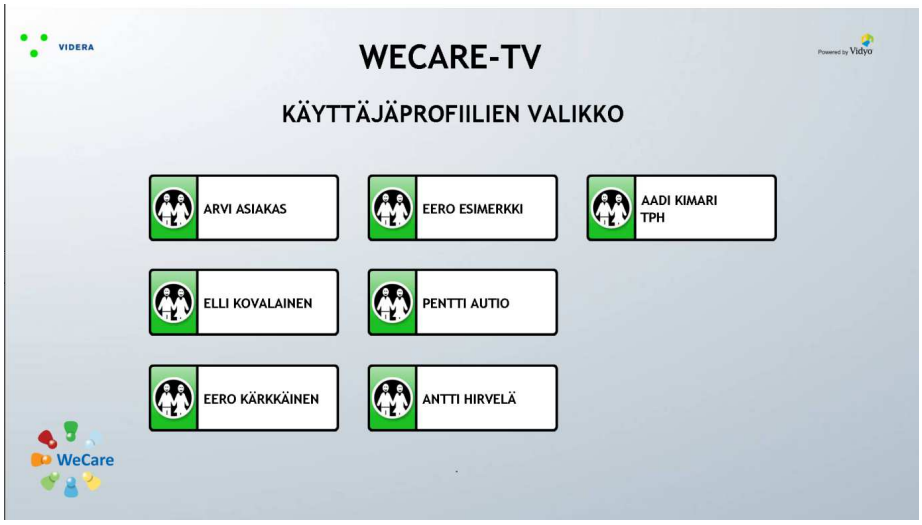


Figure 18 GUI for selection of people to participate in the video conference

The main interface of the touch screen offers an optional login screen to support user profile selection. This is practical in caring centres, where one unit can change users periodically.

The view is branded according to the specification of WP3 and Caritas Foundation.

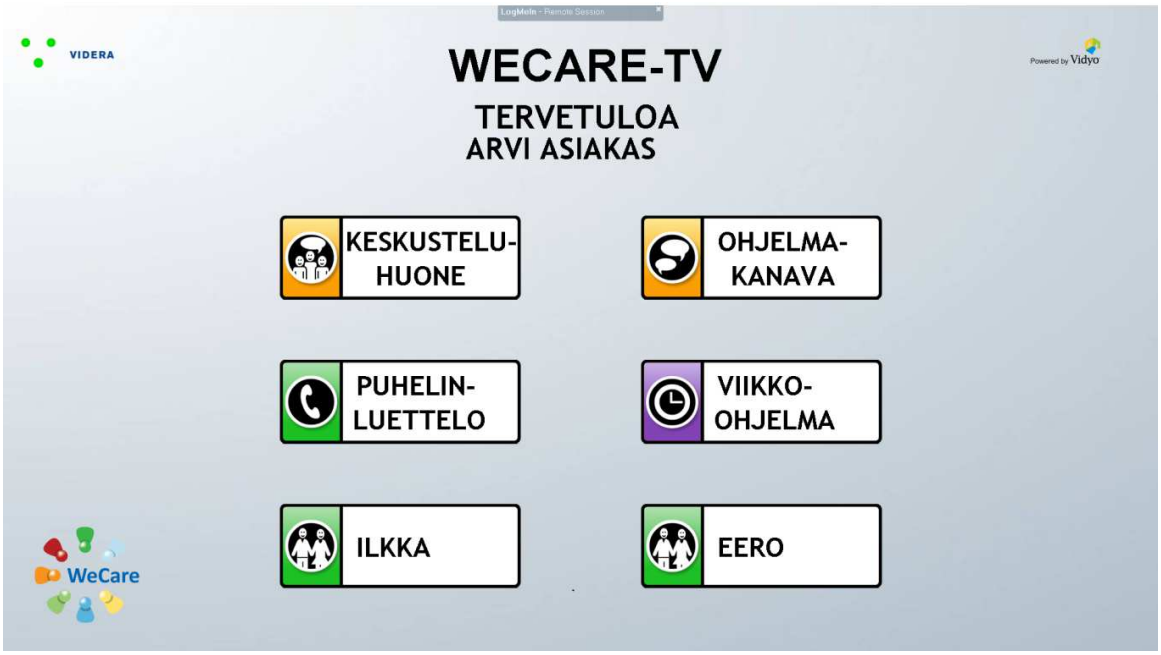


Figure 19 WeCare GUI for user login to video system

The touch screen unit offers an easy-to-use user interface, which can be customized both per user group or per user. Above is the branded version, which was agreed to fit in with the needs of the interest groups of the Finnish pilot.

The menu gives access to the WeCare chat room (where up to 50 people can meet), the WeCare programming channel (where broadcasted, interactive WeCare activities take place) and to a phonebook. It is also possible to add direct call buttons to the main view.

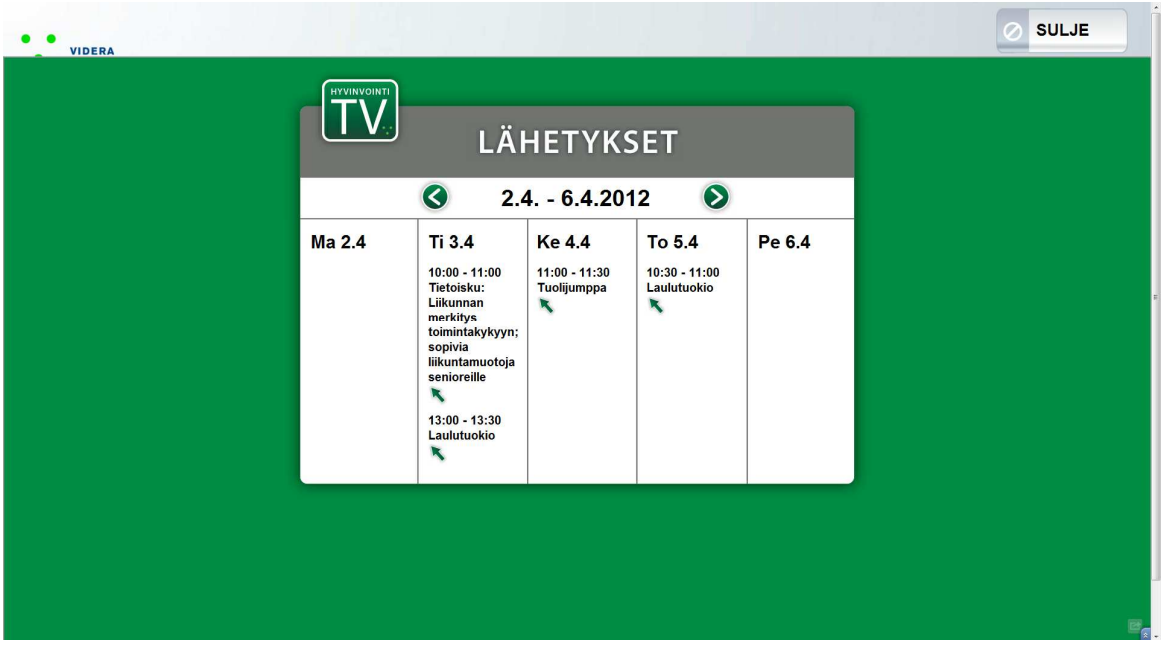


Figure 20 WeCare Program Calendar

During the execution of the pilot, a program calendar was introduced to the main view. The calendar gives an overlook of the forthcoming WeCare activities.

Phonebook ("Puhelinluettelo") lists all the users of WeCare Portal. The users can be called via videoconferencing by clicking the button on the screen.

Gray circle icon means that the user is offline and cannot be reached.

When the user is **online**, the circle icon turns **green.**

When the user is **busy**, the circle icon turns **red.**

Multiple page division. If there are more contacts that can fit to one screen, the phonebook is divide into several pages and the customer can browse through the pages with the arrow buttons on the right.

Exit. The user can exit the phonebook and return to the main menu by pushing the "Sulje luettelo" button.

Section 6: Developed Software components

In addition to the components which the WeCare technical partners adapted and integrated into the WeCare system as described in the previous section, several dedicated components were also developed during the project execution. These developed components were tailored to the exact needs of the end users as were defined during the requirements phase of the project. The components were fully integrated into the

system and trailed within the different countries. These developed components are described in the following subsections.

Section 6.1: Skytek customizations

This chapter list all customization and created components that were implemented by Skytek on the Wecare system due to trial specific requirements.

Section 6.1.1: General Information Component

This component provides access to external information sources such as news and sports. The system extracts information from external RSS feeds and presents them in a newspaper style layout which is easy for the end user to navigate and access details.

Type

Component

Purpose

Provides access to external information sources via an RSS interface. Visualisation of information is provided in an easy to navigate format.

Function

The general information component:

- Defined access to external RSS feeds.
- Display details in easy layout.
- Classify information into several subheading sections.
- Display information source title and summary text.
- Display time and date of linked information article
- Provide link for launching of external full article.

Subordinates

N/A

Dependencies

The component depends on the available of external RSS feeds where the information can be extracted from and visualised. Processing is performed on the server side parts of the component.

References

N/A

Examples

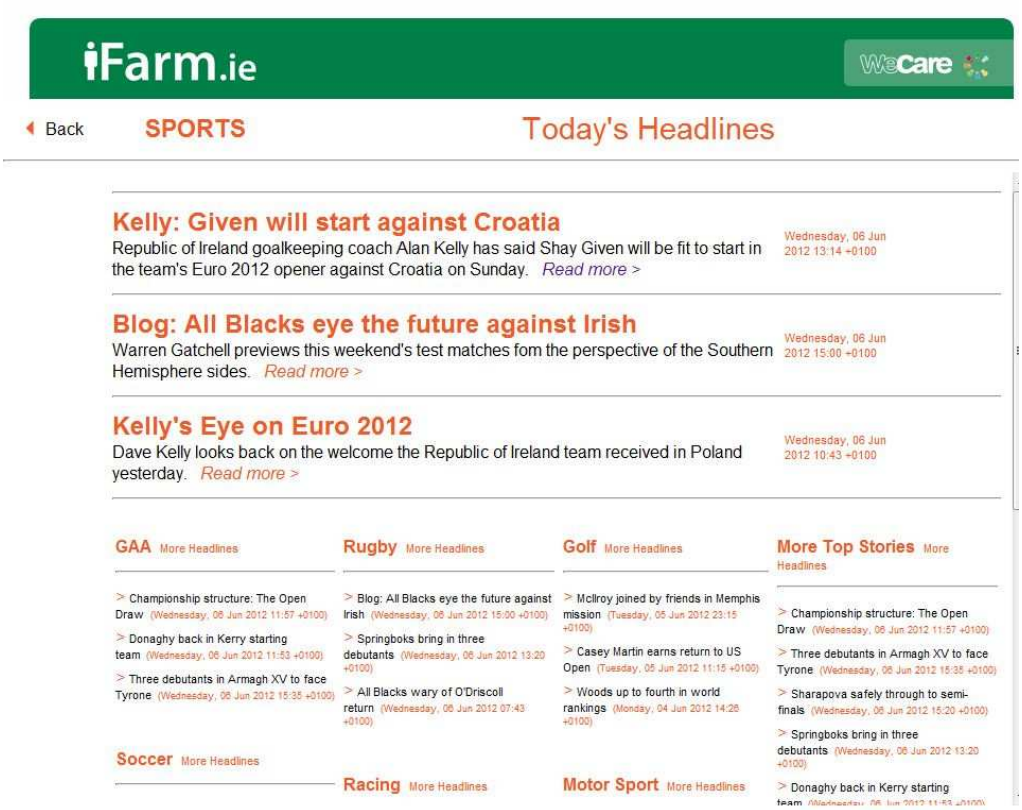


Figure 21 Information component display sports information

Section 6.1.2: Weather Services

This component provides access to weather services and information accessed from the national weather centre.

Type

Component

Purpose

Displays a variety of weather information details ranging from summary text details and charts per region through to up to date satellite imagery and detailed farming and fishing weather reports.

Function

The Irish Weather Services component main features are:

- National forecast
- Sea areas forecast
- Satellite images
- Rainfall radar

- Regional forecasts
- Coastal reports
- Charts (Rainfall, Cloud, Temperature)
- Lakes

Subordinates

N/A

Dependencies

The component depends on the available of the information from the Irish Meteorological Service through the web site www.met.ie.

References

N/A

Examples

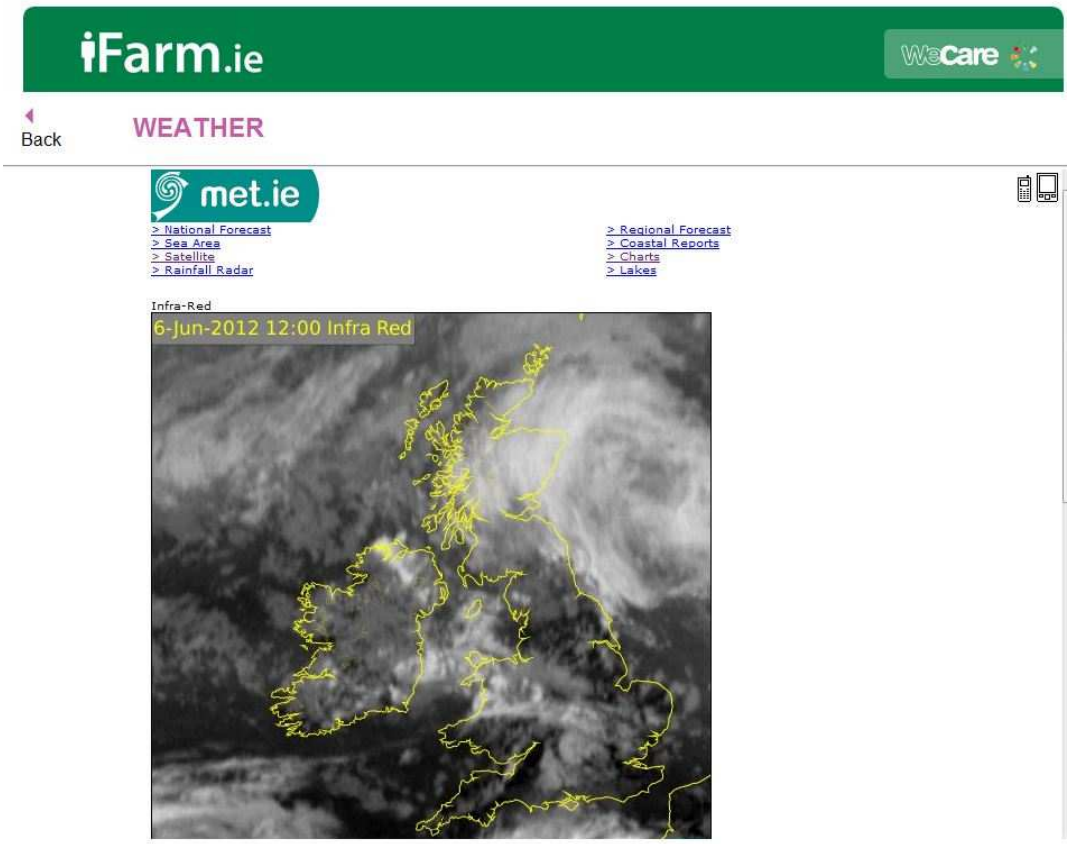


Figure 22 Weather component displaying satellite imagery

Section 6.1.3: Phone/Video Calls

This component provides VoIP phone and video calls via the WeCare system.

Type

Component

Purpose

Launches and integrates Skype based voice and phone call capabilities from within the WeCare system.

Function

The Phone/Video Calls system provides:

- Launching of a Skype client from WeCare system.
- Provides phone calls over VoIP
- Messaging capability
- Provides video calls over VoIP

Subordinates

N/A

Dependencies

The component depends on the available of the Skype client being installed on the end users system.

References

N/A

Examples

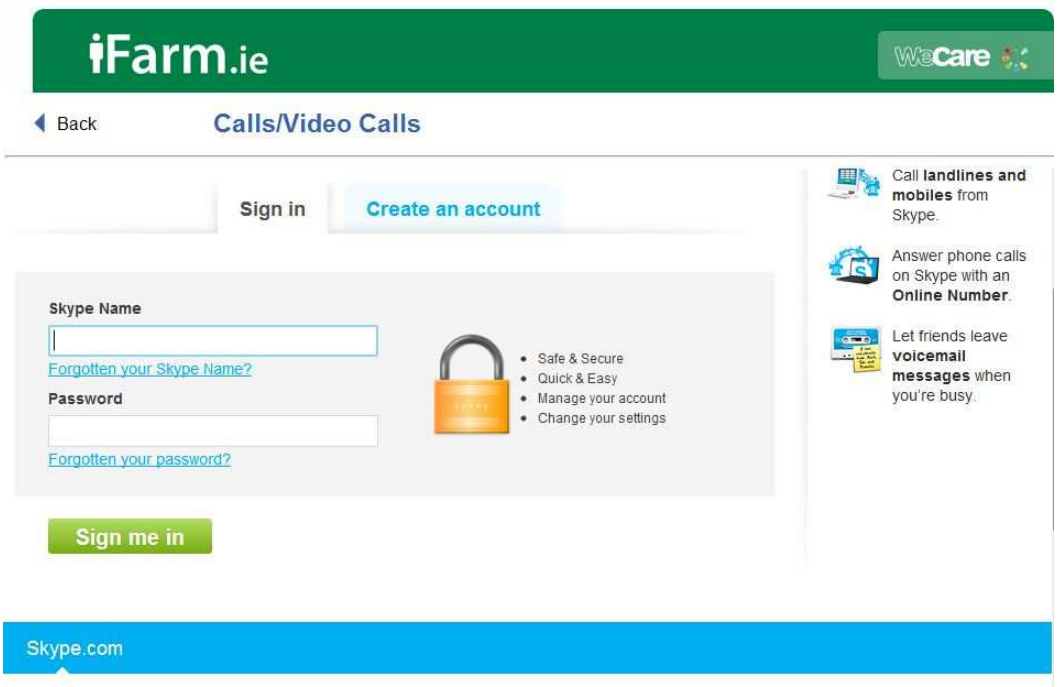


Figure 23 Phone/Video Call component in WeCare

Section 6.2: Ericsson customizations

This chapter list all customization and created component that were implemented by Ericsson on the Wecare system due to trail specific requirements.

Section 6.2.1: Consolidated Contacts List

Type

Component

Purpose

The consolidated contacts list was developed for the Dutch Trial. These had special group management requirements such as one open group called “Escamp” and user definable closed groups

Function

The CCL provides:

- Group management
  - Creation/Deletion of closed groups
  - Add/Remove users to your closed groups
  - Leave a closed group
  - Join or leave public groups
- SMS messaging to individuals or groups
- Voice call setup between individuals

- Email messaging to individuals or groups
- Join the VideoConference room.

Subordinates

N/A

Dependencies

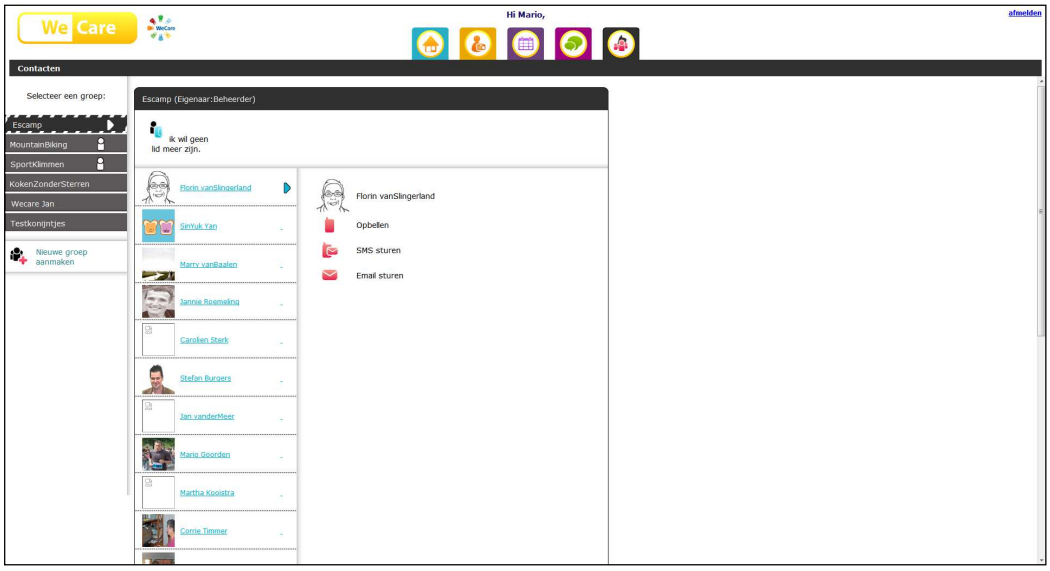
This component is heavily dependent on the ShareCare Calendar module and the Skytek user administration portal.

References

N/A

Examples

The consolidated contacts list was developed for the Dutch Trial. These had special requirements such as one open group called “Escamp” and user definable closed groups.



On the left side you have an overview of the groups. The right box contains the members of the group.

The top group ‘Escamp’ is an open group. This means that every user within the ‘Escamp’ area can become member of this group. The groups below are closed groups. Within these groups, all the contacts are shown.

A user can select one of the users and use the icons on the right to contact this person.

- Phone, The user who logged in and the user selected will be called on their mobile (or fixed phone) and linked to each other. This way a call can be setup without knowing the subscriber number (the system uses the phone-number available in the user profile, and dials both users into a two-party conference call)

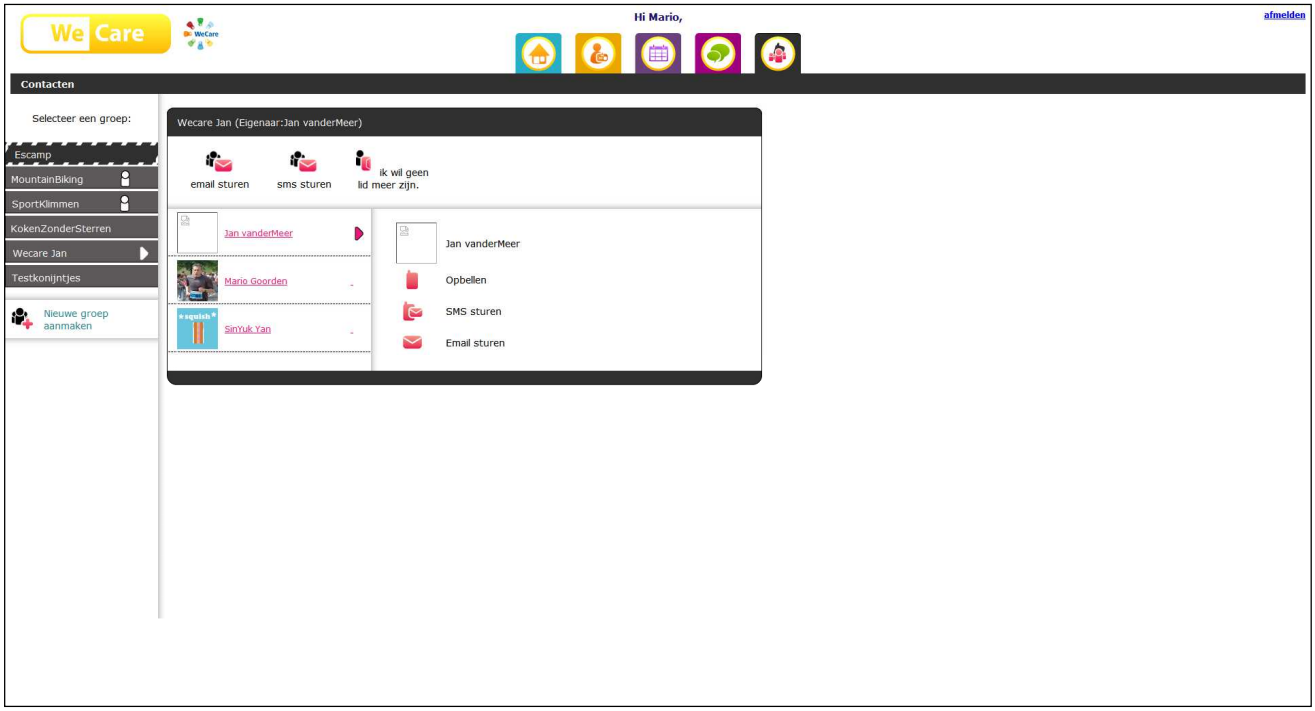
- SMS, a SMS can be send to the user selected.
- Email, an email can be send to the user selected.
- Video conference groups, four conference rooms are available

A user can be the owner of a group (white icon within the groupname) which allows this user to

- add another user to the group
- remove a user from the group
- send a SMS to everyone in the group
- send an email to the group
- Remove the group

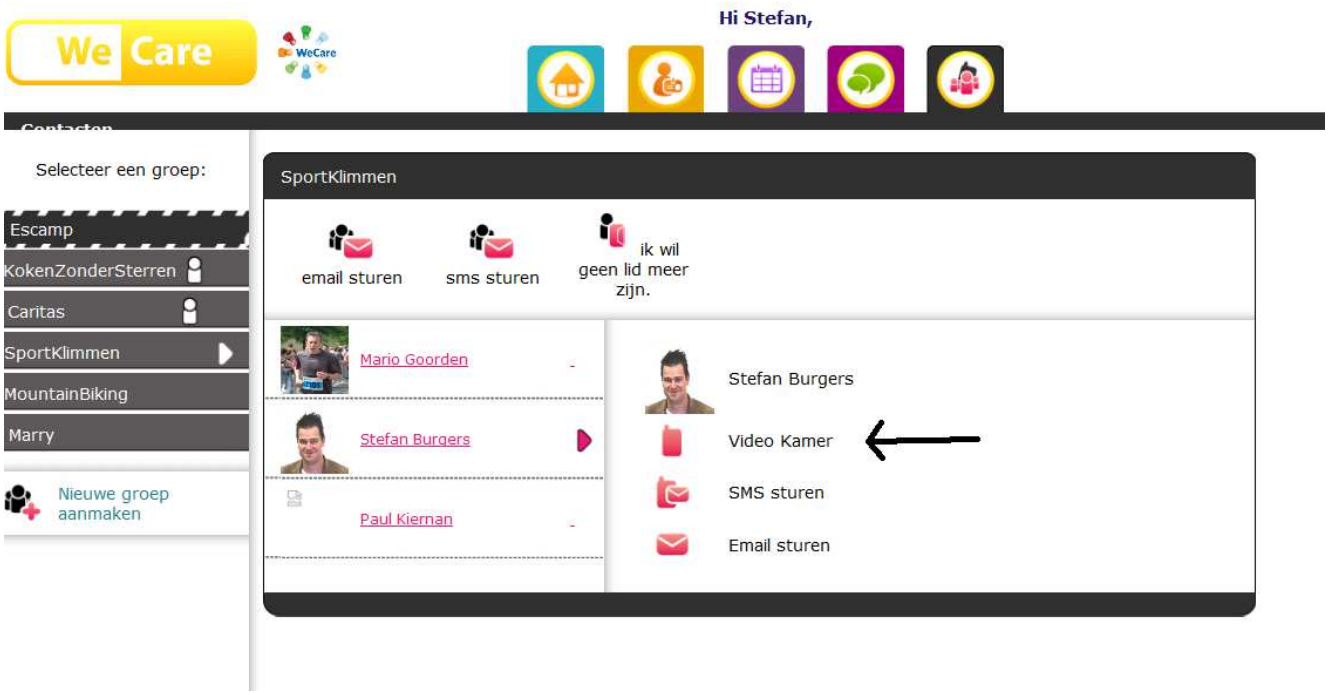
When the user does not own the group, but is a member the following options are available

- Remove himself from the group
- Send a SMS to everyone in the group
- Send an email to everyone in the group.



A user can add a new ‘closed’ group and invite users to this group.

Initiate a video conferences:



For the Dutch trail four test rooms were made available, these can be accessed by selecting your own name in the Consolidated contact list and selecting the option “Video Kamer”.

This function was included for testing purposes and was only available to some users

Section 6.2.2: Mobile application Med2Mob

Type

Component

Purpose

For the Spanish and Irish trial the medicine reminder application based on the LUXE platform described in Section 5.2.13: was heavily adjusted to fit the needs of end users.

The following adjustments were made

No native mobile client: The trial did not want to install software on the mobile device of the end-user. Alternatives had to be found for alerting the end-user. This resulted in SMS notifications and Voice call Notifications to the mobile device.

Function

The Med2Mob component provides:

- SMS notifications of user set reminders
- Voice call notifications of user set reminders
- Administration of you reminders (textual or record a voice reminder)
- Mobile client that reminds the user with more detailed information (as an option).

Subordinates

N/A

Dependencies

N/A.

References

N/A

Examples

The administration page (that makes it possible to get notifications by voice message or SMS) is depicted in figure below.

It also allows people to download the original med2mob application by typing in their mobile number an press the install button.

Figure 24 "*Module where you can set your appointments and medication reminders*"

Section 6.3: Videra customizations

This chapter list all customization that were required on the Videra system due to trail specific requirement

Section 6.3.1: Broadcasting station and the group home units

To increase the social network effect and to provide content to the WeCare pilot in Caritas Foundation, Broadcasting station unit and group home units were introduced to the network. These units were identified to be essential elements for creating a successful pilot for the WeCare concept.

The broadcasting station. Provides a standard issue of Video Conferencing room system with one 42" LCD screen on a movable stand, located in the Caritas hall. The station connected the hall to the WeCare network and enabled interactive programming to be distributed from the hall and for other network users to interact with Caritas workers and audience in the hall.

Group home units. The units were standard video conferencing mini room systems (Vidyo HD50) connected to the television sets of the group homes. The units connected the group homes of Caritas to the WeCare network.

Videra controlled the administrative components in the Finnish Pilot.

Administrative components enabled Videra to

- Control the access to WeCare network
- Control the user interfaces of touchscreen units
- Control the phonebooks of all units
- Provide reporting data of the usage.(VIS, OWA)

Section 6.3.2: WeCare environment

Videra has a web based user interface to control the WeCare environment. The UI gives control over visibility of the WeCare environment (or a part of it) to other (similar or video conferencing) environments. The same control panel gives control on both group level and individual user level.

The same control interface allows the administrator to control the UI of the touch screen units in full.

Section 6.3.3: Video Conferencing Administration

There is a dedicated control interface for controlling video conferencing specific settings of the environment (setup of the whole network, limitations of maximum users and new installations etc.). This interface is normally not needed to access after the initial setup of the environment.

Section 6.3.4: Family Portal Administration

Family portal administration panel is needed to control family portal specific variables and user creation.

Section 6.4: Family Portal (End user Component)

Type

Component

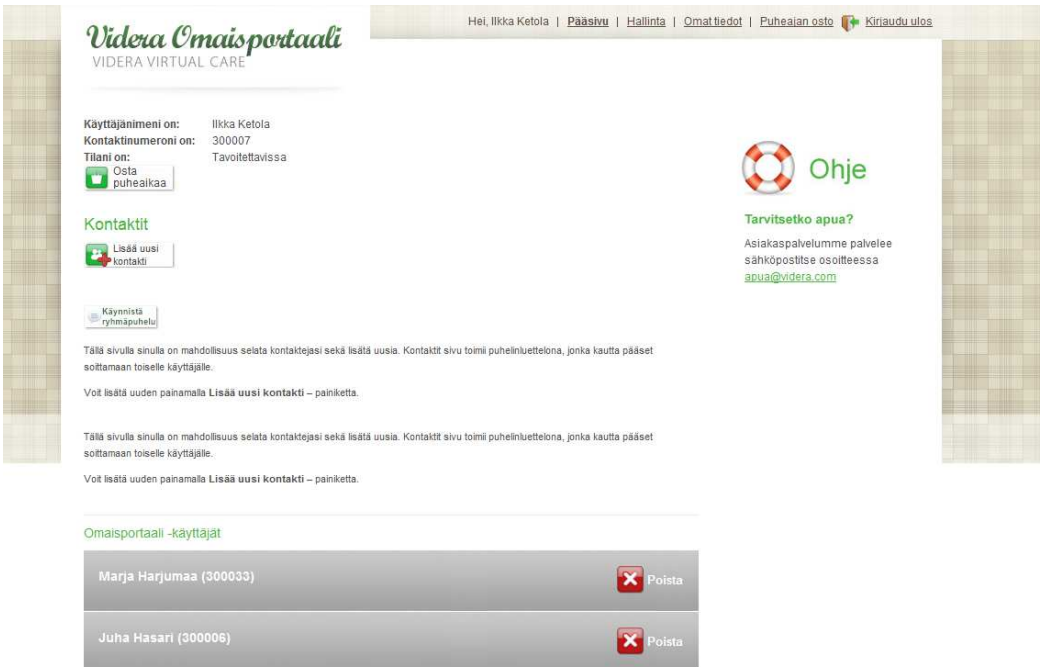
Purpose

Family **Portal** Video offers the WeCare network users a secure HD Videoconferencing connection between two or more participants. Users meet in a virtual meeting room where they can see and hear each other.

Function

- Login (authenticate)
- call user
- add user
- remove user
- check client video client status
- client functions (mute/unmute, share screen, mute/unmute video, end call)

Examples



During the videoconference, both (or all) parties hear and see each other and have the option see own image in symmetrical view (as above), as a mini image or none at all. The image quality can be up to 720p at 25 fps. Beside the image captured by an USB camera or other video capture device,

Administrative functions

Administrative components enabled to

- Control the access to WeCare network
- Control the user interfaces of touch screen units
    - layout, graphics, access, direct call buttons, url links
- Control the phonebooks of all units
- Provide reporting data of the usage.(VIS, OWA)

Videra has a web based user interface (UI) to access the features of the WeCare environment. The UI gives control over visibility of the WeCare environment (or a part of it) to other (similar or video conferencing) environments. The same control panel gives control on both group level and individual user level.

The same control interface allows the administrator to control the UI of the touch screen units in full.

There is a dedicated control interface for controlling video conferencing specific settings of the environment (setup of the whole network, limitations of maximum users and new installations etc.). This interface is normally not needed to access after the initial setup of the environment.

Family portal administration panel is needed to control family portal specific variables and user creation.

Section 6.4.1: Video Room Component

Type

Component

Purpose

The component offers an HD videoconferencing rooms which can be accessed and launched from within the final WeCare 2.0 portal. The interface is made really simple. Participants can see and hear each other via high definition video connection.

This was first introduced in the Dutch Trail and is accessible through the Consolidated contacts list.

Functions

- Join room/conference
- video client functions (meeting control during the video conferencing session).
- 720p video (H.264/SVC)
- audio
- data sharing
- See up to 8 simultaneous participants
- layout control
- exit
- up to 50 simultaneous participants
- check client status
- 4 predefined rooms
- 16 predefined login credentials made available for WeCare

Dependencies

The Video Room component requires Apache HTTP web server (for the frontend end logistics) as well as Vidyo video conferencing infrastructure (portal and router minimum) for deployment of the system. The front end code is written in Java. To successfully use the video component the end user needs to have a microphone and a web camera beside the other prerequisites of the WeCare system.

References

The Video Room Component is based on Vidyo technology and Videra Bringio.com service.

Section 6.5: Simac/ShareCare customisations

In this chapter all customisations and novelties are listed that were needed for Simac/ShareCare system to comply with the requirements of the trials.

Section 6.5.1: Logon Module for open and Closed groups

Type

Component (invisible to user)

Purpose

Designed for the Wecare Service in Spain and The Netherlands, this module was introduced to be able to manage different kinds of groups of users within the same Simac/ShareCare part of the Wecare service; in the components as described as in chapter 5 simultaneous use and access via API and portal was not possible is a satisfactory way.

Assigning of access rights, auto-correct of missing data

Marking users for further activities within the modules of the Simac/ShareCare part of the Wecare service

Internal control for support and development

Function

Connects to the ShareCare Core Dbase system (own protocol)

Connects to the front side user login (Wecare portal)

Dependencies

Data are received from the front side (portal)

Requirements for ShareCare Core Dbase system

Section 6.5.2: Open Group Module

Type

Module

Purpose

Adaptation of the Neighbourhood system, needed primarily for the Wecare Service in Spain. Can be used to and connected for future countries and requirements

Form a group of users consisting of equals

Social activities and contacts are facilitated and enabled

Control by Trial administrator

Functions

Connected via an API with front end portal for preparing and adding members

Tied to all components within the Simac/ShareCare part of Wecare

Dependencies

Front end portal (WeCare Portal GUI component) via link

Pilot group (subsystem to manage groups during trials)

Member's List (members and groups of member within the open group

Section 6.5.3: Closed Group Module

Type

Module

Purpose

Adaptation of the Care Site system, needed primarily for the Wecare Service in the Netherlands. Can be used for and connected for future countries and requirements

Form a group of users centred around one individual (primary user)

Social activities and contacts are facilitated and enabled within a safe and closed environment

Control by Trial administrator possible

Primary users have rights to control the others users

The other users can sign in and react

Functions

Connected via an API with front end portal for preparing and adding members

Tied to all components within the Simac/ShareCare part of Wecare

Dependencies

Front end portal (WeCare Portal GUI component) via link

Pilot group (subsystem to manage groups during trials)

Member's List (members and groups of member within the closed group

Section 6.5.4: Pilot group

Type

Predefined safe environment set for either closed groups or open groups, needed primarily for the Spanish Trial to enhance change and merging of groups. Due to the trial nature, much more changes and transfers are present than the handling without this pilot group construction allowed. This component is also present in the Dutch trial, though not used (active for future use). It must be seen as standard component for trial situations.

Purpose

To create a test group for the trials separated from the regular test environment

To enable to migrate and merge different kind of groups

Administrative tool

Function

Forms an easy opt-in for non-members of the trial group

Dependencies

All open and closed groups within each country version of WeCare

Section 6.5.5: Calendar

Type

Component

Note

This Calendar component is different from the Web Calendar used in the WeCare portal for some of the participating countries

Purpose

In this for the WeCare system most central component, the users in the Spanish and Dutch trails can start activities within their groups and monitor activities of other users. The Wecare Service is enhanced to communicate well with components like open and closed groups. The component is also adapted to the iPad.

Function

Tasks made as a consequence of the activities can be joined by members of the group by subscribing

Theirs is an overview of all the activities within the group

Every member can see which activities are available and to which he is subscribed

In the standard view there's an overview of the events in the current month

There is a navigational system available for the months and the days

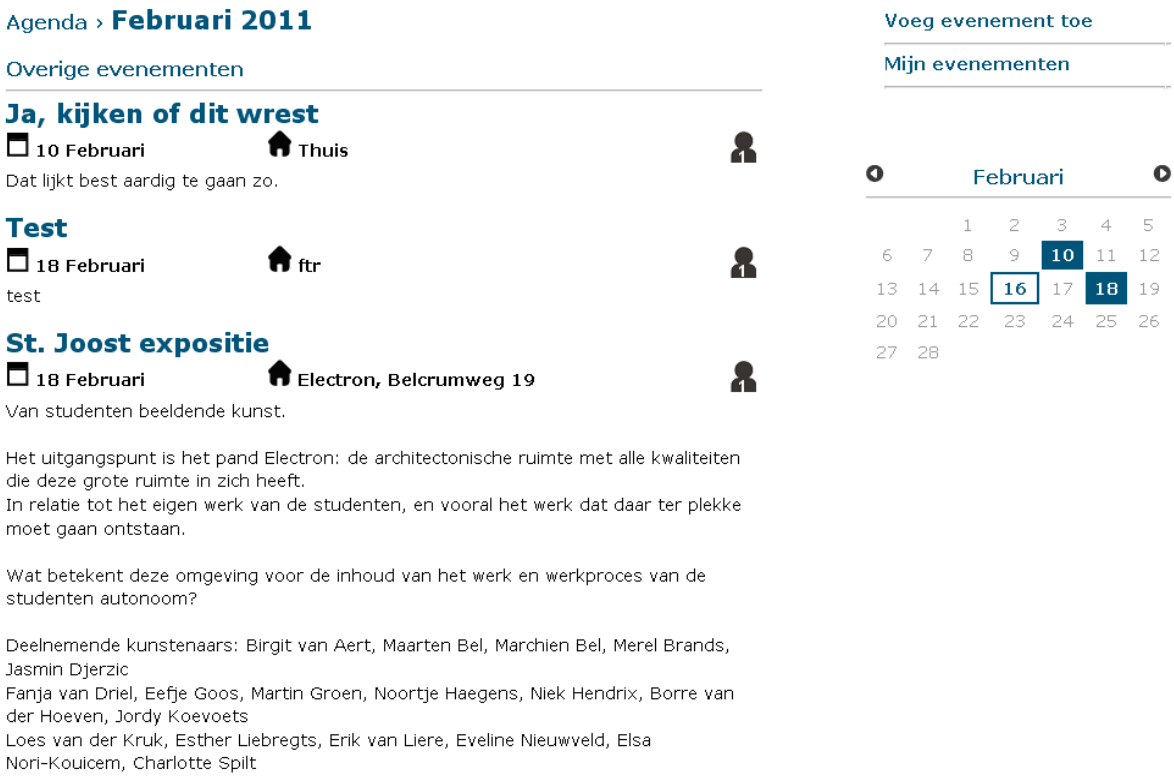
Activities become visible by clicking the chosen day

Dependencies

All active components within a group

The creator of an event the pilot group can also edit or delete an event on this page.

An administrator can control all actions and subscriptions



Section 6.5.6: Member's list

Type

Component

Purpose

The member's list is an adapted/developed component to show the user the other users in a simple way, thus allowing interaction, as needed due to the nature of the Wecare system (enhancing interaction). It was needed in both the Spanish and Dutch trials.

Function

A member, if logged in, can view the other members, their phone number and their email addresses

If selected, the list is printable

A group mail function is available to mail all or some of the members at the same time

Dependencies

Administrator's settings

Character of group (open or closed)

Needed data given by membe

Section 6.5.7: Profile page

Type

Component

Purpose

An adapted Simac-ShareCare component (split form the Neighbourhood site mentioned in section 5), needed to clearly identify and present all members in the trial as an individual

Note: the component Member's list, is a listing of other Members, whether this component focuses on each user's personal qualities

One important part of this module are the personal interests. Based on these interests, members in the situation of use in an open group, are automatically matched to other members ('Neighbours' page) and their activities.

Function

On the profile page members of the group can add and edit their personal profile

When a member starts typing in the interests box other peoples interests become available by an auto fill mechanism programmed to intelligently distinguish between existing and new tasks, to make matching easier.

Dependencies

The Member's list and the character of the group (open or closed)

Auto-selection mechanism

Section 7: Localisation

Most components described in chapters Available Software Components DesignDeveloped Software componentssupport multilingual localization

.

The trialling organization had the possibility to change the language setting.

All language setting files (resource bundles) were made available on share point <link> in the following structure:

-Localization

- -Template (containing the plain language setting files)
- -Finland (containing the Finish language setting files)
- -Netherlands (containing the Dutch language setting files)
- -Spain (containing the Spanish language setting files)
- -Ireland (containing the Irish language setting files)

Example of a language setting file:

```
   ....
#: ../../models/Member.php:84
#: ../../models/Site.php:119
msgid "Error|Lastname is required"
msgstr "Es necesario introducir el apellido"

#: ../../models/Member.php:99
#: ../../models/Site.php:133
msgid "Error|Invalid email"
msgstr "Dirección de correo electrónico incorrecta"
   .........
```

Section 8: Trials

The entire WeCare service set consists of an abundance of possible functionalities, referred to as components.

Each Trial can make a selections, mainly to make a logical ‘pick’ of the most consistent and logical combinations of modules appropriate to the specific trial user group. As a general rule all functionalities are available to all countries, though each country uses its own ‘pick & mix’.

The ‘pick & mix’ concept was at the start of the project and is described in detail in deliverable D2.2.

Trial specific delivery descriptions (D2.3_Spain, D2.3_Ireland, D2.3Finland, and D2.4_Netherlands), describe the country specific Trial delivery, which is a mainly a ‘pick and mix’ of components as described in D2.2 + custom trial requirements. Deviations to the pick and mix are explicitly mentioned in the Trial specific delivery descriptions.

Section 8.1: Runtime Status and usage statistics

For the pilot phase of the WeCare system, both the runtime status and usage statistics are to be captured. Remote access to each of the pilot sites servers will be provided to the WeCare technical partners of the consortium for access to the status and generation of usage statistics. Two separate mechanisms will be deployed for information capture:

1. Network and application monitoring.
2. Logging by individual components and the underlying web application servers

Section 8.1.1: Nagios

To provide for WeCare consortium personnel to access and monitor the network usage, application uptime and system metrics during the deployment and pilot activities the WeCare system shall integrate the open source Nagios (http://www.nagios.org) tool. Through this integration personnel will have real time access to detailed information on the pilot usage, available, and related statistics on the WeCare system through remote access to the network and detailed ‘Nagios’ reporting facilities.

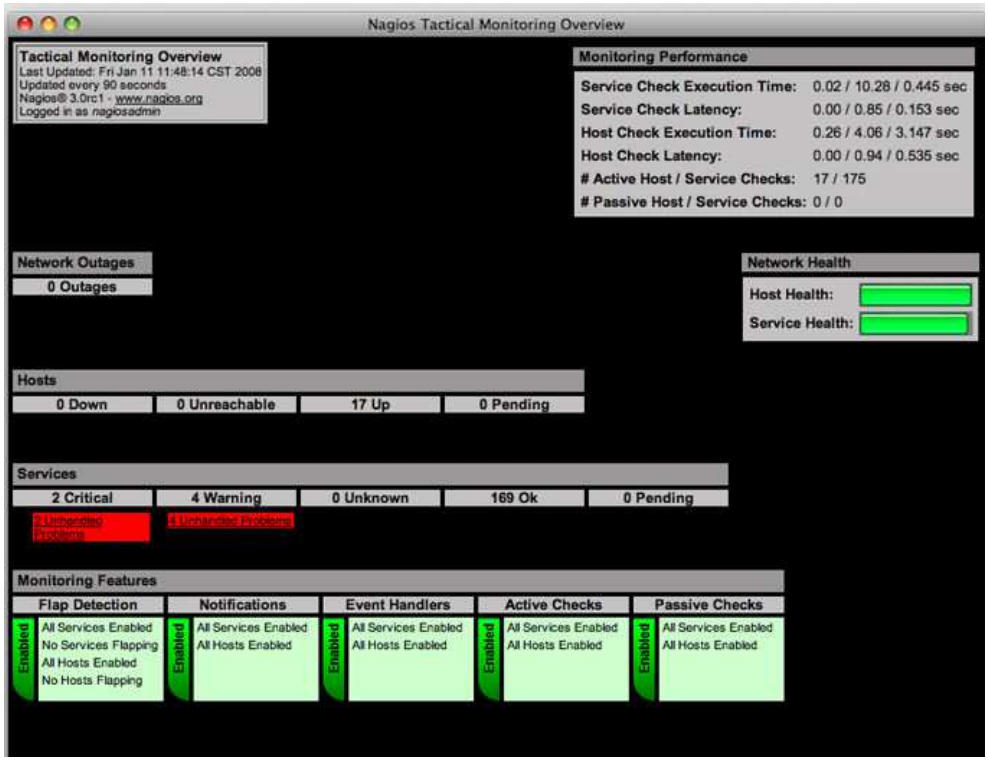


Figure 25 Example of Nagios network monitoring GUI to be used during WeCare pilot utilisation

The remote access to the WeCare system will be provided through a VPN connection and terminal servers. Through the use of WeCare log files in addition to the log files generated by both the underlying HTTP(S) and Web application servers detailed statistics on component usage will be generated.

Section 8.1.2: OWA

To support the analysis of usage of the WeCare system during the trials a sophisticated automated logging infrastructure was implemented. The logging system was based on the open source tool called Open Web Analytics (OWA) where more details can be found at http://www.openwebanalytics.com/. The logging infrastructure uses a remote server provided by Ericsson to store details of log entries. The logging server was located at http://lab-d.w3innovate.nl/owa/

Details that were transmitted to the logging server during the usage of the trials included:

Action tracking for selection of subcomponents.

During the project it was decided that action tracking would be exposed to the trialling organizations, for measurement purposes, WP2 created a web form making it possible to get the action track data. http://lab-d.w3innovate.nl/owa/wecare

On the web form the Trail organization can set, “the trial”, “start date” and “end date” for which they would like to see the history of actions tracked. As depicted in the following screenshot

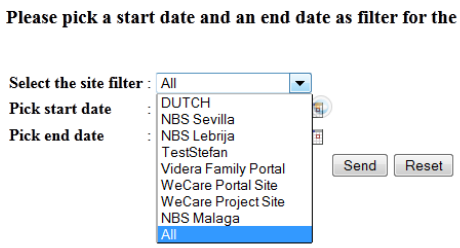


Figure 26 (http://lab-d.w3innovate.nl/owa/wecare)Select trail scope and start and end date

After pressing the send button the historical actions tracked are presented in comma separated format (for the provided end/start date/trail). As depicted in the following screenshot.

02-10-2011 11:35:17 AM,login page,nj72656974736d61,login,1,DUTCH
02-10-2011 11:38:06 AM,login page,nj6f6b6572656974736d61,login,1,DUTCH
02-10-2011 11:38:10 AM,login page,nj6f6b6572656974736d61,login,1,DUTCH
02-10-2011 11:42:04 AM,login page,nj6f6b6572656974736d61,login,1,DUTCH
02-10-2011 11:42:32 AM,login page,nj6f6b6572656974736d61,login,1,DUTCH
02-10-2011 12:02:49 PM,login page,nj6f6b6572656974736d61,login,1,DUTCH
02-10-2011 12:03:23 PM,login page,nj6f6b6572656974736d61,login,1,DUTCH
02-10-2011 12:03:30 PM,login page,nj6f6b6572656974736d61,login,1,DUTCH
04-10-2011 01:43:21 PM,login page,nm756666696e31303633,login,1,DUTCH
05-10-2011 11:30:08 AM,login page,nj616e6e696572,login,1,DUTCH
05-10-2011 11:30:18 AM,list page,nj616e6e696572,list,1,DUTCH
05-10-2011 11:30:24 AM,contactlist,n,showlist,1,DUTCH
05-10-2011 11:30:31 AM,contactlist,n,showlist,1,DUTCH
05-10-2011 11:30:39 AM,contactlist,n,showlist,1,DUTCH
05-10-2011 11:30:50 AM,contactlist,n,showlist,1,DUTCH
05-10-2011 11:31:04 AM,list page,nj616e6e696572,list,1,DUTCH
05-10-2011 11:31:05 AM,profile page,nj616e6e696572,profile,1,DUTCH

Figure 27 Action track results EXAMPLE

The comma separated format contains the following data

*[**timestamp** when tracked action occurred],[**module** tracked action occurred],[the **user** (user encoded for privacy reasons) that triggered the tracked action],[the **action** that was trigger by the user],[a **trail name**, indicating the under what trail the action was triggered]*

More details about OWA can be found at http://www.openwebanalytics.com/

Section 8.2: Pilot studies by country

The following subsections provides a overview of which components of the WeCare system will be deployed and used for each of the end user trials that will take place within the WeCare project. For each country a mapping of all WeCare system components is shown along with whether they will be active for this trail.

Section 8.2.1: Finland

The initial WeCare system end user trail will take place within Finland. The following table shows the WeCare system components which will be active and whose functionality will be available during the trial. Any customisation or tailoring of active components that had to be performed for the trail is also stated.

WeCare Component	Used within trial	Customisation/Tailoring performed on component for trail
Wecare Portal	No	A tailored and customized WeCare system was used in the Finnish trial.
Family Portal	Yes	
Videoconferencing view	Yes	
Videra Touch Screen Video	Yes	
Broadcasting station and the group home units	Yes	See customization Chapter
WeCare environment	Yes	See customization Chapter
Video Conferencing Administration	Yes	See customization Chapter
Family Portal Administration	Yes	See customization Chapter

Section 8.2.2: Spain

The second WeCare system end user trail will take place within Spain. The following table shows the WeCare system components which will be active and whose functionality will be available during the trial. Any customisation or tailoring of active components that had to be performed for the trail is also stated.

WeCare Component	Used within trial	Customisation/Tailoring performed on component for trail
Med2Mobile	Yes	See customization chapter for adaption's required on medicine reminder
Portal page	Yes	The portal page displays summary contents from Spanish components in addition to allowing direct access to these components. The page also

		highlights the groups that the current user is a member of.
Skytek forum	No	The forum was not required by the end users of the Spanish trial. Instead information was shared through the notice board features provided by Sharecare.
Logon for open and closed groups	Yes	Language, email, character set, more browsers, GUI, transparencies, API, protection, separate back-ups, interfaces, minor programming adaptations, connections with pilot group module
Open group module	Yes	Language, email, character set, more browsers, GUI, transparencies, API, protection, separate back-ups, interfaces, minor programming adaptations, connectability with pilot group module
Closed group module	No	
Pilot group module	Yes	Language, email, character set, more browsers, GUI, transparencies, API, protection, separate back-ups, interfaces, minor programming adaptations
Calendar	Yes	Language, email, character set, more browsers, GUI, transparencies, API, protection, separate back-ups, interfaces, minor programming adaptations, connectability with pilot group module
Forum, discussion	Yes	Language, email, character set, more browsers, GUI, transparencies, API, protection, separate back-ups, interfaces, minor programming adaptations, connectability with pilot group module
Member's list	Yes	Language, email, character set, more browsers, GUI, transparencies, API, protection, separate back-ups, interfaces, minor programming adaptations, connectability with pilot group module
Profile page	Yes	Language, email, character set, more browsers, GUI, transparencies, API, protection, separate back-ups, interfaces, minor programming adaptations, connectability with pilot

		group module
Neighbours	Yes	Language, email, character set, more browsers, GUI, transparencies, API, protection, separate back-ups, interfaces, minor programming adaptations, connectability with pilot group module

Section 8.2.3: Netherlands

The following table shows the WeCare system components which will be active and whose functionality will be available during the Dutch trial. Any customisation or tailoring of active components that had to be performed for the trail is also stated.

WeCare Component	Used within trial	Customisation/Tailoring performed on component for trail
Portal	Yes	The pick and mix portal has been customised for the Dutch trail. Navigation buttons are moved to the top and per group a preview of each component is given on the portal page.
Consolidated contacts list	Yes	See customizations chapter for complete description
Web Calendar	Yes	Small customizations were implemented in order to make all functionality available for the iPad
Logon for open and closed groups	Yes	Made iPad compatible, changed for the Dutch trial
Open group module	No	
Closed group module	Yes	Special version for Dutch trial, different interface, made iPad compatible
Pilot group module	Yes	Needed and build to comply with both WeCare and different trials and the customised components
Calendar	Yes	Graphics, minor details
Forum, discussion	Yes	Small customizations were implemented in order to make all functionality available for the IPad
Member's list	Yes	Heavily adapted for the Dutch trail
Profile page	Yes	Minor graphics
Neighbours	Yes	Minor changes

Section 8.2.4: Ireland

The following table shows the WeCare system components which will be active and whose functionality will be available during the Irish trial. Any customisation or tailoring of active components that had to be performed for the trail is also stated.

WeCare Component	Used within trial	Customisation/Tailoring performed on component for trail
Navigator	Yes	Tailored portal page to a look and feel for the Irish trial which allowed easy use on mobile and tablet devices.
Phone/Video	Yes	Configured in advance of trial with individual users friends and contacts details.
Weather	Yes	Configured for Irish information weather services.
AgFood	Yes	N/A
Sport	Yes	Configured for Irish sports news source
News	Yes	Configured for Irish news source
Med2Mobile	Yes	Configured with details of Irish mobile phone numbers of trial users.
User administration	Yes	N/A
Logon for open and closed groups	No	
Open group module	No	
Closed group module	No	
Pilot group module	No	
Calendar open groups	No	
Calendar closed groups	No	
Forum, discussion	No	
Member's list	No	
Profile page	No	
Neighbours	No	

Section 9: System Evolution (pre commercialization)

This chapter was added in order to capture our thoughts on what needs to be done if we take the Wecare service commercial. Obviously there is a huge difference between a trial setup and a commercial one. Once users start paying, the user will expect more from it.

This chapter will address User & Group management aspects, Security & Content issues and in conclusion a recommendation on billing is given and a cost estimation for a commercial running WeCare.

Section 9.1: User Management

Recommended for trial:

The WeCare system involved the integration of several different solution provided by different technical partners within the WeCare project. Each of these systems used different architectures, platforms and technologies. Since the WeCare system wanted to provide a single sign on access to all components for the end users involved in the trials the different solutions required exchanging user information between the central portal component and each component. For the trial infrastructure instead of consuming a large number of person months effort in the creation of a centralized user management system it as agreed that the focus should be on the development of features and functionality rather than the implementation of an advanced centralized user management system. Therefore the trial system provides a centralized portal for end user and exchanges information between components using parameters encoded within URL messages exchanged with integrated components.

This approach allows for the full testing of a range of functionality through a single sign on interface for the user. However the disadvantage was that each technical partner of WeCare had to independently manage the user accounts and update of user information. This information was stored and distributed through a shared password protected Excel file that was available on the WeCare Sharepoint server during the trial execution.

Recommended for commercial product:

The approach for user management worked well within WeCare for the trial period due to the fixed and limited number of users. However for the full commercial product it would be impractical for each technical member of WeCare to independently manage the user account information. Therefore a fully centralized user management system which automatically feeds details to each of the components independent of location and developer would be required. This system would use web services technology for the communication of updates and new user account details to all components from a single centralized source. In addition this system could integrate with emerging global centralized user account systems such as Google and/or Facebook using emerging standards such as OAuth. This would allow users use their username and passwords from services of Google and Facebook for logging into the commercial WeCare solution.

Section 9.2: Group Management:

Recommended for trail:

Based on the analysis of end user requirements two types of groups were requested. Firstly, open groups which are defined by administrators of the system and which users can be members of. For example these groups could be areas within a larger trial such as a town or village which the end user is a member of. Within the Spanish trial for example three such sub groups exist. Secondly, closed groups which are groups which individual users can create and invite other members of WeCare to be a member of. For example this could be a special interest group that the user has set up. The concept of closed groups was successfully integrated and deployed for the Dutch trial of WeCare.

Recommended for commercial product:

The concept of both open and closed groups proved extremely popular to the end users. For the commercial WeCare product the concept of 'closed' groups should be rolled out to all regions and not just Dutch users.

Section 9.3: Security

Recommended for trail:

Single sign on was required for access to the WeCare trial components, each component checked on access if the user credentials were correct and access was denied if the tests were not passed. URL encoding of information between components was used. The standard HTTP protocol was used for end users to access the WeCare trial servers.

Recommended for commercial product:

Within the WeCare commercial product information exchanged between component in particular the exchange of user details will be fully encrypted to avoid attacks through interception and decoding of information exchanges between WeCare servers located in remote locations. Additionally the commercial product will only provide HTTPS access to the system so that a secure and encrypted end to end link is provided between the WeCare servers and the end users browser.

Section 9.4: Content Management possibility

Recommended for trail

The trial system provided a number of components for content management through individual components. User could enter or consume content from the integrated components of WeCare. For several of these components there was the additional functionality that summary details of content within a component could be displayed on the end users home portal page.

Recommended for commercial product:

For the commercial version of WeCare the concept for content management should be expanded further. This can be done in two ways.

- Extension of each integrated WeCare component so that summary content information can be extracted on a per user basis and presented on the home portal page. This would allow the end user get a quick overview of added content without having to navigate into each of the individual content components.
- Sharing of content between components. Currently each component acts like an information silo where information entered is stored and managed by that component. This concept should be expanded so that information can be shared or viewed within several components and not just the component in which it was entered.

Section 9.5: Billing, Licensing and provisioning.

For the Wecare edition as used in and developed for the trials, no components were built regarding billing. Nor were additions or arrangements made for e.g. the switching on and off of modules depending on licences. For this –trail- edition it was not needed.

All users in all participating countries were allowed to use the Wecare trial for free.

Because of the scale of use during the trial all modules and country-editions were monitored either manually, partly in an indirect way by the OWA system. In this way the use could be checked and judged to avoid misuse low use or overload. None of these occurred.

This method of working worked out fine for the trials.

For a commercial version of Wecare, of course different ways of billing and licensing can be developed. Some can be straightforward, others can be built in such a way that the functionality depends on the purchased modules, or even to an extend that the level of functionality is influenced by the height of the pricing.

In short the following extensions and upgrades are possible:

- a license for a group of users purchased by a secondary or tertiary user: no new technical provisions needed.
- a simple license (possibly free of charge) for a group of primary users, but billing for additional services like video, market-activities, quantity of use: the billing can be done via a newly built module monitoring actions and use by the users that must be charged and that delivers input to a financial administrative billing system.
- Alternatively the use and quantity of use and modules (or functionalities) can be billed directly by plug-ins that link to paying systems like:
    - Credit card
    - Debit card, like iDeal (so far just for the Dutch market)
    - Automatic bank withdrawal
    - Paying by (smart-)phone charge
- the use and license as a whole, so, for the entire service, of course can be billed in the way just described for all primary users, if chosen so.
- a lot of combinations and varieties of paying and billing related to the use can be made. In practice this will depend on the choices locally made by the stakeholders. The quality of billing (safeguarding, security, complexity, taxes etc) is a matter of choice too.

In Work Package 4 (Business Modelling) more details are given, and is also referred too for the definitions of primary, secondary etc. users.

Section 9.6: Estimated HW/SW costs

WP2 provided input to WP4 (business modeling work package) but this information is relevant in this chapter as well. Please also see D4.1 Chapter 5.5.1 for more information.

A typical Wecare 2.0 deployment will consist of a set of services that are made available over the internet (through a web portal). Depending on the type of services additional end user equipment might be required. Basic and additional user equipment and internet access are not taken into account as cost, but are set as prerequisites (typical costs are included.) Also total costs are listed for a typical deployment of a Wecare service set (5 services). An important note is that some of the cost are fixed and do not possess a linear relation with the number of users.

For a start an amount of 2000 users is chosen. Some of the components (like the video service) start form a bottom limit of 4000. Some of the subservices add per 1000 users (given by the systems of the partner). The effect of a 24/7 professional support is substantial. The table shows a total of about € 200.000 for three years with a support during office hours, and a total of € 650.000 if the choice for a 24/7 service would be made. These total figures should be rounded off slightly upwards to compensate for the rounding off downwards for several isolated low cost parts per user (see table).

Hardware/ Software	Description	Unit Cost (Euro)	units	number of users	Yearly per enduser (Euro)
Video Communication *client access*	e.g. the Videra service, client access is "cost" free	pm	1		0
Video Communication *server*	Server part	5K/y		4000	1,25
Med2Mobile *client access*	e.g. the Ericsson medicine reminder, client access is "cost" free	pm	1		0
Med2Mobile *server*	Server part	4K/y		2000	2
Consolidated Contact List *client access*	e.g the Ericsson Consolidated Contact list	20/month	1		0
Consolidated Contact List *server*	Server part	4K/y		2000	2
Collaboration Component *client access*		0	1		0
Collaboration Component	e.g one of the skytek collaboration services (forum / calendar / document sharing)	4K/y	1(assuming 1000 users)	1000	4
Community connection *client access*	e.g. Simac/Sharecare connection	0	1		0

Community connection *server*	Server part	8K/y		2000	4
Typical cost server	Average Server costs (3year replacement period)	10K/3y		2000	1,65
TOTAL support 24/7	24 hours 7 days a week support requires 2FTE.	200K/y		2000	100
TOTAL support 5*8	Office hours support requires 0.5FTE	50K/y		2000	25
TOTAL client access		0	1		0
TOTAL server	A total Wecare2.0 setup consisting of 5 typical services	50K/3y	1	2000	8,35
Total per user	If 24/7 support				108,35
Total per user	If support during office hours				33,35

Section 10: Annex

Section 10.1: Generated Javadoc incl. Class Diagrams

The actual class structure, class documentation and class context information based on class diagrams is available as generated Javadoc. This information is provided as ZIP-archive containing a number of interlinked HTML files. Different views provide the package structure, the complete list of classes and finally a single documentation file for every public class.

Section 10.2: Applied checkstyle rules

The following XML section is a configuration file for the coding style reporting tool checkstyle. The configuration file lists all rules that are supported by the tool. The actual configuration is done by commenting out rules that are not to be checked and setting specific parameters for rules that need to be specified more in detail.

```
<?xml version="1.0"?>
<!DOCTYPE module PUBLIC
    "-//Puppy Crawl//DTD Check Configuration 1.1//EN"
    "http://www.puppycrawl.com/dtds/configuration_1_1.dtd">

<!-- $Id: //mmweb/main/etc/checkstyle-config-master.xml#7 $ -->
<!—   Paul Kiernan (Paul.Kiernan@skytek.com) -->

<!-- Checkstyle configuration. NOTE: There is a single master configuration
     file called checkstyle-config-master.xml. Only edit this file and
     run make to generate the derived files!
```

```
     The master file partitions checkstyle rules in three sets: ERROR,
     WARN, and IGNORE. Each rule in the master file is enclosed in a
     comment which is prefixed with one of these names. A simple regex
     replace on the master file generates the files
     checkstyle-config-error.xml containing all error rules and
     checkstyle-config-warn.xml containing all warning rules.
     Violations of error rules let the build fail. Violations of
     warning rules are documented in an HTML report on the build
     server.

     Additionally, there are tags WILL-BE-ERROR that generate a file
     checkstyle-config-will-be-error.xml. These checks generate error
     messages, but do not let the build fail. This allows for a
     transition phase where problems can be fixed. When all problems
     are fixed, the rule will be promoted to a hard error (with build
     failures).
-->

<module name="Checker">

  <!-- Base directory name; stripped off in messages about files -->
  <property name="basedir" value="${basedir}" />

  <!-- SuppressionFilter rejects audit events for Check errors
       according to a suppressions XML document in a file. -->
  <module name="SuppressionFilter">
    <property name="file" value="${etc.dir}/checkstyle-suppressions.xml"/>
  </module>

  <!-- Checks that a package.html file exists for each package. -->
    <module name="PackageHtml" />

  <!-- IGNORE:  <module name="StrictDuplicateCode"> -->
  <!-- IGNORE:    <property name="min" value="20"/> -->
  <!-- IGNORE:  </module> -->

    <module name="NewlineAtEndOfFile" />

  <!-- A FileSetCheck that ensures the correct translation of code by
       checking property files for consistency regarding their keys. -->
  <module name="Translation"/>

  <module name="TreeWalker">

    <!-- Caches information about files that have checked ok; used to
         avoid repeated checks of the same files -->
    <property name="cacheFile"  value="${build.dir}/checkstyle-cachefile"/>

    <!--
    ================================================================
    Checks for Javadoc Comments.
    ================================================================
    -->

    <!-- Checks Javadoc comments for class and interface definitions. -->
        <module name="JavadocType">
```

```
          <property name="scope" value="anoninner" />
          <property name="authorFormat" value='&lt;a
href="mailto:.*@.*&gt;.*&lt;/a&gt;' />
        </module>

    <!-- Checks Javadoc comments for methods. -->
        <module name="JavadocMethod">
          <property name="scope" value="protected"/>
          <property name="allowMissingParamTags" value="true"/>
          <property name="allowMissingReturnTag" value="true"/>
          <property name="allowUndeclaredRTE" value="true"/>
        </module>

    <!-- Checks that variables have Javadoc comments. -->
        <module name="JavadocVariable">
          <property name="scope" value="public"/>
        </module>

    <!-- Validates Javadoc comments to help ensure they are well formed. -->
        <module name="JavadocStyle">
          <property name="checkEmptyJavadoc" value="true" />
        </module>

    <!--
    ================================================================
    Checks for Naming Conventions.
    ================================================================
    -->

    <!-- WARN: <module name="AbstractClassName"/>  -->

        <module name="ConstantName"/>
        <module name="LocalFinalVariableName" />
        <module name="LocalVariableName"/>
        <module name="MemberName">
          <property name="format" value="^f[A-Z][a-zA-Z0-9]*$"/>
        </module>
        <module name="MethodName" />

    <!-- disallow underscores and uppercase letters in package names -->
        <module name="PackageName">
          <property name="format" value="^[a-z]+(\.[a-z][a-z0-9]*)*$" />
        </module>
        <module name="ParameterName"/>
        <module name="StaticVariableName" />
        <module name="TypeName" />

    <!--
    ================================================================
    Checks for Headers.
    ================================================================
    -->

    <!-- Checks the header of a source file against a header file that
    contains a regular expression for each line of the source
    header. -->
        <module name="RegexpHeader">
```

```
          <property name="headerFile" value="${etc.dir}/header.java" />
        </module>

    <!--
    ============================================================================
    Checks for Imports.
    ============================================================================
    -->

    <!-- Checks that there are no import statements that use the * notation.
-->
        <module name="AvoidStarImport" />

    <!-- Checks for imports from a set of illegal packages. -->
        <module name="IllegalImport" />

    <!-- Checks for redundant import statements. -->
        <module name="RedundantImport" />

    <!-- Checks for unused import statements. -->
        <module name="UnusedImports"/>

    <!-- Checks the ordering/grouping of imports. -->
        <module name="ImportOrder">
          <property name="ordered" value="false" />
          <property name="separated" value="true" />
          <property name="caseSensitive" value="false" />
        </module>

    <!--
    ============================================================================
    Checks for Size Violations.
    ============================================================================
    -->

    <!-- Restricts the number of executable statements to a specified limit.
-->
    <!-- TODO: reduce value -->
        <module name="ExecutableStatementCount">
          <property name="max" value="81"/>
        </module>

    <!-- Checks for long source files. -->
        <module name="FileLength" />

    <!-- Checks for long lines. -->
        <module name="LineLength">
          <property name="max" value="120" />
      <!-- ignore lines that begin with " * ", followed by just one
           word, such as within a Javadoc comment: -->
          <property name="ignorePattern" value="(^import +)|(^ *\* *[^
]+$)|(^ *\* *@see)|(^ *\* *\{@link)|(^ *\* *@author)|(^ *\* *\$Id)"/>
        </module>

    <!-- Checks for long methods and constructors. -->
        <module name="MethodLength"/>
```

```
    <!-- Checks for long anonymous inner classes. -->
        <module name="AnonInnerLength">
          <property name="max" value="40"/>
        </module>

    <!-- Checks the number of parameters of a method or constructor. -->
        <module name="ParameterNumber">
          <property name="max" value="8"/>
        </module>

    <!--
    ================================================================
    Checks for whitespace.
    ================================================================
    -->

        <module name="EmptyForInitializerPad"/>
        <module name="EmptyForIteratorPad"/>
        <module name="MethodParamPad"/>
    <!-- WARN:     <module name="NoWhitespaceAfter"/> -->
        <module name="NoWhitespaceBefore"/>
        <module name="OperatorWrap"/>
        <module name="ParenPad"/>
        <module name="TabCharacter"/>
        <module name="TypecastParenPad"/>
        <module name="WhitespaceAfter"/>
        <module name="WhitespaceAround">
          <property name="tokens" value="ASSIGN, BAND, BAND_ASSIGN, BOR,
BOR_ASSIGN, BSR, BSR_ASSIGN, BXOR, BXOR_ASSIGN, COLON, DIV, DIV_ASSIGN,
EQUAL, GE, GT, LAND, LCURLY, LE, LITERAL_ASSERT, LITERAL_CATCH, LITERAL_DO,
LITERAL_ELSE, LITERAL_FINALLY, LITERAL_FOR, LITERAL_IF, LITERAL_RETURN,
LITERAL_SYNCHRONIZED, LITERAL_TRY, LITERAL_WHILE, LOR, LT, MINUS,
MINUS_ASSIGN, MOD, MOD_ASSIGN, NOT_EQUAL, PLUS, PLUS_ASSIGN, QUESTION,
RCURLY, SL, SLIST, SL_ASSIGN, SR, SR_ASSIGN, STAR, STAR_ASSIGN,
TYPE_EXTENSION_AND, WILDCARD_TYPE" />
        </module>

    <!--
    ================================================================
    Modifier Checks.
    ================================================================
    -->

        <module name="ModifierOrder"/>
    <!-- IGNORE:   <module name="RedundantModifier"/> -->

    <!--
    ================================================================
    Checks for blocks.
    ================================================================
    -->

    <!-- Finds nested blocks, i.e. blocks that are used freely in the code.
-->
        <module name="AvoidNestedBlocks"/>

    <!-- Checks for empty blocks. -->
```

```
        <module name="EmptyBlock">
          <property name="option" value="text" />
        </module>
        <module name="LeftCurly"/>

        <module name="NeedBraces"/>
        <module name="RightCurly"/>

    <!--
    ============================================================================
    Checks for common coding problems.
    ============================================================================
    -->

    <!-- IGNORE:   <module name="ArrayTrailingComma" /> -->
    <!-- WARN:     <module name="AvoidInlineConditionals" /> -->
        <module name="CovariantEquals" />
        <module name="DeclarationOrder" />
        <module name="DefaultComesLast"/>
        <module name="DoubleCheckedLocking" />
        <module name="EmptyStatement" />
        <module name="EqualsHashCode" />
        <module name="ExplicitInitialization" />
        <module name="FallThrough"/>
    <!-- IGNORE:   <module name="FinalLocalVariable" />  -->
        <module name="HiddenField" />
        <module name="IllegalCatch" />
        <module name="IllegalInstantiation">
          <property name="classes"
value="java.lang.Boolean,java.lang.String"/>
        </module>
        <module name="IllegalThrows" />
    <!-- IGNORE:   <module name="IllegalToken" />  -->
    <!-- IGNORE:   <module name="IllegalTokenText" />  -->
    <!-- IGNORE:   <module name="IllegalType" />  -->
        <module name="InnerAssignment" />
        <module name="JUnitTestCase" />
    <!-- IGNORE:   <module name="MagicNumber" />  -->
    <!-- IGNORE:   <module name="MissingCtor"/> -->
        <module name="MissingSwitchDefault" />
        <module name="ModifiedControlVariable"/>
        <module name="MultipleVariableDeclarations"/>
        <module name="NestedIfDepth" >
          <property name="max" value="4"/>
        </module>
        <module name="NestedTryDepth" />
        <module name="PackageDeclaration" />
        <module name="ParameterAssignment" />
    <!-- IGNORE:   <module name="RedundantThrows" /> -->
        <module name="ReturnCount" />
    <!-- IGNORE:   <module name="RequireThis"/> -->
        <module name="SimplifyBooleanExpression" />
        <module name="SimplifyBooleanReturn" />
        <module name="StringLiteralEquality" />
        <module name="SuperClone" />
        <module name="SuperFinalize" />
    <!-- WARN:     <module name="UnnecessaryParentheses"/> -->
```

```
    <!--
    ============================================================================
    Class Design Checks.
    ============================================================================
    -->

    <!-- Checks that classes are designed for extension. More
    specifically, it enforces a programming style where superclasses
    provide empty "hooks" that can be implemented by subclasses. -->
    <!-- WARN:     <module name="DesignForExtension" /> -->

    <!-- Checks that a class which has only private constructors is declared
as final. -->
        <module name="FinalClass" />

    <!-- Make sure that utility classes (classes that contain only
    static methods) do not have a public constructor. -->
        <module name="HideUtilityClassConstructor" />

    <!-- IGNORE:   <module name="InterfaceIsType" /> -->

    <!-- Ensures that exceptions are immutable. That is, have only final
fields. -->
        <module name="MutableException" />

    <!-- Restricts throws statements to a specified count. -->
        <module name="ThrowsCount">
          <property name="max" value="2"/>
        </module>

    <!-- Checks visibility of class members. Only static final
    members may be public. -->
        <module name="VisibilityModifier">
          <property name="protectedAllowed" value="true"/>
        </module>

    <!--
    ============================================================================
    Metrics Checks.
    ============================================================================
    -->

    <!-- WARN:     <module name="CyclomaticComplexity"/> -->
        <module name="CyclomaticComplexity">
           <property name="max" value="30"/>
        </module>
        <module name="BooleanExpressionComplexity"/>
    <!-- WARN:     <module name="ClassDataAbstractionCoupling"/> -->
    <!-- WARN:     <module name="ClassFanOutComplexity"/> -->
        <module name="NPathComplexity">
           <property name="max" value="300"/>
        </module>
    <!-- WARN:     <module name="NPathComplexity"> -->
    <!-- WARN:       <property name="max" value="100"/> -->
    <!-- WARN:     </module> -->
    <!-- WARN:     <module name="JavaNCSS"/> -->
```

```
    <!--
    ============================================================================
    Miscellaneous Checks.
    ============================================================================
    -->

       <module name="ArrayTypeStyle"/>

    <!-- IGNORE:  <module name="DescendantToken" /> -->
    <!-- IGNORE:  <module name="FinalParameters"/> -->

    <!-- WARN:    <module name="GenericIllegalRegexp"> -->
    <!-- WARN:      <property name="format" value="System\.out\.println"/> -
->
    <!-- WARN:      <property name="message" value="Consider using logging
instead of System.out.println"/> -->
    <!-- WARN:    </module> -->

    <!-- WARN:    <module name="GenericIllegalRegexp"> -->
    <!-- WARN:      <property name="format" value="System\.err\.println"/> -
->
    <!-- WARN:      <property name="message" value="Consider using logging
instead of System.err.println"/> -->
    <!-- WARN:    </module> -->

       <module name="Indentation"/>

    <!-- FIXMEs and DOCMEs should be fixed before submitting code -->
       <module name="TodoComment">
         <property name="format"
value="([fF][iI][xX][mM][eE])|([dD][oO][cC][mM][eE])"/>
       </module>

    <!-- TODOs may be fixed later -->
    <!-- WARN:    <module name="TodoComment"> -->
    <!-- WARN:      <property name="format" value="[tT][oO][dD][oO]"/> -->
    <!-- WARN:      <property name="severity" value="warning" /> -->
    <!-- WARN:    </module> -->

    <!-- Checks for uncommented main() methods (debugging
    leftovers). Test code should be in unit tests. -->
    <!-- TODO:ERROR:    <module name="UncommentedMain"> -->
    <!-- TODO:ERROR:      <property name="excludedClasses" value="\.Main$"/>
-->
    <!-- TODO:ERROR:    </module> -->

    <!-- IGNORE:   <module name="TrailingComment"/> -->

    <!-- Checks that long constants are defined with an upper ell. -->
        <module name="UpperEll"/>

    <!--
    ============================================================================
    Unused code.
    ============================================================================
    -->
```

```
    <!-- These are optional rules: drawback is that they currently
         need a lot of processing time -->
    <!-- IGNORE:    <module name="usage.UnusedLocalVariable"/> -->
    <!-- IGNORE:    <module name="usage.OneMethodPrivateField"/> -->
    <!-- IGNORE:    <module name="usage.UnusedPrivateField"/> -->
    <!-- IGNORE:    <module name="usage.UnusedPrivateMethod"/> -->
  </module>
</module>
```