|  |  |
| --- | --- |
|  | **2012** |
|  | Lyubomir Yanchev Hristo Stoyanov |

|  |
| --- |
| **[Documentation for “ModernSteward“]** |
|  |

Contents



[Goals 3](#_Toc320910030)

[Plugins 4](#_Toc320910031)

[Gmail plugin 4](#_Toc320910032)

[PowerPoint plugin 4](#_Toc320910033)

[Searcher plugin 4](#_Toc320910034)

[ElectricityManager plugin 4](#_Toc320910035)

[Stages of realization 4](#_Toc320910036)

[Level of complexity of the project 5](#_Toc320910037)

[Logical and functional description of the solution 5](#_Toc320910038)

[Architecture 5](#_Toc320910039)

[Core 5](#_Toc320910040)

[Graphical User Interface (GUI) 6](#_Toc320910041)

[PluginWizard 6](#_Toc320910042)

[Plugins 6](#_Toc320910043)

[Realization 6](#_Toc320910044)

[Platform and User Interface 6](#_Toc320910045)

[Source control 6](#_Toc320910046)

[Speech recognition 7](#_Toc320910047)

[Application description 7](#_Toc320910048)

[Plugins 8](#_Toc320910049)

[Plugins development 8](#_Toc320910050)

[Grammar 8](#_Toc320910051)

[Interface 9](#_Toc320910052)

[Conclusion 12](#_Toc320910053)

# Goals

ModernSteward is an application, which enables the user to command verbally its computer and also other electronic devices connected to it.

The product is developed by two students – Lyubomir Yanchev and Hristo Stoyanov, who have been working together as a team since 2009. We chose this topic in our desire to develop the IT industry and make our life easier.

After a deep research we didn’t come across a finished product which completes the given task.

ModernSteward is the first product giving this kind of functionality on personal computers that we see in modern operation system (by February 2012) like Siri in iOS.

We found an interesting material which shows the usage of verbal commands to cooperate with GIMP under Linux. Unfortunately, we couldn’t connect to the author to compare ModernSteward to the application used in the video.

Our aim was to create an application which can be used by everybody without need of special training or programming skills. Our goal was to create a flexible software product which facilitates the user as much as possible.

We made a plug-in-based environment, which enables the user to easily add plug-ins, which are executing specific commands. The developers of ModernSteward plug-ins are limited only by the hardware and their imagination – from internet search engine to, for example, a plug-in that gets your current GPS location from your phone and gives you live information about traffic and public transport.

For facilitation we lead in four main terms:

* Main program - The program that is started
* Plug-in – add-in, which gives specific functionality (for example: turning on a specific device)
* PluginWizard – the wizard, which guides the programmer through the process of developing a new plug-in
* Grammar – A tree representing possible commands which the user can use to trigger a specific function of the plug-in

## Plugins

* The main functionality of ModernSteward is built via these plug-ins. More information about how to make a plug-in can be found in the “Realization” section.

### Gmail plugin

* + Integration with one of the most popular email messengers. Checks the users Gmail account for new incoming mail and shows a window with the unread emails.

### PowerPoint plugin

* + This plug-in controls PowerPoint presentation. It can open a presentation, navigate through the slides and close the presentation. There is one dependency - PowerPoint has to be from Microsoft Office 2007.

### Searcher plugin

* + This plug-in search through some of the most popular search engines, for now – Google, Wikipedia and IMDb.

### ElectricityManager plugin

* + Via a special device, this plug-in controls two standard electricity contacts. It’s used to turn on and off electrical devices like TV, lights, coffeemaker, etc.

# Stages of realization

There are four main stages of realization.

The first one is to make the idea clearer and to pick which functionalities to implement and which to postpone.

The second stage is to think out the architecture and take into consideration how to make it in a way, which satisfies the laid requirements of the functionalities. We needed an architecture which sets as lower limits as possible for the plug-in programmers.

The product realization continued with implementing the whole idea and the first plug-ins.

The last stage is to fix the found bugs, to search for new ones and to test the application under real conditions in everyday life.

For most of the time we have been working as a team – thinking out the architecture, the plug-ins, the choice of programming language, the speech recognition engine library, etc. Then we ended up with this decision:

* Lyubomir Yanchev implements the main program;
* Hristo Stoyanov creates the plug-ins and tests the application for bugs.

# Level of complexity of the project

The main problem we faced was the choice of speech recognition engine library that satisfies our requirements. Our decision is explained in details in section “Realization: Speech recognition”.

Other major problem was how to control the special devices connected to the computer. For the demonstration we made a device which controls two SCHUKO switches. This device have prove that ModernSteward can be used in various ways – including controlling home stand-alone devices like TV, lights, cookers, washers, etc.

# Logical and functional description of the solution

## Architecture

The product consists of four main modules:

* Core
* Graphical User Interface (GUI)
* PluginWizard
* Utility
  + Module which contains general variables, classes and etc.

### Core

The Core loads the plug-ins loaded by the user. It contains the speech recognition engine and the “translator” of the plug-in’s grammar, which loads the grammar into the speech recognition engine.

### Graphical User Interface (GUI)

With it the user loads and initializes the plug-ins. After all desired plug-ins are loaded and initialized the user can start the recognition engine.

### PluginWizard

Tool that guides the programmer through the steps of making a new plugin.

### Plugins

The plug-ins gives real functionality of the product. They are created with help of the PluginWizard and are loaded into the application through the GUI. They are stand-alone .NET libraries which are compressed along with their dependencies in .zip archives.

# Realization

## Platform and User Interface

During the development of the program were used:

* NET 4.0 Framework
* C# programming language along with Visual Studio 2010 Expess Edition



Telerik kindly placed their product RadControls for WinForms at our disposal.

## Source control

Effective team work requires SCM – source control management software. We chose Mercurial – a decentralized system, meaning that every clone poses full history of the code.



We needed professional quality thus we used Kiln On Demand. This product allows free usage by students restricting the users up to two.

In addition we used a bug tracking software called FogBugz.

## Speech recognition

In the table below is a comparison between different projects among we had choice for a recognition library. Dragon NaturallySpeaking is paid and this rendered this option nonviable. To conform to the concept for easy development of plugins we chose the built-in Windows Speech Recognition.

It makes us able to provide everyone with the possibility to create their own functionality. More can be found in article “Plugins”.

We made use of Windows PowerShell. It is included in the system instruments of Windows 7. We added the module for integration with Mercurial (posh-hg) aiming easy communication with the server.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Name** | **Company** | **Code** | **$** |
| 1 | Windows Speech Recognition | Microsoft | closed-source | 0$ |
| 2 | Dragon NaturallySpeaking | Nuance | closed-source | 199.99$ |
| 3 | CMU Sphinx | Carnegie Mellon University | open-source | 0$ |

# Application description

The application is installed when starting the file setup.exe found in the directory ModernSteward\Installation\ on the CD.

After the installation, the product can be started from the icon created on the users’ desktop.

The user interface is intuitive and leads the user step by step through the work with the application.

Below , there is an explanation of the main nomenclature used by the authors while working on “Modern Steward”.

## Plugins

Plugins are independent .NET libraries that are loaded and saved in the user’s profile. Every developer describes the grammar and the commands the plugin will handle. The grammar is send to the core where the speech is recognized.

### Plugins development

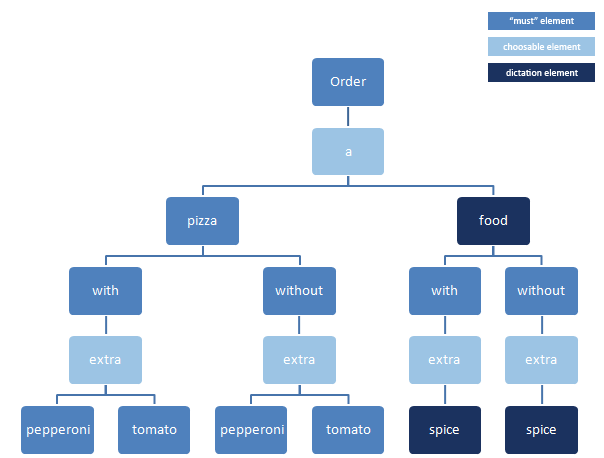
To facilitate the development of plugins we created the PluginWizard. It helps the user to create his own plugin in a few easy steps. After that a C# project will be created at the specified place. It is independent so you can include any library developed for .NET.

## Grammar

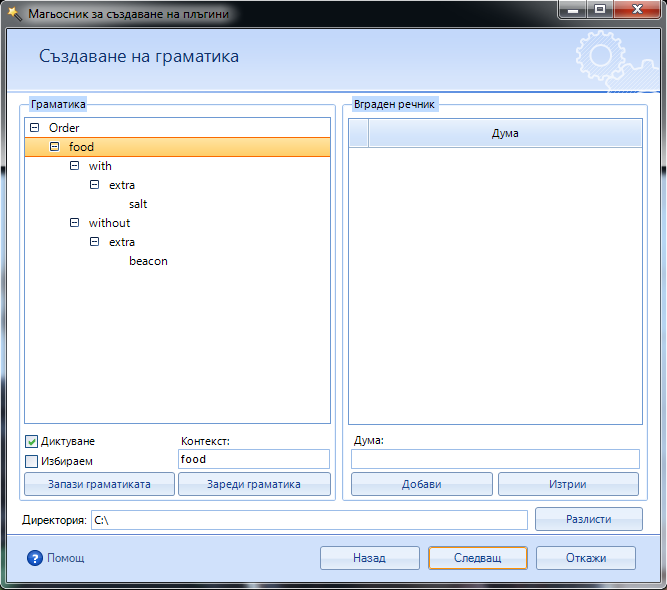
The grammar is a description of the commands the user is allowed to use. Every plugin can build its own as there is no limitation on the number of the words or the design.

Commands are built as a sequence of words that is represented as a tree. Every phrase has subtrees that can be said after it.

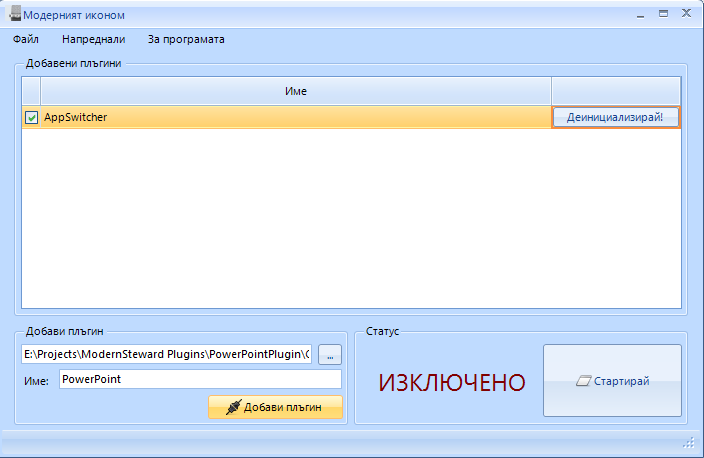
The maintainer can put commands on every node of that tree. This way he is free to build flexible and effective plugins. Below, there is a scheme with an example for a grammar.



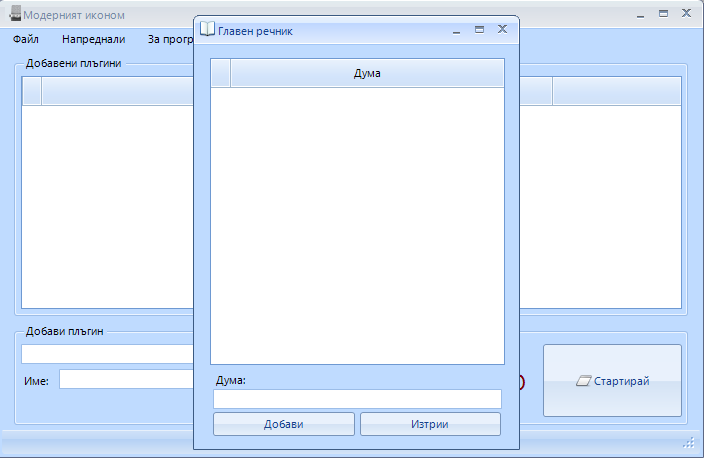
## Interface



The PluginWizard with some random grammar.



Example for added and initialized plugin and example how to add a new one



The MasterDictionary where can be added words to the master dictionary of the computer. It’s used for better recognition.

# Conclusion

The application “Modern Steward” has a well-rounded appearance and according the developers it completes the goals set in the beginning. It can be installed on every device with Windows that has Windows Speech Recognition and .NET 4.0. The usage is limited only by the resources and the imagination of the developers. From urban transport, through banking service, to military the program can be used everywhere with proper refinement of the speech recognition.

The program was successfully demonstrated to the student in Private Language High-school "Erich Käsnter" and has been used in number of private households.

The development of the project does not stop here. The authors are confident and enthusiastic about its future. We would like to make additional functionality with more plugins. We have planned to build a market from where with only a few clicks they will be downloaded and installed and “just works”.