The GPII on Desktops in PCs OSs: Windows and GNOME

Javier Hernández Antúnez[1], Colin Clark[2], and Kasper Galshiot Markus[3]

[1] Emergya, Sevilla, Spain
[2] Inclusive Design Resource Center, OCAD University, Toronto, Canada
[3] Raising the Floor - International, Geneva, Switzerland
jhernandez@emergya.com

Abstract. Since The Global Public Inclusive Infrastructure [1] aims to become an international standard, one of the biggest challenge of the GPII project is to support the many Operating Systems that are running in our Personal Computers. Nowadays, we make an extensive use of personal computers—both laptops and desktops—and their Operating Systems in a lot of circumstances, and as a result of this, we can say that a lot of people could have difficulties when they start using a new computer. Add to this the diversity of different software applications that people use and their many different restrictions of use, and the problem becomes bigger when a person needs some Assistive Technology [2] to use a computer efficiently .As part of the Cloud4all project, the implementation of the GPII on PC OSs is taking place to address these problems. By having a first version of this implementation, the GPII is ready to help the users to solve their problems when using a new PC's OS for the first time.

Keywords: Accessibility, Internet Access, Health, Social inclusion, Cloud.

1 Motivation

1.1 Background

To demonstration how we use personal computers and different operating systems in different contexts, we can describe some real-life scenarios:

- We use our own PCs, either for work or for other personal affairs
- We use others' or shared PCs at work, at home, or at any public place
- We use public "kiosks" [3] that run standard PC OSs such as Windows or Linux for a variety of different purposes

Also, most people with disabilities who need Assistive Technology (AT) to bridge the gap between them and the digital era are using the most popular OSs. These typically include Microsoft Windows, Apple's Mac OS X, or the open-sourced operating system GNU/Linux. Users generally employ a variety of ATs running on their PC OSs.

C. Stephanidis and M. Antona (Eds.): UAHCI/HCII 2014, Part IV, LNCS 8516, pp. 390–400, 2014.

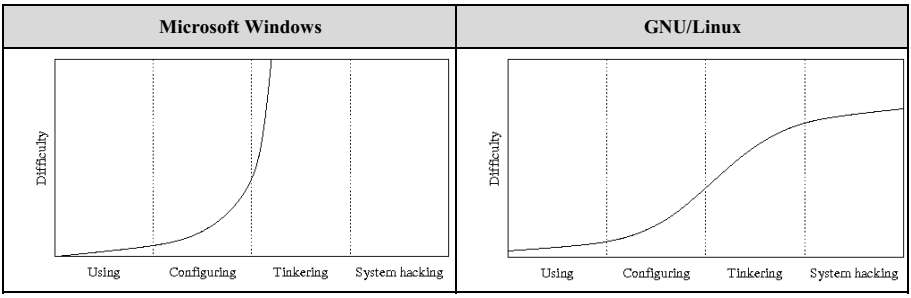

Fig. 1. Learning curves and typical usage of operating systems

If we take a look at the following OS learning curves, we see that there is a progression from beginner to experienced user. It takes some time before a user can properly configure the system to fit his personal needs and preferences. Once a user becomes more familiar with the concept of an OS, what he can do with it, and how he has to use it, the next step in the progression is to be able to configure the system to his needs and preferences. This typically requires more time and it depends on the platform specific built-in capabilities.

1.2 Benefits

Because of the importance of making it easier for users to configure their systems for their personal needs and preferences, the implementation of PC OS autoconfiguration is a key deliverable of the European-funded Cloud4all project. Significant effort has been invested to accomplish this goal and to create a real life demonstration of the benefits of using the cloud-based approach that the GPII project is building right now.

The primary benefit of using the GPII on any PC OS is to take advantage of the Auto-Personalisation capabilities from the users' needs and preferences, which can take into account all the preferences that are coming either from the built-in OS's capabilities, from any third-party application, or from any AT, such as a screen reader for people who are blind, or a screen magnifier for people who are visually impaired. By addressing this implementation, a lot of people will save time and won't have to learn the complex, required platform-specific processes to adapt the system to their needs and preferences.

Another benefit of this implementation is that PC OSs provide more interoperability and flexibility in terms of development, and importantly have significant computing capacity. PC OSs will be the systems where the GPII will run in order to interact with the many different GPII components over the Internet, and taking part in the Cloud infrastructure that will be built around the GPII.

2 Development

The resources that the Cloud4all project consortium are putting into this implementation are mainly focused on helping to build and improve the GPII architecture framework so that it fits the needs to be covered in two of the most relevant Desktops in PCs OSs, Microsoft Windows and the GNOME Desktop on GNU/Linux.

2.1 Goals

From the Cloud4all project perspective, the goals to be achieved in this implementation were explained in detail in the Cloud4all Description of Work document. The main objectives can be summarized as follows:

1. To work with Microsoft, GNU/Linux and GNOME communities, to:
    a. Identify all of the accessibility related features in their respective OS's
    b. Build "auto-personalization from profile" (APfP) capability into:
        i. the built-in accessibility features in GNU/Linux within the GNOME desktop
        ii. the bundled accessibility software with GNU/Linux within the GNOME desktop
        iii. the built-in accessibility features in Windows
2. To develop translation mechanisms to convert the generic preference profiles into specific settings appropriate for the accessibility related features and software on the OS.
3. To demonstrate the ability of each of these operating systems and accessibility applications:
    a. to automatically load all accessibility related preferences from the OS/application up to the cloud (and other preference storage mechanisms) on user request
    b. to download accessibility related preferences for their OS/ application and apply them either permanently or temporarily (so that original settings return when the user is finished) on user request.

2.2 Results

After more than two years of research and development, the Cloud4all project has successfully created the required mechanisms to turn this vision into a reality on PCs. This work included:

- Finding all the relevant common preferences, provided both by the OS and by third-party applications

- Creating the required mechanisms to interact with all these preferences from many different resources
- Providing cross-platform translation mechanisms for the preferences between the different OSs and third-party software
- Identifying how the environmental context and device's capabilities could affect the people regarding the use of their Personal Computer
- Focusing on covering all the most critical aspects in terms of accessibility

To specifically describe what that this means in terms on this particular implementation of the GPII on Desktops PCs OSs, the efforts of the project could be summarized by the following two points.

- To provide a *technical solution* to:
    - Identify and provide the ability to interact with, and to configure, the most relevant accessibility built-in features from both Microsoft Windows and from one of the most popular desktop environments for the open-sourced GNU/Linux platform, GNOME.
    - Identify and to provide the ability to interact with the most relevant settings in the most popular screen readers that are available on the two focus platforms, NVDA (for Microsoft Windows) and GNOME Orca (for GNU/Linux), and provide the required mechanisms to automatically configure their settings.
- To create the *transformation mechanisms* that will bring the ability to switch from one platform into another. This mechanism will not only be available for these two platforms but for others such as the Android platform or any AT solutions running in the same, or in any other platform.

These two points are explained in detail in the following two points, *Technical solution* and *Transformation mechanisms*.

2.3 Technical Solution

Since the GPII Personalization framework was conceived to cover any need on any platform, it shares the same code base for all platforms. It can even be used by platforms that cannot directly run the GPII by themselves through an online instance of the GPII's Flow Manager [4], which is the component responsible for orchestrating the personalization of any solution on any device.

Having said that, from a technical perspective, the implementation of the GPII on PC OSs is mostly focused in covering the gaps between some of the architectural components of the GPII Personalization framework and the OS itself.

Since the GPII architecture has been already explained in detail in A Cloud-Scale Architecture for Inclusion: Cloud4all and GPII [5], and briefly explained in the following schema,

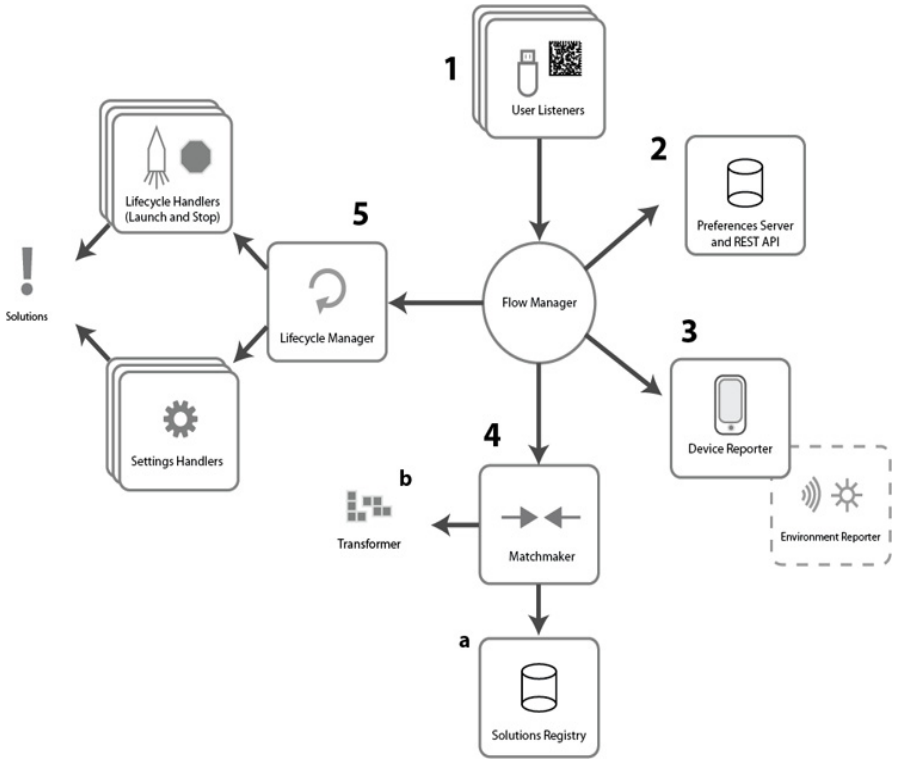

Fig. 2. A *CloudScale Architecture for* Inclusion: Cloud4all and GPII [5]

we would like just to mention those components that are relevant to this specific implementation.

Settings Handlers

In terms of personalization, the GPII's settings handlers are responsible for configuring the settings of an application, access feature, or assistive technology. They are often, but not always, specific to a particular platform, so the GPII provides support for some common standard configuration mechanisms based on well-known formats such as JSON, XML and INI [6].

As an example of a platform specific settings handler, the GNU/Linux implementation makes use of GSettings, the GNOME's configuration system, to make on-the-fly personalizations of the desktop environment, such as:

- Turn on/off some built-in accessibility services and features such as the on-screen keyboard, screen reader, screen magnifier, high-contrast, visual alerts, etc.
- Set specific values to some aspects of these accessibility services such as the magnification factor.

- Change some default behaviours of the user interface that are not specifically accessibility-related such as the preferred applications, background, font-size, windows animations, etc.

This settings handler was created to fill this requirement on GNOME-based PCs. Many other platform-specific settings handlers were written too, including:

- Windows Registry Settings Handler, which deals with Microsoft Windows' registry.
- Windows SystemParametersInfo Settings Handler, which handles Microsoft Windows SPI calls.
- GNOME Orca Settings Handler, which handles GNOME Orca's settings.
- GNU/Linux ALSA, to deal with system's output volumes.
- GNU/Linux Xrandr to deal with the screen's resolution and brightness.

Lifecycle Manager

The GPII has to know about how to orchestrate all these settings handlers in order to automatically configure the system to the user, and this is the responsibility of the lifecycle manager.

In this case, the lifecycle manager usually makes use of well-known cross platform mechanisms, known as *Lifecycle Actions*, to apply a configuration using the settings handlers. They also include some simple platform-specific mechanisms for starting and stopping an application. Since lifecycle actions tend to be simple and can be implemented in a cross-platform compatible way, we haven't had to implement any additional platform-specific component to address this. Instead, we have just to had define, using a declarative JSON specification, how a solution should be treated by the GPII.

Solutions Registry

The solutions registry is where the full information about how to configure a solution is stored, including:

- The application described as a solution
- The settings handler(s) that a solution uses and the way it makes use of it
- The required lifecycle actions to trigger the personalization process
- The available transformations of some settings into the already supported common terms

User Listeners

In order to allow the user to automatically personalize his system through the GPII, a mechanism needs to be provided with every PC OSs implementation called a *User*

Listener. Right now, both Windows and GNU/Linux platforms support the most relevant authentication mechanisms, including:

- USB Drive
- NFC Tag

Since the authentication mechanism is, like the rest of the GPII, an HTTP request, it's very easy to add new user listeners to a platform, such as a fingerprint-based one or by using face recognition.

2.4 Transformation Mechanisms

Nowadays, every software in the world has a lot of settings that allows to any user to make changes on them to fit their preferences. But this can be a problem, because in most cases, each application defines their own settings without taking into account the settings that other similar software has already defined.

One of the biggest challenges of the GPII and Cloud4all projects is to address these problems, by providing the required mechanisms to easily move any application-specific setting into another, and similar in concept, a different application-specific setting. To explain this we can take some similar settings from two different screen readers, NVDA [7] for Microsoft Windows PCs and GNOME Orca [8] for GNU/Linux PCs.

Table 1.

NVDA	GNOME Orca
speech.espeak.rate	voices.default.rate
speech.espeak.pitch	voices.default.average-pitch
speech.espeak.volume	voices.default.gain
speech.espeak.voice	voices.default.family
keyboard.speakTypedCharacters	enableEchoByCharacter
keyboard.speakTypedWords	enableEchoByWord
speech.symbolLevel	verbalizePunctuationStyle

As you can see in the table, these settings are conceptually the same thing but with different names. Also, another implication is that every setting has different value ranges for its values depending on the application. As an example, if we take both the NVDA's speech.espeak.voice and Orca's voices.default.family, their values aren't structured in the same way:

Table 2.

NVDA's *speech.espeak.voice*	GNOME Orca's *voices.default.family*
en\\en	{"locale": "en", "name": "english"}

By having a *common term* as a general way of defining a preference and its value range, implementers can easily map an application-specific setting into a common term, which can be used to move an application specific preference into another application specific preference.

For this reason, and due to the lack of standardization, the GPII and Cloud4All are putting efforts on creating a Common Terms Registry [9] to provide an unified list of settings to be used during the developments of the GPII personalization framework.

Right now, implementers of the Cloud4all project are making use of this common terms registry within the model transformation mechanisms available in the GPII framework [10] in order to move settings from one platform into another. But this is just part of the solution to the problem.The Cloud4all project is investing a lot of resources on this topic, and they are working on creating the required mechanisms to match the needs and preferences of a user, and move them all from one platform into another. Currently, there are several approaches to *matchmaking* mechanisms, and we have successfully made use of them in this implementation to move the preferences from one platform into another. Right now, we are making use of three matchmakers:

- Flat Matchmaker
- Rule Based Matchmaker
- Statistical Matchmaker

To learn more about these matchmakers, and the general matchmaking mechanisms employed by Cloud4All, we recommend the following references:

- Profile Matching Technical Concept [11]
- Matchmaking on the GPII wiki [12]
- Matchmaking Workflows on the GPII wiki [13]
- Improving Accessibility by Matching User Needs and Preferences [14]

3 Delivering the GPII to the End-Users

A critical goal of the GPII is to deliver this infrastructure to end-users, giving them a way to easily access their needs and preferences on any device, with any platform, and in any place. In case of this particular implementation, both Windows and GNU/Linux repositories can be found at GPII's Github project [15][16] and can be installed and used by any user easily.

These repositories contain all the platform-specific source code that, within the GPII's *universal* repository [17], makes this implementations a reality.

Since users will ultimately either accept or reject to use any software, based on how easy it is to use and how useful it is, it's very important to collect all the valuable feedback that any user of the GPII can provide in any early phase of the developments, and incorporating this feedback into the development process.

For these reasons, and as an early outcome of the implementation of the GPII on PC OSs, which is at the same time the most complex and the most mature one, a first pilot phase with real users has been performed in Germany, Greece and Spain. A second pilot phase is running right now. As part of the Cloud4all project, during these pilots, some focus groups with different disabilities have had the chance to try the GPII and to evaluate the current implementation of the GPII on PC OSs. The results from the pilots have confirmed that this upcoming technology will be very helpful to them in their lives.

3.1 Pilots Scenario

In the pilot tests at the Greek, Spanish and German pilot site, we had 130 beneficiaries: 84 were end users (people with disabilities), 28 were developers and 16 were stakeholders.

While the pilots of Greece and Spain took place in premises of the organizations: CERTH/HIT in Thessaloniki and Technosite in Madrid, in the case of Germany they took place in labs and also in the usual environment (e.g common working places of the participants).

Concerning the duration, all pilots took around 3 months. The methods that have been used per pilot site are described in detailed below. They include basically 3 different approaches:

- For end- users: Usability test / Field test, Mock-up paper or click demo, Observation, Structured interview
- For developers: usability test + Observation + Structured interview
- For stakeholders: Focus group

The participants of the pilots were recruited according to different variables:

user group; gender; age; educational background; and operating system used. In the case of the stakeholders and developers/AT vendors, the recruitment didn't follow socio-demographic criteria but aspects related to their expertise.

Since the auto-configuration on PCs OSs was not the only outcome from the Cloud4all project that has been tested on this pilot iteration, here is a full list of validations that have been done during the pilots:

- Cloud4all concept validation
- Preferences Management tool (PMT) with basic functions validation
- Preferences Management tool (PMT) with extended functions validation

- Auto-configuration scenario validation
- Semantics alignment tool (SAT) validation

The auto-configuration scenario validation, which is related to this implementation (PCs OSs), has been tested by 43 users: 16 in Greece, 18 in Spain and 19 in Germany.

All users that tested the auto-configuration scenario started from platform A and moved to platform B and then configured manually platform B and go back to platform A. So, all users tested both the transformation of the setting from Windows to Linux and vice versa, using one matchmaker when going from A to B and the other when returning from B to A.

3.2 Feedback

The users felt that the system met their needs and preferences when they went from Windows to Linux and from Linux to Windows. In both cases, we did not find significant differences between the performance of the system according to their needs and preferences. However the system met their needs and preferences slightly better when moving from Windows to Linux in Greece. When measuring the difficulties that users faced when they had to perform tasks on each platform, they encountered fewer problems on Windows in comparison to when they had to perform the same task on Linux, especially in Greece. The fact that most of the users are only familiar with the Windows OS in their daily life accounts for these results. We can say the users believe that their needs and preferences are less met when going from Linux to Windows.

The users felt that the system meets their N&Ps when they used both the Rule-based and Statistical MM. It seems that both matchmakers work similarly in terms of their ability to provide the proper auto-configuration to fulfill the user's needs. In the case of users in Spain, there were relevant differences between the statistical and the rule based matchmakers. The Rule-Based Matchmaker worked better than the Statistical regarding the configuration of the preferences of these users. In this case the configuration provided by the Rule-Based MM was rated with almost a 5, while the statistical MM with a bit more than 3. In general, both MMs failed more times with the auto-configuration for low vision users than for blind users.

Also, the users gave some recommendations about the system:

- The auto-configuration should be faster and more refined
- It would be very useful to include some information messages that can guide during the auto-configuration process, such as "loading Cloud4all", "logging out from Cloud4all", etc. Visual and spoken messages.
- Low vision users recommended to include more configuration options

References

1. http://gpii.net
2. http://en.wikipedia.org/wiki/Assistive_technology
3. http://en.wikipedia.org/wiki/Interactive_kiosk

4. http://wiki.gpii.net/index.php/A_Detailed_Tour_of_the_Cloud4all_Architecture#Flow_Manager
5. Clark, C., et al.: A Cloud-Scale Architecture for Inclusion: Cloud4all and GPII. Assistive Technology: From Research to Practice
6. http://wiki.gpii.net/index.php/A_Detailed_Tour_of_the_Cloud4all_Architecture#Settings_Handlers
7. http://www.nvaccess.org/
8. https://wiki.gnome.org/action/show/Projects/Orca#About_Orca
9. http://wiki.gpii.net/index.php/Common_Terms_Registry
10. http://wiki.gpii.net/index.php/Architecture_-_Available_transformation_functions
11. http://www.cloud4all.info/pages/news/detail.aspx?id=92&tipo=4
12. http://wiki.gpii.net/index.php/Matchmaking
13. http://wiki.gpii.net/index.php/Matchmaking_Workflows
14. Loitsch, C., et al.: Improving Accessibility by Matching User Needs and Preferences. Assistive Technology: From Research to Practice
15. https://github.com/GPII/windows
16. https://github.com/GPII/linux
17. https://github.com/GPII/universal