StarOffice / OpenOffice.org “Q” Product Concept

Lutz Hoeger, August 2003

Table of Contents

1 Introduction

(Lutz Hoeger, July 2003)

This document provides a comprehensive, high-level overview of concepts for features in the next release of StarOffice / OpenOffice.org (SO/OOo) currently in the planning phase.

The specification for such a large project covers many areas and many details. Consequently, it will not provide a complete list of all the details, but occasionally will include some details to aid in clarifying the discussion. Many more details have already been and will soon be submitted as feature issues in IssueZilla, completing the picture.

In compiling this document, we've gathered feedback from users, developers and existing or potential customers: in direct communication at trade shows like Linux World, during events like the OpenOffice.org Conference, using surveys, and last but not least in discussions on the OpenOffice.org mailing lists. We then put together ideas to respond to these requirements, prioritized them, and finally assembled it all into the Product Concept.

During the prioritization for this release, we had to strike a balance between the importance of each requirement and the goal of an 18-month cycle for releases. We've learned from discussion, particularly with enterprise customers, that this is their expected “heartbeat” for major releases of an office productivity suite.

The structure of this document provides different levels of detail about the concept:

- The Executive Summary (0.5 pages) is ideal for a “Got-only-5-minutes-hurry-up!” reader who wants to know what the overall direction is. It's a brief description of our answers to the highest priority requirements. This section contains neither any details nor a full list of areas we plan to work on.
- The Concept Overview section (five pages) explains the prioritization and discusses all requirement areas. This section should provide a good “30-minute” overview of the responses to the requirements. While leaving out many of the details, it covers all the areas.
- And finally, the Detailed Concept section (about 20 pages) completes the previous section with a much more detailed view into all aspects of the requirements, with some background information if available, and even illustrates parts of the concept by drilling down on some details.

Before beginning, let me introduce one convention used throughout this document. Code name “Geordi” stands for SO/OOo 1.1, and “Q” for the next major release after 1.1.

And one final note, you'll find the document chapters tagged with different author names reflecting that many people contributed to this document. We all would be happy to get your feedback. So if you want to tell us something we could have done better, we may have forgotten or just want to ask a question, please refer in your comments to the section on which you are commenting. This will help us identify the person who can best respond to your feedback.

2 Executive Summary

(Lutz Hoeger, July 2003)

“Q” will be the next major release of StarOffice / OpenOffice.org (SO/OOo). Listed below are the concepts to address the key requirements of our existing and potential customers. These CTQs (Critical to Quality) are all centered around the main theme: migration to “Q”.

- Lower cost of interoperability with Microsoft Office. By providing better interoperability with Microsoft Office, “Q” will significantly decrease the cost of conversion between the two applications.
- Lower cost of retraining
  “Q” will lower the training needs for average users by enhancing feature areas toward better standards of interaction. This will make the migration to SO/OOo cheaper.
- Better Performance
  “Q” will come with noticeable improvements in startup performance as well as time to open or save a document.
- General usability improvements
  Redesigning parts of “Q” to increase usability will increase productivity of the average user and the overall level of satisfaction when working with the product.
- Enhanced programmability
  “Q” will make it easier to automate typical office suite tasks. By improved extensibility, “Q” will be better customizable to individual customer's needs.
- Better integration into desktop systems
  Particularly on Gnome desktop systems, “Q” will integrate better with the existing infrastructure. This leads to a more intuitive use of “Q” for users who are already familiar with the desktop system, and provides better interoperability between SO/OOo and other desktop applications.

3 Concept Overview

This section explains the prioritization and provides a general overview about all concept areas. More details are given in section 4, Concept Details.

The priorities used in this concept have been set according to the general guidelines of priorities from different bug tracking systems, including IssueZilla. Although the latter doesn't specifically exclude feature issues from having priority 1, a P1 issue normally means that an area is broken and cannot be worked with. Typically P1 is reserved for bugs only. That's why the scale for requirement priorities ranks from P2 (highest) to P5 (lowest). We assigned them in the following way.

3.1 Key Drivers (P2)

The overarching motto and strategy for this release is to lower the barrier for customers to migrate to StarOffice/OpenOffice.org (SO/OOo). Our concept addresses this at various levels, in different areas of the product. The following areas describe the key drivers for "Q" - the areas that we hear about most in our customer research. These are the areas with highest priority and also often the highest effort in development engineering.

3.1.1 Microsoft Office Interoperability

(Dieter Loeschky, April 2003)

Interoperability between Microsoft Office and SO/OOo is generally influenced by three factors. The first factor is the quality of the filter component, that is, the quality of the component that translates the content and structure of a Microsoft Office document into the content and structure of a SO/OOo document. The second factor is the compatibility of the feature set of the application in Microsoft Office versus the corresponding SO/OOo application. The third factor is the behavior or interpretation of features that exist in both applications. Even if another application and SO/OOo both support a certain feature, there might be differences in the way the feature is actually applied, so that documents in fact might look different although they have exactly the same values stored for the feature within their text engines.

One (already resolved) example of this is the upper paragraph margin that exists in Microsoft Word and SO/OOo Writer. The latter, like many DTP and professional word processing applications, ignores this value for the first paragraph of a page. Microsoft Word does not. The result of this is that the documents look different, although exactly the same values are stored in both text engines.

3.1.2 Usability

(Matthias Mueller-Prove, May 2003)

"Q" will change its overall appearance in order to improve the usability for the majority of non-SO/OOo customers. These changes affect the menu structure, the toolbar User Interface, the terminology, and finally the overall window layout.

In general, usability is about task conformance, familiarity, predictability, flexibility, robustness, customizability, and learnability. Several minor usability improvements support the usability of SO/OOo in aspects of these usability qualities. All new features will be evaluated against these qualities by the Sun StarOffice User Experience Team.

Task conformance pushes us to reconsider the necessary steps for an action and reduce the number of mouse clicks in "Q" as much as possible. Predictability demands that we strive for a consistent user interface. We will provide a conceptual model that is predictable and consistent for all SO/OOo applications instead of conforming with the majority of competing applications. On the

other hand, predictability calls for conformance with styleguides for the target platforms.

3.1.3 Performance

(Matthias Huetsch, May 2003)

SO/OOo will improve its performance in four areas that are especially important to customers. The two most visible areas of improvement will be decreasing the Startup Time and Document Open/Save Time. The third area of improvement will be Responsiveness in rendering more complex drawing objects as they, for example, occur in presentations. The fourth area is in the Scalability of large scale multi-user deployments like on Citrix, Tarantella, and SunRay servers.

The performance improvements in the first three areas were identified as being mostly of an architectural nature. Significant improvement in any of these areas can be achieved through either refactoring the current architecture, or replacing architecture. The fourth area of scalability issues originates in a number of areas that can be addressed individually.

3.1.4 Programmability

(Kay Ramme, May 2003)

Programmability describes the possibility of usinge SO/OOo or parts of it in custom solutions programmatically. The underlying technology for SO/OOo programmability is the SO/OOo component model UNO (Universal Network Objects) and the SO/OOo API . “Q” will make it easier on many levels to create solutions utilizing UNO and SO/OOo's API. This will be achieved through improved language bindings, IDE integration, easier deployment of add-on components into a SO/OOo installation, and a new scripting framework that allows use of other languages in addition to BASIC for scripting purposes.

3.1.5 Integration into Gnome Desktop and Microsoft Windows

(Christof Pintaske, June 2003)

The overall requirement is to provide an easy to use desktop with an appealing and modern Look and Feel that is consistent throughout all applications.

To achieve a better system integration, "Q" will make increased use of information and infrastructure that is provided with the Gnome desktop instead of creating platform independent solutions.

3.2 General Improvements (P3)

The next areas are still mandatory requirements. They fall under general improvements, that help facilitate a successful deployment of SO/OOo.

3.2.1 Stability and Quality

(Matthias Huetsch, June 2003)

“Q” improves stability and quality by addressing the response to SO/OOo crashes with unsaved documents, which can result in lost data. Currently, SO/OOo performs an emergency backup of those documents only. Upon next startup, a Document Recovery dialog then offers to restore those documents to a previously known good state.

The probability of data loss due to a crash will be reduced by augmenting the current emergency

backup with regular backup copies of documents. The Document Recovery dialog will be improved by providing more guidance and feedback while restoring documents. An Error Report tool, available in the “Geordi” release, should provide data to help identify and eliminate crash causes.

3.2.2 Administration and Deployment

(Dirk Grobler, Christof Pintaske, May 2003)

With regard to deployment, "Geordi" offers a platform independent installation process, which can not easily be managed by any kind of management software. Instead of adding additional management capabilities in this area, SO/OOo "Q" changes the installation process. The platform independent installation is replaced by the support for standard, platform dependent installation routines - so called native installers - and their native installation packages. These are the Solaris Package (pkg) installation, the Linux RPM package manager (RPM) installation, and the Microsoft Installer (MSI) installation.

The advantage of this approach is to leverage already existing deployment technology based on those native formats. With "Q", administrators will be able to install "Q" with their existing tooling.

3.2.3 Modern Look & Feel

(Christian Jansen, May 2003)

Users are asking for various look and feel improvements, which fit mainly into three categories.

- The SO/OOo look and feel themes should be more intuitive.
- The user interface should be better organized and emphasize the important information.
- SO/OOo needs to provide more user feedback and responses.

For SO/OOo "Q" we will address these areas by improving the look & feel and adjusting it to the expectations of the majority of non-SO/OOo users while balancing it with the expectations of current SO/OOo users.

3.2.4 Integration (Plugins, 3rd party applications)

(Mathias Bauer, May 2003)

Integrating SO/OOo with 3rd party applications can be done on several levels:

- Create import and export filters to convert between the file formats.
- Use the system clipboard for data exchange.
- Use external communication means like pipes, files, command line parameters etc. to establish a work flow that uses SO/OOo and other applications together.
- Link or embed data or files from other applications in(to) SO/OOo documents and vice versa.
- Use the SO/OOo API or the API of other applications to integrate on a functional level.

The first two options are already widely used in SO/OOo. Adding support for more 3rd party applications simply means providing the necessary conversion routines and will not be discussed here in detail.

External communication means are usually tailored for every single use case. Examples for this kind of integration are external e-mail clients.

3rd party integration on an API level can be done in two ways:

- the 3rd party application uses UNO to connect to SO/OOo and uses its APIs to get the wanted functionality
- SO/OOo creates adapters to use the middleware technology of the 3rd party application (like Com, .NET or SOAP).

This concept document concentrates on the first approach. The latter one is addressed by UNO bridges like the ones we have (or will develop) for OLE Automation, .NET or SOAP.

3.3 Upcoming Opportunities (P4 and P5)

The final part of the concept contains optional enhancements in the area of upcoming opportunities that we can foresee today.

3.3.1 Asian and CTL languages

(Falko Tesch, July 2003)

Internationalization (I18N) describes the ability and availability of a product for international markets. This means that the product has to comply with 3 important issues:

1. It needs to address locale specific features and customs.
2. It needs a good and comprehensive documentation and UI.
3. It needs localized tools and support for local utensils and tools, such as IME (input method engine) and the like.

StarSuite 6.0 and "Geordi" support IME that enables the input of East-Asian scripting like Chinese, Korean and Japanese. Furthermore "Geordi" pays tribute to the most needed features for text processing in Asia, like vertical text layout, Ruby support, specific font effects and other. While "Geordi" covers features and needs for CJK markets, it takes only the first steps towards a specialized offer to the CTL (complex text layout) markets (Arabic, Hebrew, Thai, Hindi etc.). “Q” will complete the CJK (Chinese, Japanese, Korean) feature set and complete CTL features started with "Geordi":

- Refining already existing CJK features and adding competitive features to fully compete with other office productivity suites for CJK, like local office software, such as Hangul (Korea), Ichitaro (Japan) and WPS Office (PR China).
- Introducing full localized CTL support for Arabic and Hebrew-spoken countries as well as for India (Hindi plus 6 other major languages spoken)

With “Geordi”, SO/OOo demonstrates it's suitability for the CTL markets (e.g. Arabic and Hebrew regions)

3.3.2 Digital Signatures

(Michael Brauer, May 2003)

Security for office document content can be divided into two features, digital signatures and encryption.

Digital signatures themselves are a relatively new topic for office applications. The requirement to protect data from being modified has existed for many years , however. In the past, it has been addressed by features that protected documents from being edited within the office application. With digital signatures, these features will be enhanced to offer secure protection of document

content, inside SO/OOo, and outside of it.

3.3.3 Compelling upgrade features (e.g. multi media, OCR, DB)

Encryption is a feature that has been supported by office applications for a long time. Enhancement in this area mainly affected the encryption algorithm themselves, to make them more secure. However, since there was no standardized way in which encryption algorithms are applied to documents, processing such documents outside an office application was complex. By supporting new XML encryption standards, and due to SO/OOo's XML file format, this will become much easier.

Although the requirements cover many improvements and new features for SO/OOo, this section is reserved for features that make the upgrade particularly compelling for existing SO/OOo customers. Since we don't know of any specific customer requirements that fit exclusively into this section, it's content will be determined later.Web Services

(Kai Sommerfeld, May 2003)

Web Services become more and more important as they easily can be used to establish B2B communication in a platform independent way. There are lots of Web Services available that offer a broad range of functionality, for example Google offers a Web Service for searching the Internet.

It would be valuable, if SO/OOo could act as a Web Services client, meaning that it could access and use arbitrary Web Services.

Additionally it could be interesting to offer SO/OOo's functionality as Web Services. An example for such a Web Service is a document transformation service, which would be based on SO/OOo's document import and export.

3.3.4 Intelligent Document Tags

(Andreas Martens, May 2003)

SO/OOo will support an interface for intelligent and extensible document tags. This interface will allow third party vendors to offer additional information/services on the basis of textual analysis. This is a win-win situation. SO/OOo gets a feature without the effort of reinventing the wheel. For third party vendors it creates a business opportunity around SO/OOo.

Initially, all words/phrases which are recognized as keywords will be underlined. The user is able to open context menus for these marked words and to choose additional actions.

Example: A vendor offers a component with different dictionaries (technical, juristic, medicinal). The SO/OOo document will be scanned for entries in these dictionaries. For marked words the context menu contains the names of the dictionaries where the word has been found. The user chooses a dictionary and the component displays the complete entry.

Example: The supporting component is based on a customer list. If the name of a customer is present in a SO/OOo document it will be underlined. The context menu may offer actions like “address”, “send email”, “show business volumes”, “show last order” et cetera. The chosen information will be displayed or another program like an email client will be started.

3.3.5 Collaboration

To be completed later

4 Concept Details

This section contains details about the product concept.

4.1 Key Drivers

4.1.1 Improving Microsoft Office Interoperability

(Dieter Loeschky, April 2003)

In the past, interoperability between Microsoft Office and SO/OOo was improved by completing the import and export filters. In "Geordi" interoperability between Microsoft Office and SO/OOo was improved by also adding features to SO/OOo and by changing SO/OOo's behavior and interpretation of single features.

For “Q”, interoperability will be improved in a very large scale by continuing to add features to SO/OOo. Changes in the filters are of course still required to support the new features or changed behavior.

The movement of effort from the filter development into the application development is in fact a very natural process. At a certain stage, interoperability of a certain feature cannot be improved by the filter any longer, because it already converts all data that is supported from one application to the other. At this point, if the feature still behaves differently, the SO/OOo application itself may be enhanced.

4.1.1.1 SO/OOo Writer

For SO/OOo Writer, there are four main areas in which interoperability with Microsoft Office will be improved: tables, drawing objects, numberings/bullets and spacing. The improvements in these main areas will be completed by many other small enhancements in other areas that also raise interoperability issues.

Tables

The most important improvement will be the support of automatic page breaks within cells. Currently, this can cause significant differences in layout of the same document between a non-SO/OOo application and SO/OOo Writer if the document contains tables that do not fit on a single page. Two other important improvements will be the support of tables within tables and of tables where text flows around them. Table interoperability will be completed by a border model that provides better interoperability with other applications, including Microsoft Word. Interoperability improvements mentioned above are considered to be a much higher priority than those that can be achieved by introducing a different table model.

Drawing objects

A second very important area where interoperability will be improved is in SO/OOo drawing objects. The most important improvement will be supporting other applications' drawing objects within page headers and footers. In addition to this, missing alignment, anchoring and position modes will be added. Finally, there will be a unification of drawing objects and SO/OOo Writer's so called fly frames, i.e. graphic, text box and OLE objects. Many options that are available for frames will be made available to drawing objects as well and vice versa. The user interface of both kind of objects will also be unified.

Numbering and bullets

Interoperability of numberings and bullets, the third main area mentioned above, will also be addressed by many small enhancements. Negative indents will be supported and it will be possible to change the indent values for every paragraph individually. Tabulators will be supported within numbers. Last but not least, the the calculation of chapter numbers and table of contents will be enhanced.

Spacing

The fourth important area for improvement can be summarized by the word “spacing.” Although the differences are very small (typically less than half a point) and will hardly be noticed by the user, the end result might be that an imported page contains two lines less of text on a SO/OOo Writer page. This again can result in images and text boxes appearing on different pages, and the document may look completely different in SO/OOo Writer.

4.1.1.2 SO/OOo Calc

Most interoperability problems between Microsoft Excel and SO/OOo Calc that exist today result from an incompatible feature set. The biggest advance in interoperability can be achieved by adding new and changing existing features in Calc to achieve better compatibility.

Breaking the row limit

Currently, 32,000 is the maximum number of rows in a Calc spreadsheet. Cell content beyond row 32,000 is lost when a file with more rows is imported into SO/OOo. If more than 32,000 rows are needed for a specific spreadsheet, there's no easy way to get equivalent results using only SO/OOo's 32,000 rows. It is therefore important to adjust our row limit. The number of rows in a spreadsheet will be increased to 65,536, with provisions to make it relatively easy to further increase that number in later versions.

Calculation

The second important issue is calculation. Although the set of formula functions is very similar across most spreadsheet applications, some special-case features of calculation with these functions are not yet supported in SO/OOo. Currently, when documents that use these additional features are loaded in SO/OOo, the formula cells will show error values or even different results from those in the original document. To get the same results as in the original, the spreadsheets would have to be changed manually. Especially in the case of array formulas, these required changes would be quite extensive, making it difficult to create an equivalent calculation in SO/OOo. We have identified a set of features that cause problems.They will be addressed in "Q".

Form controls

Another important feature that will be added to SO/OOo is the ability to create form controls (list boxes, check boxes, etc.) that are connected to spreadsheet cells. These controls get their values from the cells, and update the cells if something is selected. With the increased flexibility that will be needed to support the form layer's own new features, we will be able to also implement this connection to spreadsheet cells.

Cell attributes

SO/OOo supports slightly different cell attributes than competing applications. This leads to differences in the appearance or behavior of imported documents that contain these attributes. We will add corresponding features to SO/OOo to eliminate the differences in imported documents. We will extend the validation feature, annotations, scenarios, ranges and hyperlinks, as well as some cell formatting attributes.

DataPilot

Currently, basic functionality and interoperability is available in SO/OOo's DataPilot. In "Q", we will extend the DataPilot to the state-of-the-art feature set for reports based on spreadsheet cell data. The most important of these features are page fields, settings for fields and individual elements, and the table formatting options.

Sorting

Finally, it's common amongst the competing applications to preserve as much of the original sorting order as possible, when sorting a spreadsheet according by some sort keys. In SO/OOo they end up in a random order. We can change the implementation of sorting so that the original order is preserved where sorting criteria are equal.

4.1.1.3 SO/OOo Impress

"Geordi" improved interoperability with Microsoft PowerPoint in many aspects. To further improve interoperability completely new features need to be added to the application and the drawing layer.

Build effects

The current animation system of SO/OOo Impress supports one effect per shape with all effect types individually implemented. Execution of all effects is either automatic or manually per slide. Users request to have an equivalent set of effects and to preserve the order of execution of effects when exchanging presentations between SO/OOo Impress and non-SO/OOo applications.

Instead of manually implementing every new shape effect, it is necessary to support an animation core that is flexible to handle new effects that may be added later. The first step is to add a core component that can fully load and store non-SO/OOo effects for improved interoperability. Due to the higher demands for hardware supported rendering this first step will not be visually equal on all effects. Also playing the effects in real time will not be solved by this.

Timeline

To support a timeline concept, there will be a timeline feature for SO/OOo Impress. With this timeline, interoperability is enhanced by preserving the order and timing by which the shape effects are executed. Users will be able to add more than one effect to each shape and to have effects that are partly automatic and partly triggered by mouse clicks.
This will also include a user interface to change the new introduced timeline and effect features. “Q” will also provide a shortcut, for the user to select an animation theme for a slide, so that the slide and shape effects are automatically set to fit the theme.

Collaboration

Interoperability with other presentation programs, including Microsoft PowerPoint, will be improved by adding means to use sticky note like comments in Impress. Currently, upon import, SO/OOo Impress ignores this information. This improvement will enhance interoperability by preserving document content and by giving SO/OOo Impress users who need to interoperate with other application users the ability to exchange comments on their work.

Header and Footers

Sometimes headers and footers in Impress presentation slides get lost during a roundtrip from Microsoft PowerPoint to SO/OOo Impress. The reason for this is the different way headers and footers are used in Microsoft PowerPoint and SO/OOo Impress. The SO/OOo Impress method for using headers and footers is much more flexible than Microsoft PowerPoint's.

To meet this requirement and to improve ease of use, SO/OOo Impress will provide easier access to it's header and footer feature. This will not only enhance interoperability by keeping header and footer information persistent, but will also provide the user with an easier interface to create them in SO/OOo Impress. There will also be an easy to use dialog to change the header and footer for the current page or for all pages.

Extended Drawing Shapes

Often drawing shapes, such as "AutoShapes" in Microsoft PowerPoint, can be found in competing presentation program applications. They provide an easy way of adding drawings to presentations. Therefore a set of predefined shapes of different categories like 'block arrows', 'flow charts', 'stars and banners', 'callouts' and 'action buttons' can be provided.

For example, in the current version of SO/OOo, imported Microsoft Office documents that contain Autoshapes are mostly displayed correctly. There does not exist, however, the ability to edit WordArt objects or to change 3D-objects. Interchanging documents between SO/OOo and Microsoft Office leads to the loss of functionality of these drawing objects.

To meet this requirement, an implementation for extended drawing objects in SO/OOo will be designed, including new dialogs to edit imported WordArt objects and to change imported 3D objects. This will not only enhance interoperability through supporting all Microsoft Office AutoShapes, but make them fully editable and allow SO/OOo Impress users to create functionally equal drawing shapes themselves.

4.1.1.4 SO/OOo Base (Forms)

An evolving standard for data entry applications are W3C's XForms. XForm controls can be embedded into any XML based file format like XHTML, and they support data entry based on XML schemes. It is likely that XForms will replace at least HTML/HTTP forms in the mid term future. By adding XForms support to SO/OOo, it will not only keep up with recent development trends, but in combination with the OASIS OpenOffice XML file format it also becomes a powerful alternative to XHTML/XForms. What makes it interesting to use an OASIS OpenOffice XML and XForms based SO/OOo as a data entry client is that it offers much more layout capabilities than XHTML/XForms based clients, but uses exactly the same open protocols as XHTML/XForms based clients on the server side. Moreover, since SO/OOo is also a powerful tool to create forms, even form creation gets easier and does not require additional tools.

While XForms are already an improvement for data entry applications, it is conceivable that data entry applications will improve further. In some of the recently developed applications, data entry is not limited to classical form controls, but arbitrary text or other content can by included into the form data. Additionally, some applications provide the ability to extend forms at run time. An example for this are meeting minutes, where an agenda item might be duplicated any times. These capabilities allow it to use data entry applications not only for static and simple forms, but also for such complex things like minutes, bills or letters. In any case, the actual form data will stay in an XML schema. These schema could be either a user defined one, or one that is standardized by organizations like OASIS. In the later category are the UBL schemes that are currently developed by an OASIS technical committee.

In such scenarios, the user creates new documents based on existing document templates. In the new document, data can be entered only at predefined locations, for instance within certain fields, controls or text boxes. Moreover only very few editing features will be enabled. For instance, text sections might be duplicated but not inserted at any location. This way, it can be ensured that the document always has a defined structure. This again allows it to bind every editable data to elements of an XML schema and therefor to submit them as XML form data.

SO/OOo capabilities for such form based work flows will be improved. The starting point is the XForm enhancements described above. Further steps will be the inclusion of for instance field or paragraph contents into XML form data and possibilities to enter data into them in Writer's read only mode. The capabilities to specify application behavior within templates will be improved. Finally, certain complex editing operations, like duplicating a section, will be allowed in Writer's read only mode as well. By adding data entry capabilities to SO/OOo step by step, more and more work flows can be implemented with SO/OOo.

SO/OOo will also improve its interoperability with the form functionality of Microsoft Office, in particular of Microsoft Word. Microsoft Word features two completely different concepts for form functionality: form fields and form controls. Form fields are the more ancient concept and are not supported by "Geordi", but are nevertheless frequently used by Microsoft Word users. SO/OOo will address this by implementing equivalents for these form fields.

Form controls offer a functionality similar to form controls in SO/OOo applications. For “Q”, we will implement some new control types like scroll bars, toggle buttons, and a rich text control, and enhance existing control types with functionality currently not present, for example multi-columnar list boxes, and spell checking in text controls.

4.1.1.5 SO/OOo Chart

Additional functionality will be added to the Chart core. This added functionality will ease Bug fixing and integration of new features.

The most important feature that will be added will allow data series to get their input from individual cell-ranges for x-values, y-values, labels, etc. Without it, some imports from non-SO/OOo spreadsheets can only produce an empty output. Technically, this requires new interfaces for the communication between SO/OOo Chart and either SO/OOo Calc or SO/OOo Writer. Moreover, the source data concept within the SO/OOo Chart needs to be redesigned completely.

Some new chart types have to be integrated too – especially bubble and surface charts. A bubble chart requires an additional input range for the bubble sizes. As this additional range does not fit into the current source data structure the new source data concept mentioned above is a prerequisite for offering a bubble chart as well.

Moreover, the integration of a new chart type into the old SO/OOo Chart core could jeopardize existing functionality, as chart type dependent code is spread over the whole project. To eliminate

this fundamental problem a new SO/OOo Chart core will encapsulate chart type dependent code in separate UNO components with well defined interfaces.

The re-implementation of the SO/OOo Chart core is a necessary, long-lasting task that has to be finished before any of the requested new features can be started. The re-implementation of the current functionality is expected to take at least until the end of the calendar year.

The missing features for interoperability with competing applications include:

- Flexible source range selection for different parts of data series (x-values, y-values, labels etc.)
- Flexible combination of different chart types (mostly line & bar)
- Free positioning of data point labels (often requested for pie charts)
- Take format for data point labels from corresponding SO/OOo Calc cell
- Pull out 3d pie segments (often requested)
- Pseudo 3d look (this type of 3d charts is often used in magazines)
- Bubble chart
- Surface chart
- More flexible legend (resizable, removable entries)
- Arbitrary content for data point labels
- X error bars
- Filled radar chart
- Pull out outermost segments of 2d donut charts

The order of the items reflects a proposed prioritization. The highest priorities are given to those missing features that cause more information loss and are used more often.

4.1.2 Usability

(Matthias Mueller-Prove, May 2003)

4.1.2.1 Menu Structure

The redesign of the menu structure will take the general concepts of the competing applications into account. Two prominent examples of menu changes for “Q” are the organization of menu items of the View menu and a new Table menu for Writer.

4.1.2.2 Toolbars

The user interface of toolbars will be completely redesigned. “Q” provides for small toolbars that can be rearranged with drag'n'drop. The new design makes better use of the screen estate by using the space more efficiently. The dynamic exchange of toolbar content will be eliminated.

4.1.2.3 Terminology

The terminology used for the user interface for menus, prominent dialogs and alert messages will be revised in order to utilize the existing knowledge of office productivity users.

4.1.2.4 Window Layout

Impress will get a new window layout that introduces a tool pane for SO/OOo. The tool pane offers specific commands for presentations to make it more easy for the user to manage and edit slides,

import styles and slides from other files and to run her presentation.

4.1.2.5 Application Separation

For the user, “Q” will also look more like separate applications. The Microsoft Windows Start menu / Linux desktop panel, SO/OOo's “File” and “Window” menu, document window titles, and the options dialog will reflect this change.

Along with this new concept we will eliminate obsolete settings from the options dialog and make clear which settings have effects in which part of the application.

4.1.2.6 General Interaction

The interaction model for text, tables and images in Writer and for objects and layers in Draw and Impress will implement state-of-the-art selection techniques.

For “Q” we will revise the interaction concepts for compound tasks like mail merge and creating charts.

Finally, the enhanced Undo functionality will significantly improve robustness of the product.

4.1.2.7 Usability of the Database application

(Frank Schoenheit, May 2003)

Currently, the database access (DBA) functionality in SO/OOo exposes various differences from other database applications. Those differences can be assigned to two groups: usability and interoperability. These issues are predominant with users who are not able to locate the functionality they need (though often it is present). They then do not feel comfortable with SO/OOo's user interface which follows a different philosophy in this area.

Additionally, having a competitive database application, instead of a set of database access components, clearly is an important feature for potential users and customers.

We will address this with a redesign of SO/OOo's database application. In the first instance, this will lead to it being recognized as a state-of-the-art end user oriented database application, and make former users of other database products, like Microsoft Access, feel more comfortable. In the second instance, it will address most of the problems customers have with the current user interface.

A New Database Application

SO/OOo DBA (with a new name which is to be discussed) will be an own application, instead of hiding within Writer, Calc and Impress documents. This will be perceived as a competitive database application then, which will be equal to the other major applications in SO/OOo.

This new application will be accessible from the main menu – File|New is where users expect to create a new database, so we will add a Database item, which opens the application and offers the user a reasonable set of choices how to proceed (e.g. create a dBase data source, connect to an existing foreign database via an auto pilot).

We will not abandon the concept of data sources. This is not only for reasons of feasibility (a database engine is a very heavy-weight project), but also for reasons of openness: We continue SO/OOo's philosophy to allow the user to work with whatever database is already present in her work environment, from client-side single-user file-based databases up to server-side databases participating in a corporate work flow.

The layout and user interface of this new application will, at the first glance, be recognized as

relatively similar to what typical database application users already know. Where it makes sense, the application will provide a look&feel, including functional concepts, that are intuitive.

As a consequence of this new database access application, we will give it its own top-level topic in SO/OOo's help system. This will increase the user's ability to find help with database topics.

The functionality of the current Data Source Administration dialog, and large parts of the functionality of the current data source browser, will be merged in the new application, providing one single entry point for database functionality in SO/OOo.

Data Sources

We will re-implement the user interface of SO/OOo DBA. We will move away from the current technician-oriented user interface to a more end-user oriented work flow and terminology.

In addition, several measures will be taken to make the “data source” concept feel more like the “database” concept the user is used to from other office suites. Though the underlying core concepts will not be changed, SO/OOo users then will have less problems when migrating from other database applications.

We will change the way that data sources in SO/OOo are administrated – we will put more focus on meeting the user in her own vocabulary of concepts, by creating a more intuitive user interface, and by exposing most of the technical details to advanced users only.

An auto pilot will be created which allows the import/export of data sources (file based), and thus migrate them between installations. This will not only unburden the user from the manual work which is currently required for this, but also support the impression of a “database” in opposite to a “data source” which the user cannot really control.

Form Documents

Form Documents provide a highly customizable view to database data, thus they are the user's primary choice when working with such data.

The main improvement desired by SO/OOo users in the area of form functionality is usability. Thus, we will make changes to SO/OOo affecting the user interface in several ways. The user will recognize SO/OOo data-aware forms (which covers form design as well as form usage) as a dedicated solution, in opposite to the current situation, where form functionality is recognized as a small and hidden add-on to SO/OOo text documents.

This list ranges from small and rather cosmetic changes which will sum up to a much better user experience, to larger projects such as a dedicated simple mode for the property browser, which consists of specialized views for the different object types, and presents only the most often used aspects of these objects to the user, this way exposing the full power of the property browser to advanced users only.

Additionally, functionality enhancements will be made to SO/OOo's form layer. Included here are additions to the form controls which are required in the Microsoft Interoperability Chapter, but also new implementations (such as real-time filter controls) which will bring SO/OOo nearer to the current state-of-the-art.

Common Tasks

We will provide the user with more auto pilots for the most common tasks in the database area,

respectively enhance the existing auto pilots. In SO/OOo “Q”, we will have an enhanced report and form auto pilot, and new pilots for designing new tables, and designing customized data views aka queries.

4.1.3 Performance

(Matthias Huetsch, May 2003)

4.1.3.1 Startup Time

The most visible performance improvement in SO/OOo will be the substantial shortening of the time required for first startup, in particular when the system has not been used for some time or after a system reboot. The improved first startup time will correspond to the time used by competing applications.

Generally, the decrease in startup time will be addressed through refactoring the code and removing the strong interdependencies between:

Application 'Core', 'User Interface' and 'Filter' code

- Application code and 'Common User Interface' code spread over several shared libraries
- Application code and 'Accessibility Helper Class' code spread over several shared libraries
- Several large shared libraries tied together due to the use of a global initialization pattern

In combination, these refactoring actions shall result in loading a relatively lightweight framework plus the necessary application code and document filters upon startup, only, and will thus reduce both the startup time as well as the initial memory consumption.

4.1.3.2 Document Open/Save Time

The second most visible performance improvement in SO/OOo will be decreasing the time required for opening or saving larger documents, in particular for documents in the native XML format. This time improvement will be comparable with competing applications.

SO/OOo will implement this time savings with SO/OOo API enhancements that are optimized for XML import / export and are fully supported by the application internal implementations. Code optimizations along the entire import / export component chain will be evaluated to identify and eliminate bottlenecks.

Responsiveness

"Q" will speed up the graphics display and increase the responsiveness in rendering complex drawing objects.

This improvement will be addressed with a redesign of both the graphics toolkit, adding e.g., support for modern hardware acceleration, and the presentation engine, making use of double-buffering and other caching techniques.

4.1.3.3 Scalability

Large scale multi-user deployments in which possibly hundreds of SO/OOo application instances are running on a single multi-processor server will be improved beyond what “Geordi” already addresses. Scalability issues can be summarized and addressed as follows:

System Load

A high number of context switches per second is caused by SO/OOo instances temporarily yielding control to other processes or threads. This behavior induces a high load on the system scheduler, and may as well adversely affect the responsiveness of SO/OOo itself.

This will be addressed by giving up current cooperative and timer based polling concepts in favor of preemptive and event driven concepts.

Memory Consumption

The amount of non-shareable memory per SO/OOo process to a large extend determines the number of instances that can run simultaneously on a given server machine.

This will be addressed by reducing the size of writeable data segments in shared libraries and special purpose memory allocators for compact mass object storage.

4.1.4 Programmability

4.1.4.1 UNO/Office API Improvements

(Stephan Bergmann, Daniel Boelzle, Kai Sommerfeld, May 2003)

The Office UNO API is very fine grained. Although this offers maximum flexibility, programmers often want to have simplified high-level UNO interfaces. This enables rapid application development and helps even lower-skilled developers to reach their goals quickly. To address this problem we need to provide a new set of task-oriented high-level convenience UNO services.

UNO defines language independent concepts like services, interfaces or the Any data type. The binding to concrete languages often maps these concepts One to One. This is not always the optimum because it may require the developer to learn constructs that are unusual or even strange for a certain programming language. A popular example for this is the queryInterface call, which should be a simple cast in Java. For the next Office release we will provide new UNO features which will simplify the usage of UNO in general or for particular language bindings.

4.1.4.2 Integrated IDE for multiple scripting languages

(Thomas Benisch, May 2003)

With the language independent scripting framework it will be possible to script SO/OOo in any supported scripting language. This raises the need for a lightweight scripting IDE, which is integrated into SO/OOo and supports multiple scripting languages including SO/OOo Basic, Java and JavaScript.

The scripting IDE will be based on a language independent IDE binding concept. A scripting language can only be integrated into the IDE, if the corresponding interfaces are supported by the scripting runtime.

It is planned, that the current Basic IDE and its dialog editor will be refactored and modularized, which will be the base for the new scripting IDE.

4.1.4.3 Type Life Cycle

(Daniel Boelzle, May 2003)

The Universal Network Objects (UNO) Component Model with its support for different languages gains popularity. UNO paves the way to access the SO/OOo API and development of extensions by 3rd party developers.

This way, developers have to face quite some problems introducing and using UNO types concurrently. If it is feasible, it is desirable that UNO types have a controlled life cycle.

4.1.4.4 .NET Bridge

(Joachim Lingner, May 2003)

SO/OOo uses software bridges in order to have pieces of code, which may be subjected to certain language- and runtime specific requirements and restrictions, interact with each other. For example, Java code or C++ code cannot interact naturally. The same goes for C++ code compiled with different compilers.

All those particular requirements to a piece of code is referred to as environment and it is said that a piece of code lives (or runs) in an environment. Now, to have code from one environment access code in another environment, one needs to have a bridge. Because of the multitude of possible combinations of environments, bridges use an intermediary environment, namely the UNO environment.

If a new compiler is to be used or software written in a new language, then one only needs to provide a bridge into the UNO environment and all other environments for which there is also a UNO bridge may interact with the new environment.

The CLI – UNO bridge enables programmers to use languages which produce CLI (Common Language Infrastructure) compatible code. This includes languages such as C#, C++ .NET, and Visual Basic .NET.

4.1.4.5 Live Deployment and GUI

(Daniel Boelzle, May 2003)

The development of 3rd party extensions for OpenOffice.org using the OpenOffice.org API is gaining popularity. The common tool to deploy those extensions into an Office installation is pkgchk. The user raises the pkgchk deployment tool specifying a package file to be installed. The tool then scans the content of the package file and handles different content types with respect to their file extensions:

- UNO shared library components (.dll, .so files)
- UNO Java components (.jar files)
- UNO Python components (.py files)
- UNO typelibrary files (.rdb files)
- SO/OOo Basic scripts and dialogs (script.xlb and dialog.xlb files)
- Configuration schemata and data (.xcs, .xcu files)

Currently users have to shut down a running office process before running the pkgchk deployment tool adding or removing packages. It may be desirable that users can deploy packages while running an office process and trigger removal of existing packages. Moreover, the currently used pkgchk deployment tool is a command line tool. If deployment can be performed while running the office process, a proper user interface is nice, too (e.g. in “Tools” menu bar).

The need to shut down the offices using a shared installation is even worse, e.g. Imagine having a shared installation used by a thousand users, before something can be deployed into this shared installation, it has to be ensured that every running office started out of this installation has been terminated. Otherwise this office instances may work with inconsistent data, which can lead to data loss or crashes.

Different content types are deployed differently. Thus the problems that occur upon insertion/removal of a package at runtime are different.

4.1.4.6 Scripting Framework

(Noel Power, April 2003)

Currently Scripting support for SO/OOo exists for one language only, SO/OOo Basic. Users, Developers and ISVs who wish to extend SO/OOo using scripting are locked into this single offering. It is advantageous from strategic point of view to support Java and Java derivatives such as JavaScript and BeanShell for scripting. Other scripting languages such as Python could also be added as the market dictates.

A framework is needed to allow new scripting languages to be added and supported. It should be possible to easily plug in additional languages runtimes. The framework needs to provide an environment to enable users to easily and quickly develop, deploy and execute scripts. Scripting should be easy. A script developer should be able to write and test a script with a minimal set of steps. They should be able to experiment and learn within the scripting environment, enabling them to accomplish their scripting tasks easily and quickly.

Scripting in SO/OOo will be made more easy, intuitive and low cost. This is mainly due to the complexity and low level nature of the current SO/OOo API. In order for scripting to be be easy, intuitive and low cost, the SO/OOo API needs to be simplified. Such a simplified API should be well documented, script focused and high-level. This would allow users to intuitively discover the necessary objects and interfaces needed to program a specific task rather than having to work through a mountain of documentation.

4.1.5 Integration into the Gnome Desktop, and Microsoft Windows

(Christof Pintaske, June 2003)

(Requirements for integration into Microsoft Windows need to be determined later)

4.1.5.1 Look and Feel

“Geordi” emulates the color-scheme and basic font setting of a running Gnome-2.x desktop. "Q" is required to pickup the style of all basic widget elements (buttons, slider, fixed-text, check-boxes and so on) in a way that exactly reflects the current GTK widget theme. The majority of dialogs should not show any immediately visible differences to a dialog implemented in the GTK widget set. Focus, modality, and guidelines for widget placement will not be adapted.

"Geordi" uses a proprietary engine for installing, naming, and selecting fonts. Font names on the system may be invalid in SO/OOo, or denote to a completely different font and vice versa. The "Q" font handling scheme has to be consistent with the Desktop and the major applications. Naming of fonts shipping with "Geordi" or older versions of SO/OOo must not change. Explicit settings for font replacement must remain intact as well.

This will be addressed by using the systems font configuration (Fontconfig2). Doing so will be as well beneficial for the startup performance since "Q" will not need to examine fonts at startup.

"Geordi" uses a mixture of X11/cursorfont mouse pointer, pointer specifically designed for SO/OOo, and pointer that are modeled similar to those found on other platforms, like Microsoft Windows. This is inconsistent with other applications on the Gnome desktop and it inhibits proper pointer settings for accessible desktop themes (large cursor settings, high contrast themes, ...). "Q" should use Gnome compliant cursors where appropriate.

Hotkey/Accelerator scheme should be aligned with the Gnome desktop. Common tasks should

reflect in equal keys. Keys already taken by the window manager should not be offered or used for application bindings.

It would be nice to have Gnome input methods available in SO/OOo as well (currently only available in Gnome-edit and Gnome-terminal).

4.1.5.2 Printing

"Q" is required to be able to provide the same printing experience as "Geordi" without the need of neither own printer configuration functionality nor own printer configuration data on a Gnome Desktop system. It may eventually provide an own UI to the system configuration if the systems UI is insufficient or otherwise unusable.

The CUPS system encapsulates and provides access to locally connected printer, and remote printer accessed through the IPP, LPD, SMB and many other protocols.

CUPS provides a legacy layer that enables "Geordi" to chose from and to print to all printers configured within CUPS. However "Q" will be enhanced to query printer capabilities from CUPS and comply with CUPS configuration settings, thus making the proprietary SO/OOo printer administration obsolete in favor of the according desktops system solution.

Currently there is no system-wide print dialog available. If there'll be shareable UI available in "Q" timeframe it should be integrated instead of implementing a proprietary UI.

4.1.5.3 Integration of network neighborhood

"Q" is required to be able to load and store files on SMB shares to allow for interoperability in a network of Microsoft Windows computers. The UI has to be extended to give convenient access to this functionality. Further on it is desired to be able to handle all URLs that are valid in the Gnome Desktop file manager (Nautilus)

Gnome provides a virtual file system service (GnomeVFS). It makes various file systems transparent to the user. Amongst others it is able to cope with protocols like HTTP, FTP, NFS, SMB, local files, WebDAV. Further on it provides MIME information and maintain an according application registry.

"Q" will be enabled to use GnomeVFS. The main benefit is to improve interoperability with Windows SMB shares. GnomeVFS will as well improve interoperability with the Nautilus file manager since contents will be available in both applications through the same URI. Further on SO/OOo will be able to handle external data conforming to the system MIME settings.

4.1.5.4 Desktop integration

A user of Microsoft Windows is able to create new, empty SO/OOo documents on the desktop. The Gnome Desktop shall offer a competitive feature. This will be addressed by creating an according Bonobo component for Nautilus.

The most important applications on the Gnome desktop will be immediately available from the Gnome start menu. An according top level entry for SO/OOo needs be added.

It is required to inform all application on the Gnome desktop about how to handle SO/OOo files. Especially the browser and the file manager need to be able handle them appropriately and to show an appropriate icon. This will be addressed by registering the SO/OOo MIME types in the according system registry.

4.1.5.5 Application interoperability

Available browser plugins will be made available for the SO/OOo “Insert -> Object -> {Plugin|

Sound|Video} mechanism.

The Ximian Evolution address book will be imported into SO/OOo.

4.1.5.6 Installation

The Gnome Desktop system will be delivered as a set of packages in the RPM[1] format. It's necessary to seamlessly deploy and update SO/OOo together with other parts of the operating system.

Currently most Linux distributions provide RPMs that correspond to a complete "Geordi" network installation. "Q" will be delivered as a more fine grained set of packages that enable individual installation, update and patching of functional entities. Shareable files for system integration will be installed in the system and not in every users home directory as in "Geordi"

4.1.5.7 Configuration

SO/OOo makes use of a set of helper applications. Application that can be used as helper application in SO/OOo need to useable without further administration. "Q" will make use of the Gnome desktop configuration where possible. Examples here are default mailer, default web browser, and proxy settings. The SO/OOo Configuration component will provide a transparent access to the system settings, which will be resolved via a plugable backend architecture.

4.2 General Improvements

4.2.1 Stability and Quality

(Matthias Huetsch, June 2003)

4.2.1.1 Document Recovery

Any potential crash of SO/OOo that could result in lost data will be addressed by defaulting to a new Auto Recovery mechanism performing a regular backup of documents and settings. This Auto Recovery mechanism will restore a users work environment to the state before SO/OOo terminated.

The Document Recovery dialog will be enhanced to guide a user through this Auto Recovery process of restoring documents and to provide detailed feedback upon the recovery progress.

4.2.1.2 Error Reporter

Stability improvements for SO/OOo are required to be measurable. In order to be able to characterize the current stability level, this requirement was pulled-in to the SO/OOo “Geordi” release and has been addressed by an Error Reporter application, recording crash data and user feedback about document loss or preservation.

This Error Reporter will help to identify and eliminate crash causes, as well as provide a metric to show improvements in the stability levels achieved for SO/OOo “Q” and subsequent releases.

4.2.2 Administration and Deployment

(Dirk Grobler, Christof Pintaske, May 2003)

4.2.2.1 Deployment

The installation paradigm changes from being primarily single user oriented to being administrator

1 Http://www.rpm.org

oriented. It is looking for large scale installations and administrations. This issue will be addressed by conforming the installation process to the respective systems installer services, since these services already provide or enable an existing installation tooling.

The "Geordi" user-installation as a required step of a network-installation will cease to exist. The office applications will be fully functional from the first start. Gathering user specific data will be postponed to the first application start or queried directly from the system. Document and setting defaults will be generated on the fly during the first start or on demand.

The repair mode of the setup will be dropped. Repairing the installed files is already handled by the native package mechanism (usually by a reinstall). This does not affect user driven changes since user and system files are strictly separated.

Upgrading and patching will be performed by means of the respective system installer.

Since Solaris and Linux do not necessarily provide a GUI for the installation we will provide a graphical front-end to guide the user through the installation process of the packages. Nevertheless the respective packages can be installed completely independent of this front-end.

4.2.2.2 SO/OOo Configuration

Improvements for the handling of configuration data are planned that effect the whole application. These are additions to the existing configuration technology:

- Reset of configuration settings within the Options Dialog

  Currently, it is not possible for a user to return to his initial settings provided during the installation process. This gap will be addressed with the introduction of 'reset to default' functionality in the SO/OOo Options Dialog. A user is able to select a specific settings page within the options dialog and can press the reset button and the respective settings will be reset to their installation default. Quitting the dialog with the OK button will make the settings persistent.

- Protection of configuration settings.

  Administrators have the possibility to protect configuration settings against changes by the user, so that the settings are effectively read-only. "Geordi" reflects the read-only status only for a few selected settings in the option dialog. Settings not addressed could still be overwritten by the user and applied for the lifetime of the SO/OOo session. For the "Q" release this gap is closed and all read-only settings are reflected as such in the Options dialog.

- Notification handling

  Changes of configuration settings are currently very seldom reflected to the user. Only a few components within SO/OOo are able to listen and apply notifications of configuration settings. In some specific deployment scenarios, like SunRay deployments, it is very common that applications are long running.

  Therefore it is important that SO/OOo is able to react and apply configuration changes, when they occur instead of requiring a restart. "Q" addresses the misbehavior by introducing a consistent configuration notification handling.

4.2.3 Modern Look & Feel

(Christian Jansen, May 2003)

Since we have started to improve the look and feel in "Geordi", we will continue this work with "Q". This chapter gives an overview of what will be done to get a more up-to-date user interface which provides more satisfaction to work with.

4.2.3.1 Toolbars

The look and feel of toolbars will be improved by introducing a completely renewed technology for toolbar handling. They will get a modern, intuitive look and will be improved for usability. For example, the handling of dragging a toolbar will be improved by adding an area which indicates that it is possible to drag a toolbar. Toolbars will also be improved by adding more feedback during drag and drop interactions. This means for example that toolbars will be “magnetic,” so that they jump automatically to their correct position. Another important feature is that toolbars in "Q" will not use the whole application width anymore when they are docked. From the visual appearance toolbars will get a more flattened look.

4.2.3.2 Widget style

Another very important area to improve is to renew the look of widgets (widgets in graphical user interfaces, are a combination of a graphic symbol and some program code to perform a specific function. E.g. a scroll-bar or button.) Depending on the level of system integration, SO/OOo will use the widget style of the Desktop. If this is not possible for "Q" the current widget style will be improved in a way that SO/OOo looks attractive enough in environments like Gnome 2.x /3.x or Microsoft Windows Longhorn (the successor of Microsoft Windows XP).

4.2.3.3 Tool Pane

With SO/OOo "Q" a “new” type of dialog will be introduced. We call this dialog Tool Pane, since it is related to tool windows, e.g. the Stylist. It works like modeless dialogs do. The Tool Pane by default will be vertically docked and positioned to the left or the right of the document. Tool Panes will be introduced in SO/OOo Impress first, because SO/OOo Impress is an application where turning to more modeless user interface makes most sense.

Tool Panes can contain for example a list of predefined presentation layouts which can easily be applied to the presentation by dragging a thumbnail of a layout onto the slide. To give the Tool Panes a modern look they will come in a web-like style. Tool Panes will have less complex layouts than “standard” dialogs, this assures a much easier handling and saves time when creating a presentation.

4.2.3.4 Dialog Design In General

SO/OOo "Q" dialogs will be improved in the following areas:

- Disabled items will not have a 3D style, a disabled state will only be grayed out.
- Tool Windows, like the color palette, will be modernized by removing the old fashioned 3D-borders.
- Selections, like they are for example displayed in the Gallery, will be replaced by a more state-of-the-art look.

4.2.3.5 AutoPilots (Wizards)

Currently, many AutoPilots in "Geordi" are designed differently from each other. Some of them are more than five years old and never changed in their appearance. For SO/OOo "Q" the look of the AutoPilots will be unified to a design which is orientated towards a typical Wizard design used by Mozilla or other applications like an Installer. Having a consistent look over all AutoPilots within SO/OOo improves usability and provides a higher overall satisfaction because users will come faster to a result. The renewed AutoPilots will also come with a minimum of technical expressions, users will be assisted through the steps by adding short notes or explanations into the dialog.

4.2.3.6 System Integration

Being a good citizen in a desktop environment is a very important topic. The SO/OOo system integration will be improved for Gnome based Desktops. System integration for the look and feel means that typical desktop behaviors or looks have to be also used in SO/OOo e.g., the mouse pointer has to look in SO/OOo the same as on the desktop.

A good system integration includes also a set of document and application icons which fit the style of the desktop, on that SO/OOo runs. For "Q" we will have icon sets, that will take care of the typical style of the desktop environments.

4.2.4 Integration (Plug-ins, 3rd party applications)

(Mathias Bauer, May 2003)

The main subject of this chapter is about linking and embedding data in or from SO/OOo. To better understand this, here is a short description of the current functionality and the history of its development.

In the past SO/OOo was made up by single applications that supported the OLE compound document architecture and so were able to embed and link objects from other applications that also support this architecture. The same technology was used to embed objects from the different SO/OOo applications into each other.

The biggest drawback of this approach was that it worked only on Microsoft Windows. SO/OOo versions on other platforms were only able to display a cached representation of the embedded object (usually a GDI metafile).

When SO/OOo was redesigned to follow a platform independent approach for its GUI, the support for OLE was removed with the exception of the ability to embed 3rd party objects into SO/OOo and activate them in an external window on Microsoft Windows (so called “Outplace activation”). The more advanced “Inplace activation” of 3rd party objects (editing embedded objects in an own window inside the embedding container document) was abandoned and also the ability to embed SO/OOo objects into 3rd party applications was completely removed. Another dropped feature was the ability to link to external files instead of embedding them.

The internal embedding functionality (embedding SO/OOo documents into each other) still used the old code formerly written to wrap the OLE interfaces, but most of the OLE related parts of the implementation were removed. Because we didn't have any replacement for OLE, all SO/OOo modules had to be put into one single application, otherwise we would lose the ability to activate SO/OOo “Inplace” even in our own applications.

Meanwhile, customers responded with several suggestions for improvement. These suggestions included the following:

- Inplace Activation is seen as indispensable functionality of modern applications on Microsoft Windows.
- The inability to embed SO/OOo objects into other applications is a strong competitive disadvantage against the competing applications if the customer uses such application techniques in his own IT solutions.

4.2.4.1 Support for OLE

The following concept will show how SO/OOo can get the needed functionality and how we can develop this in a multi step procedure.

SO/OOo can embed its own formats on all platforms and also OLE objects from other applications

on Microsoft Windows. "Geordi" adds the new feature that SO/OOo objects can be embedded via OLE into other applications' documents on Microsoft Windows. In this case and also in the case of 3rd party objects embedded in SO, activation of the object is only possible “Outplace”, no “Inplace” activation is possible.

“Q” will allow “Inplace” activation in both cases and also add the currently missing feature of linked objects.

Besides that the whole embedding functionality and API (currently C++, in future UNO based) will be redesigned to support arbitrary embedding techniques, objects and persistence models. The switch to an API solution will also drastically improve the maintainability, stability and extendibility of the code.

Some examples for features that will be enabled by these changes:

- It might be possible to add support for comparable embedding and activation procedures from other platforms like Bonobo on Gnome (if there is a demand for that). This can be done without changing any existing code, only some new services have to be implemented that use the same interfaces as the services that are used to support OLE. Because the OLE and Bonobo interfaces are quite similar, this seems to be a reachable goal for later steps.
- Embedding, persistence and activation will be treated as orthogonal functionality, that means embedded objects are not forced to support persistence or activation. So we can have embedded and activatable objects without own persistence data (“lightweight” objects) like embedded charts that get their data from an embedding container. This is a feature that makes our own charts more usable and more comparable to state-of-the-art solutions.

The enhanced embedding capabilities also improve the interoperability of Microsoft Office with StarOffice on the file level. In StarOffice 6.0 embedded SO/OOo objects in exported Microsoft Office documents are stored in Microsoft Office file format or in our old binary format. The first alternative has the drawback that the object must be exported and imported back when loaded again and this export/import cycle may cause some data loss. The latter alternative has the drawback that there is no way to activate the embedded object in the document if it is loaded into the Microsoft Office application, and of course the old binary format is also at risk to cause data loss, e.g. we will lose all CJK attributes).

Since "Geordi" those objects can be stored as full featured OLE objects using our XML file format, with full support for activation in a Microsoft Office container without any possible data loss. “Q” will add the Inplace Activation of these objects in the Microsoft Office container and so enhance the user experience and interoperability.

4.2.4.2 Browser Integration

A first implementation of a Mozilla/Netscape plug-in for Windows already exists, along with ActiveX control for Microsoft Internet Explorer, and a signed Java applet. All these objects when embedded into a browser use a local SO/OOo installation to view SO/OOo documents inside a browser.

In “Q” the plug-in will be completed and extended to other platforms. All objects will get some enhancements and optimizations, because even the ActiveX control and the Java applet suffer from some usability problems.

We will enable editing functionality for documents loaded by the embedded browser objects.

4.2.4.3 Integration of 3rd party UNO components

In "Geordi" we developed the very powerful Addon concept that already enables customers to use

our ODK to develop components that can be easily integrated into SO/OOo and the SO/OOo GUI. Especially the GUI integration will be enhanced in “Q” together with the changes in the application framework GUI code.

4.2.4.4 Integration with mail clients

Will be determined later.

4.2.4.5 Integration of media files into SO/OOo

Currently SO/OOo can only use browser plug-ins to play videos in e.g. a presentation, and unfortunately these plugins do not work in SO/OOo on Linux or Solaris. So a different and more powerful way to work with media files is needed. Possibly this can be combined with the new embedding API mentioned in the section about OLE.

4.2.4.6 Support for filter development

SO/OOo already has a concept for the development of filter components. In the past very often the necessary configuration files for filters caused some problems, so the filter configuration will be changed for a better manageability. Additionally all internal filters will be converted to real filter components to make it easy to replace them.

4.3 Upcoming Opportunities

4.3.1 Asian and CTL languages

(Falko Tesch, July 2003)

4.3.1.1 CJK Requirements

Since SO/OOo is already CJK enabled and has a lot of CJK-specialized features, the "Q" release aims more on a sophistication and refinement of existing features.

This applies to all applications such as Calc, Writer and Impress/Draw.

Improvements are to be done in the following areas:

Text Grid

For Asian users the text grid is a valuable help to learn writing in school as well as a traditional way of writing in good manner and style.

SO/OOo text grid abilities will be enhanced by more versatile printing options, more default templates to choose from and additional adjustments that can be applied on the behavior of Western text within text grid.

Doing so will put SO/OOo in line or ahead of our competitors and enhancing our text grid it is a valuable issue gathering more acceptance of traditional users.

Enhanced IME Support

The IME is the most crucial instrument for all East-Asian users. It enables them to input their characters. While Sun Solaris provide standard IMEs, the Linux community provides various ones.

Customers have requested that SO/OOo support as many IMEs as possible. It will help SO/OOo to

consolidate its position within the Asian Linux world as it is already leading in the Western world.

Furthermore we will have a more sophisticated read-out of IMEs. Since the IMEs that come with Microsoft Windows are all using an extensive unified interface (contrary to Linux for example) users are used to certain amenities, such as automatic language detection while typing. Unfortunately the variety of different IMEs throughout Microsoft Windows, Sun Solaris and Linux brought us to choose the smallest common denominator. This will change.

AutoCompletion et al

“Q” will support AutoCompletion and AutoCorrection for all East-Asian languages.

4.3.1.2 CTL Requirements

Numbering and Bullets

In Arabic and Hebrew regions people write right-to-left (RTL). Therefore also they use numberings and bullets on the right side with right alignment.

Since “Geordi” takes care of basic features of aligning bullets, "Q" will give more intelligent automatism like in well-known products on these markets.

Also a bigger variety on localized numbering symbols will attract more users.

Special Formatting

Like East-Asians also CTL users are accustomed to various ways of formatting their text. These are mostly minor issues but having them will give SO/OOo a more complete and respectable estimate.

Also addressing certain features not only to CTL users in general but to address them to the specific user group (like Arabic) will enhance it even more.

RTL-UI

While a lot of Arabic and Hebrew people are used to work in an English left-to-right (LTR) environment to accomplish their (RTL) work, they highly value if a product is also available in their local (RTL) flavor. Even if a RTL-UI is not perfect and the users might switch back to an English LTR-UI the RTL-UI can often be considered the “door-opener” to these people.

“Geordi” already provides a framework for RTL-UI. Early adopters such as Tk Open Systems in Israel have brought up first promising prototypes. "Q" will be a major leap forward to give those people tools at hand to generate a full RTL-UI enabled office for the Arabic and Hebrew market.

4.3.1.3 Generic I18N Requirements

Documentation and UI Style

Most users in East Asia and in CTL countries are used to an existing set of terms and nomenclature. SO/OOo will use this to ease transfer and learning for the users. It will also familiarize them quicker with our product.

Special attention has to be given on possible cultural issues and clashes with UI terms as well as with the localized documentation (see the PR China - Taiwan example in the Overview section).

To accomplish this goal “Q” will provide the user with a good and easy-to-read online help as well as printed documentation in their native language.

Another important issue is technical documentation that will give engineers in those countries the feeling of being taken serious and valued (even if they feel more comfortable with the English originals after all).

Impress and Calc Issues

While there is a UI and there is a document area there is also a canvas that resides “in between” the document view and the application UI.

Even in a non-RTL-UI RTL documents and RTL canvas are common to Arabic and Hebrew customers.

”Geordi” already takes this in account and automatically mirrors canvas tools such as rulers, scroll-bars etc.

In "Q" Impress/Draw and Calc will follow.

RTL users are have the habit to start reading from upper right to lower left. This means for Calc and Impress that the column and row headers have to mirror (in Calc) and the arrangement of slides templates in Impress have to be mirrored.

4.3.1.4 Tool Integration

Every country has its own special cases and rules. And not all can be addressed by a single product such as SO/OOo.

But when launching in those markets a build-in support for the most important tools and their vendors is advisable.

Examples:

- Multi-language multi-script countries like India (7 major languages and scripts) are used to use on-screen keyboards to visualize the various keyboard layouts.
- East-Asian countries like Korea that use Chinese characters (Hanja) in their everyday work need transliterations, dictionaries and special input methods.
- Arabic world that uses a very sophisticated style of Arabic to create and edit holy texts (Koran).

All these countries have one thing in common:

They all have local vendors that create, program and sell value-added software for office productivity suites.

Supporting these tools and/or giving developers of such companies documentation at hand to provide tools also for SO/OOo can raise the meter for acceptance considerably. (Again this shows how valuable a localized technical documentation is.)

4.3.2 Digital Signatures

(Michael Brauer, May 2003)

As said above, digital signatures themselves are a relatively new topic for office applications, but the requirement to protect data from being modified has always existed. In SO/OOo, like in other office suites, it has been addressed by documents and document pieces that could be protected from being edited, like text sections in Writer or tables in Calc. Sometimes the protection is a simple option. Sometimes the protection is combined with a password. In both cases, these features are nothing more than an edit protection built into SO/OOo. They do not add any real security to a document, because they cannot protect documents from being changed outside SO/OOo. This in

fact has become much easier with the introduction of the XML file formats, that even removed the little piece of security that the proprietary and undocumented binary file formats had.

Though the existing protection features don't offer security, they show where demand for and benefit of digital signatures seems to be highest. Therefore, adding digital signatures to SO/OOo in “Q” will start with adding digital signatures to the existing protection features. This not only makes these features secure, it also allows customers to use signatures with very little learning effort. Most important however is that customers this way get a solid set of digital signature features that they can use in their environment and in their workflows. To better support digital signature based workflows from the beginning, multiple independent signatures will be supported for one and the same document piece. Based on customer experience with these features, support for hierarchical signatures or other kinds of nested signatures might be added in later SO/OOo versions as well.

To actually save digital signatures in SO/OOo files, W3C's XML Signature recommendation will be used. It is not only a standard that very well fits into the strategy of reusing existing standards for SO/OOo's file format, but it also can be assumed that this standard will be supported widely in the near future.

SO/OOo's signature features for documents will make use of the same security framework that is used for other security features in SO/OOo, like for Basic macros. That is to say, there will be one key management and a common set of signature methods in SO/OOo.

In contrast to signatures, encryption is a feature that has been supported by office applications for a long time. In "Geordi", encryption is available for whole documents. Since there was no encryption standard available at the time the XML file format was introduced, encryption currently is done as a SO/OOo specific extension of the ZIP packages that contain the XML files. As soon as encryption based on the recently finished W3C XML Encryption recommendation becomes more widespread, that may be the case for “Q” or later, it will be supported by SO/OOo as well. It again will make use of SO/OOo's common security framework for key management and encryption methods.

Since the W3C Encryption specification does not only support encryption of whole documents but also of document parts, future SO/OOo versions might also support encryption for document parts only, like this will be the case for signatures in “Q” already.

4.3.3 Compelling upgrade features (e.g. multi media, OCR, DB)

To be determined later.

4.3.4 Web Services

(Kai Sommerfeld, May 2003)

4.3.4.1 SO/OOo Web-Services Client (WebServices-1)

Basically, from the current state, there are three approaches to cover Web Services partly or fully. The different approaches are:

A) Access Web Services via Java mechanisms from Java.

B) Use an invocation based approach.

C) Have a full featured UNO language binding

A: As UNO components for SO/OOo can already be written in Java, there is already the possibility to access web services. The usage of web services via Java is limited to Java and does not allow to export the Offices API as a web service.

B: A diploma thesis has been written to have more direct UNO support. With this UNO component,

it is possible to seamlessly access web services from Star Basic (please see http://udk.openoffice.org/java/examples/wsproxy/component_description.html for more details). The implementation of this component is not yet feature complete. Most Web Services can be accessed, but some, that use special features currently cannot be used. To use it for general purpose, it is necessary to extend the component to support this features. Being a UNO service, this component can be in principal used from any language supported by UNO, for instance SO/OOo Basic, C++ or Java. But access to web services is only for these languages seamless, which support the XInvocation and the XPropertySet for invoking methods respectively accessing properties. The only language currently doing this is SO/OOo Basic. For all other languages it is kind of nasty and inconvenient.

C: The most general and most expensive approach to web services support in SO/OOo respectively UNO is a web services language binding. This renders web services to be fully UNO accessible and allows the export of UNO objects as web services as well as the import of any web service. Web services are then directly programmable in any UNO supported language, without any additional burden.

4.3.4.2 SO/OOo UDDI Browser (WebServices-2)

There are several free UDDI browsers available from 3rd parties. A good example is a web based browser at http://www.soapclient.com/uddisearch.html. This browser includes a generic SOAP client, which supports even live testing of Web Services. UDDI browsers can be used to obtain the URL of the WSDL file for certain Web Services. The WSDL file can then be used to create a UNO component instance representing the Web Service, like described above. There seems to be no real need for a SO/OOo UDDI Browser, because the actual value-add is very small. The implementation effort would be medium to high.

Another approach would be the integration of Web Services into the coming Scripting Framework. Web Services could then seamlessly be offered via the macro selection dialog. Further investigation is needed, to clarify details.

4.3.4.3 SO/OOo Web Services Provider (not covered by requirements so far)

Web Services use SOAP for remote method invocation. A UNO-SOAP language binding could be implemented to offer every SO/OOo functionality that is available as a UNO service as a Web Service. See also “SO/OOo Web-Services Client”.

The effort to implement a UNO-SOAP language binding is very high (raw estimation: at least 3 months).

4.3.5 Intelligent Document Tags

(Andreas Martens, May 2003)

4.3.5.1 User

If an Intelligent Document Tag component is installed, the user is able to activate and deactivate it. If it's activated the recognized keywords are underlined. The mouse cursor changes its appearance if it is above such keyword. A context menu shows up if the user presses the right mouse button. The content of the menu are the possible actions for this keyword, typically the names of the dictionaries which contains the keyword. The user choose an action and the component reacts accordingly.

4.3.5.2 SO/OOo

SO/OOo has to separate the document content into sentences. These sentences will be passed to the component. If any sentence is changed by the user it has to be analyzed again.

SO/OOo controls the underlining and the mouse cursor appearance for keywords. If the user calls the context menu the SO/OOo has to ask the component for the menu entries. If the user chooses an entry the SO/OOo calls the component with the chosen entry (in form of a number, an ID).

4.3.5.3 Component

The component has to support three interfaces.

If a sentence has been passed the component has to deliver all start- and end-positions of keywords in this sentence.

If a sentence and a character position have been passed the component has to deliver the available actions as context menu entries and the corresponding IDs.

If the component is called with such an ID it has to react accordingly (see examples above).