The Development of SAOImage DS9

Lessons learned from a small but successful software project

William Joye
Eric Mandel
Smithsonian Astrophysical Observatory
Cambridge, MA 02138
SAOImage DS9
Lessons Learned

Introduction
Observations on scientific software development
The Design Cycle
Where do we spend our time?
Why are we successful?
What works?
What does not work so well?
Future Challenges
SAOImage DS9

DS9 is a FITS visualization application

5 years since our first public release
35,000 copies of binaries and source code downloaded by 10,000 sites
Supported on 9 hardware platforms
Composed of 20 Tcl/Tk, C, C++ packages

Funding

Initial funding - NASA's Applied Information System Research Program
Current funding - Chandra X-ray Center and NASA's High Energy Astrophysics Science Archive Center.
SAOImage DS9

SAO

William Joye, Eric Mandel

Code Contributions

Dave Berry (Starlink), L. Brown (HEASARC), Mark Calabretta (ATNF), Mike Fitzpatrick (NOAO), Doug Mink (SAO), P. T. Wallace (Starlink), R. F. Warren-Smith (Starlink), Peter Wilson (HEASARC)

Collaborators and Beta testers

Steve Allan (UCO/Lick), De Clarke (UCO/Lick), Maureen Conroy (SAO), Uwe Lammers (ESA), Steve Murry (SAO), Mike Noble (MIT), Frank Primini (SAO), John Roll (SAL), Peter Teuben (UMD)
Observations on scientific software development

Software degrades with time

Requires constant maintenance to support existing features

Software chases hardware

“Today’s software is written for today’s hardware”

More bodies do not equal better results

Small teams working closely produce a better product in less time

Requirements are dynamic, not static

Users expand and redefine requirements over time
The Design Cycle

The design cycle is an iterative process

Microsoft Axiom - “It takes three times to get it right”

Prototype, prototype, prototype

Many intermediate releases

Development is driven by users needs

Work with, and listen to, users

They will define your requirements one step at a time
Where do we spend our time?

20% Answering email and supporting users
30% Updating and maintaining existing capabilities
30% Enhancing existing capabilities
20% Adding new functionality
Why are we successful?

Keep it simple

Graphic User Interface
Installation process

Do no harm when adding new capabilities

Only add new capabilities that benefit the community at large
Hide complexity that benefits only a few

Listen to customers

“People vote with their feet”
What works?

Continued support for technology we chose

Tcl/Tk and C/C++

Open software model

Package developers have total creative control
We benefit from the work of many others

Hardware improvement

In past 5 years, hardware performance has increased more than 400%
What does not work so well?

Growing Pains with work load and scheduling

Configuration and porting

Software Architect (GUI)

Support the primary software package (SAOtk)

Large amount of time spent updating and integrating packages

Some packages with minimal or no support

Never enough time for documentation
Future Challenges

Very Large Data Sets

<table>
<thead>
<tr>
<th>Current (per observation)</th>
<th>Future (per observation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOAO Mosaic 1 and 2</td>
<td>ESO Omegacam</td>
</tr>
<tr>
<td>Palomar QUEST</td>
<td>Lowel DCT</td>
</tr>
<tr>
<td>CFHT Megacam</td>
<td>LSST</td>
</tr>
<tr>
<td>SAO Megacam</td>
<td></td>
</tr>
<tr>
<td>Chandra Cas A Megasecc</td>
<td></td>
</tr>
<tr>
<td>~135Mb</td>
<td>~536Mb</td>
</tr>
<tr>
<td>~347Mb</td>
<td>~770Mb</td>
</tr>
<tr>
<td>~707Mb</td>
<td>~707Mb</td>
</tr>
<tr>
<td>~780Mb</td>
<td>~707Mb</td>
</tr>
<tr>
<td>~19Gb</td>
<td>~707Mb</td>
</tr>
</tbody>
</table>

Support for 64 bit Operating Systems

Multiprocessor support (threads)

“dual-core” processors
Summary

We spent large amount of time maintaining and updating existing capabilities

We try to anticipate future trends in hardware development

We utilize the open software model

Our users drive our development schedule