Office Shell Ideas and Issues

The latest version of this document can always be found on \\design\public\chrisgr\offshell.doc. Please open it as READ ONLY.

Summary

This paper investigates a proposal that the next major version of Office after Chicago should consist of a Windows shell and applications optimized to work together. The proposal originated at a senior technical retreat at Hood Canal in June/93.

Recommendation: We should follow the "Aggressive" version of the plan outlined below.

Proposed Plan

- Bundle an enhanced Windows shell with the next major version of Office to ship after Chicago.
- The Office shell would be functionally a superset of the Chicago shell, designed for maximum synergy with Office..
- Enhancements to the shell could include minor modifications to the shell UI for optimal interaction with Office apps; increasing the extensibility of components such as the Explorer, the Desktop and the Tray; the provision of app-specific extensions to take advantage of them; and additional applets, file viewers, OLE servers and other tools.
- Apps in the Office shell release would include Excel 6, Word 7, PowerPoint 5 and Access 3.
- The Office shell would define the next standard Windows UI after Chicago. At an appropriate time after Office+Shell ships, the enhanced shell would become the next standard Windows shell for both Chicago and Cairo.

Schedule

Q2/94	- **Chicago ships** - Shell has limited extensibility. (See below for details.)
Chicago + 6 months	- **Office ships with optimized shell.** - Shell adds features for optimal support of Office requirements. (See below for details) - Office includes many features that exploit the new shell. - New shell not initially available with Windows itself - The Office shell should be approximately a superset of Chicago shell features (although some components, such as the Tray, may be replaced.) - Note that the Office shell date may not be strictly dependent on when Chicago ships. If the Office shell used a different code base, then a slip of Chicago could reduce the delta to less than 6 months.
Sometime after Office ships	- **Cairo ships with a shell that is a superset of the Office shell** - Excludes any components that we choose to keep only for Microsoft Apps. - Extended to use special features of Cairo
When Cairo ships	- **Enhanced shell added to Chicago** - An alternative would be to add the Office shell back into Chicago when Office ships. This should still give Microsoft Apps a significant development lead.

Pros

- Chicago team can concentrate on shipping within their memory targets and schedule because they would have to add less OLE support, and would not have to provide as much extensibility.
- Office gets a shell optimized for its use.
- Office gets a big jump on competitors in creating apps optimized for the new shell.
- Since the new shell is bundled with Office, we don't have to assume that it needs to run on Win 3.1. (Issue: Actually, this would require bundling all of Chicago with Office.)
- Assuming the Office shell is upward compatible to the Cairo shell, then Office apps will be automatically much more optimized for Cairo.
- Simplifies the cross-group interaction necessary to produce synergistic versions of apps and the shell.

Cons

- Risk of ISV retaliation.
- Negative impact on corporate image.
- Would probably delay release of Excel 6, Word 7 and other Office apps to do work necessary to leverage shell. This would probably mean we would not get Chicago-optimized releases within 3 months of when Chicago ships, as originally planned.
- Might require some extra work by Chicago to provide enhancements or hooks needed for eventual use by the Office shell. (We don't want to have to ship new versions of GDI and User in the Office time frame.)
- Increases the pressure to sim ship major apps, and adds the shell as another component to sim ship.

Product Vision

There are two possible plans we might follow:

1) Conservative plan: We develop enhancements to the shell and modifications to apps that are relatively well understood, and don't change current designs too much. The emphasis would be on creating an Office shell that has considerably higher value added than the shell in Chicago, both by limiting what we provide for free in Chicago, and by adding features in the Office shell. We would also add features to applications to leverage the currently planned shell features.

   Advantages:

   - This plan has less impact on current designs and schedules. For example, we originally wanted minor upgrades of major apps to ship as soon as possible after Chicago to optimize them for Chicago, and to showcase Chicago features.

   Disadvantages:

   - We may not be taking full advantage of this opportunity.
   - Assuming we do intend to eventually do the changes described for the aggressive plan, it would come later, and might have to be done in parallel with the Office shell work.

2) Aggressive plan: We use this opportunity to bring about a major improvement to the model of how users interact with the shell and applications. This could include changes as large as switching

apps to SDI, and the necessary changes to the shell to optimize it as an environment for SDI or document-centric apps, and to make progress on the problem of factoring functionality between apps and the system.

Advantages:

- We could gain a much bigger advantage from the Office shell. We coudld pull off the "UI Paridigm Shift" to document centricity possibly two years sooner than if we did not folow this plan. Major breakthroughs in app usability may be possible. This would give us a very significant lead over out competitors, and make our competitors' products look "old".

Disadvantages:

- It would certainly take longer to ship the Office shell and related apps because the design issues are less well understood and the development work would be greater.
- It could delay the minor Chicago-optimized releases of apps. We could still ship minor app upgrades soon after Chicago. However this may cause too many upgrades too close together. This would dilute design, development and testing resources, and could delay the release of the Office shell. We would have to resist the temptation to add too many features to these minor releases.
- Implications for Mac core-code/core-doc strategy are not well understood. The aggressive plan would cause us to confront these issues sooner.
- Implications for the ability to run on Win 3.1 are not well understood. We probably could produce a version that would install and run in a limited way on Win 3.1, but it would take more work.
- In the past, people have assumed that developing next-generation apps ("Cairo apps") should include major architectural changes in addition to user model changes: However, the proposed aggressive plan puts more emphasis on the user model, although it does include some less extensive architectural work such as enhancements to OLE, improved OLE support, and enhanced programmability. Deeper architectural changes, as appropriate, would come in subsequent versions.

The following is a list of possible features in the Chicago shell, the Office shell and the Cairo shell. These specific features are largely orthogonal to whether we pursue the conservative vs. aggressive plans described above.

Chicago Shell Includes

- Most of the features currently planned for Chicago, including:
    - Combined program manager and file manager
    - New visuals
    - Context menus, drag/drop, NDD, etc.
    - Interoperability enabling. i.e. Supports drag/drop compatible with OLE
    - OLE 2.0
    - Simple Idispatch enabling of shell and applets. (So Excel 5 VBA can get the benefit of being the best language that can program the shell.)
    - Probably supports extensibility of document property sets and commands.
    - Assuming there is a "simple shell", it is upward compatible to the Office shell.
    - If there is a tray, it is not extensible, and not replaceable
- But not including:

- Extensibility e.g. Explorer not extensible (Capone hard coded into explorer)
- Vbasic
- Full-featured document viewing. Maybe only allow viewing thumbnails with the shell. A full set of document viewers would only ship with the Office shell.
- Some of the features of Mail. Can we limit the feature set of the Mail that is included with the system, to leave more value-add for Office?
- Other changes, TBD, to shell for optimal interaction with Office apps.

Office Shell Adds

(I assume that only some of these things could be done in the time available.)

- Moving apps to SDI. I'm optimistic that we could make SDI work very well given the opportunity to design apps and the shell together to make the shell an optimal environment for SDI app windows to reside.

  (Note that doing SDI would require following the "aggressive plan" described above.)
- VBA, including ability to automate cross-app scenarios that include the shell.
- Explorer extensions to browse into app document types: OLE Objects in Docfiles, Excel workbooks, Clipart files, etc.
- New tray designed for maximum benefit to cross-app requirements of Office
- OLE-based workbook
- OLE extensible Explorer
- OLE extensible desktop
- OLE extensible tray
- LoisO's document library as a low end document library solution for Chicago. Would be supported on desktop and in File Open, etc. Cairo doc mgmt should be upward compatible.
- Enhanced commdlg.dll, and commdlg code sharing with apps in shell/office bundle
- Investigate feasibility of adding multiple, switchable desktops
- Useful objects that could be placed on the OLE-container enabled desktop:
  - Information displays such as Post It Notes, data fields, tables.
  - Controls like buttons or sliders, that could activate VBA scripts.
  - Graphical indicators like warning or status lights, gauges, or even charts.
  - Special purpose information containers such as "document piles", "parts bins", etc.
  - Communication devices or devices that interact with the "Microsoft At Work" office
  - Decorations, such as clip art, pictures of one's family, etc.
- Enhanced mail: Add back what we took out of Chicago mail. Also add features for synergy with Office apps.
- The Office shell would be the target platform for Ren.
- Can some support for smart folders and/or project folders be added at this point? VBA programming of smart folders.
- Toolbar code sharing with apps in shell/office bundle?

Cairo Shell Adds

- Query based Explorer into OFS and summary catalogs
- Smart folders
- Project folders
- Other features necessary to work with OFS/DFS, security, etc.
- Infobooks

Assumptions

- The Office shell would start with the Cairo shell code base, but would be subsetted and adapted to run on Chicago, and shipped in time for Office.
- The office "shell infrastructure" would still be developed and productized by Systems. However, the Integrated Office group would in parallel develop extensions. The Systems base code and the Office extensions would ship simultaneously and appear as a seamless part Office. Some of the app extensions might eventually become part of Cairo and Chicago 2.
- Since Chicago shell does not need all the bells and whistles, it should now be easier to meet its memory goals and schedule.
- We will be able to make OLE fast enough, and reduce the working set enough to support the desired scenarios.
- We would have a little more time to design apps synergy features into Office shell
- Changing apps to SDI would be more feasible because of the opportunity to optimize the shell itself as the working environment for Office.
- "Integrated Office 1" would be redefined as Office Shell + Office Apps.
- Participants in the Office ISV program would be brought into the plan soon enough to announce support when the Office shell ships.
- Ren would probably require the advanced shell since it relies on Explorer extensibility.

Issues

- Need to determine ASAP any features needed in Chicago to support enhanced shell. e.g. What to we need in USER to support planned features?
- Would need to ensure compatibility of enhanced shell with 3rd party apps.
- What staffing would be required? How to organize?
- Keeping in sync with Chicago and Cairo versions. There's no way we can support three separate shell code bases. We'd need to divide the responsibilities clearly.
- Code base for Office shell? Probably the Cairo shell code.
- Do we also include the shell with the non-office versions of apps?
- If apps rely on shell extensions for important functionality, then to be cross-platform, we would have to duplicate these things on the Mac. For example, the Mac desktop isn't an OLE container.
- Can the Office, including the new shell require more than 4 meg of RAM? (I think the answer is probably yes, assuming the late 1994 time frame, but preferably basic functionality would still work in 4 meg.)
- Is the above schedule too tight? If so, is there a way we can scale back the plan, or stretch out the schedule?

- Are the Office apps of this generation only available as 32 bit?
- Does the Office shell use win32 OLE with LRPC as IPC?
- What kind of 16/32 interop work is required?