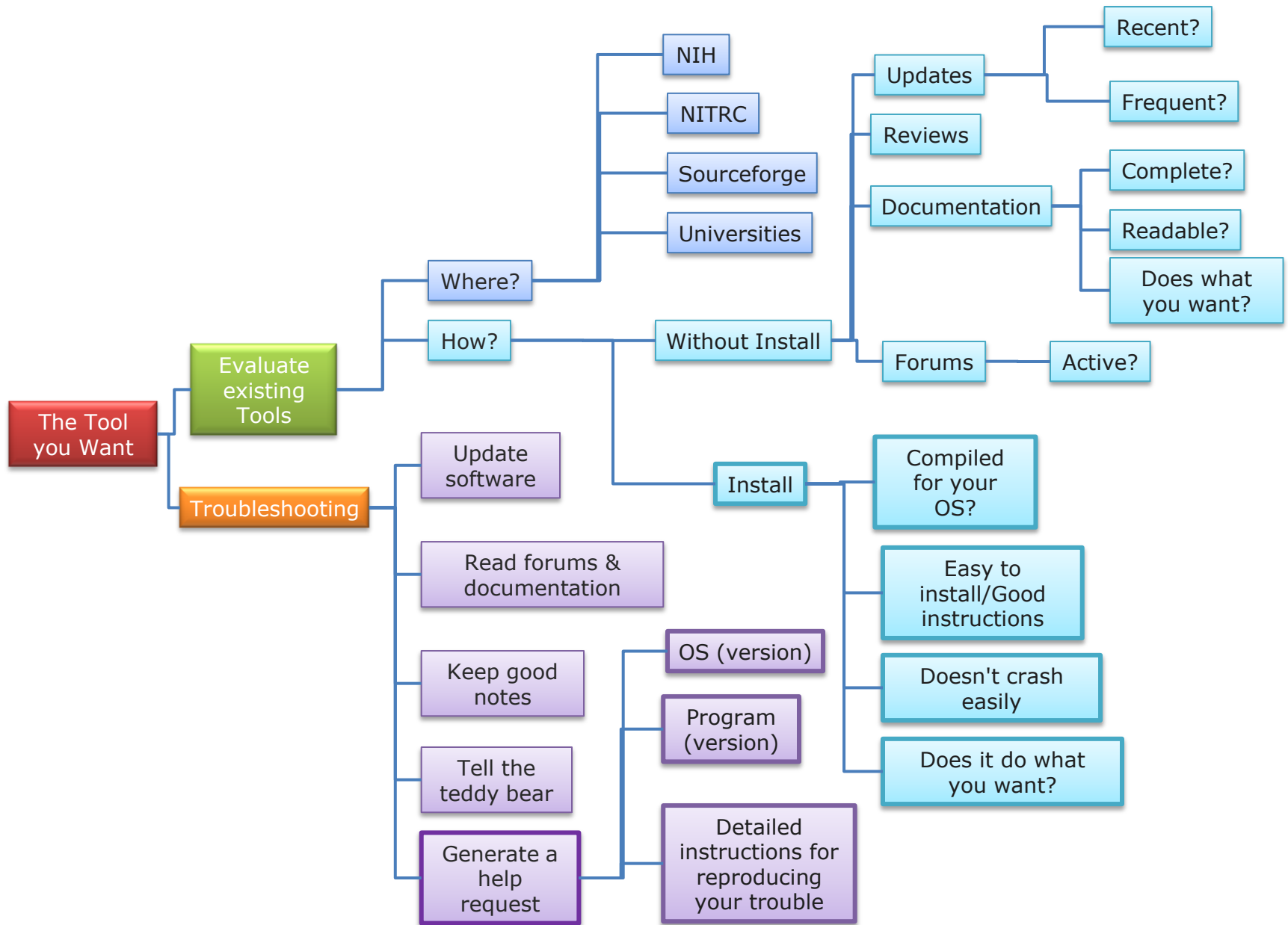
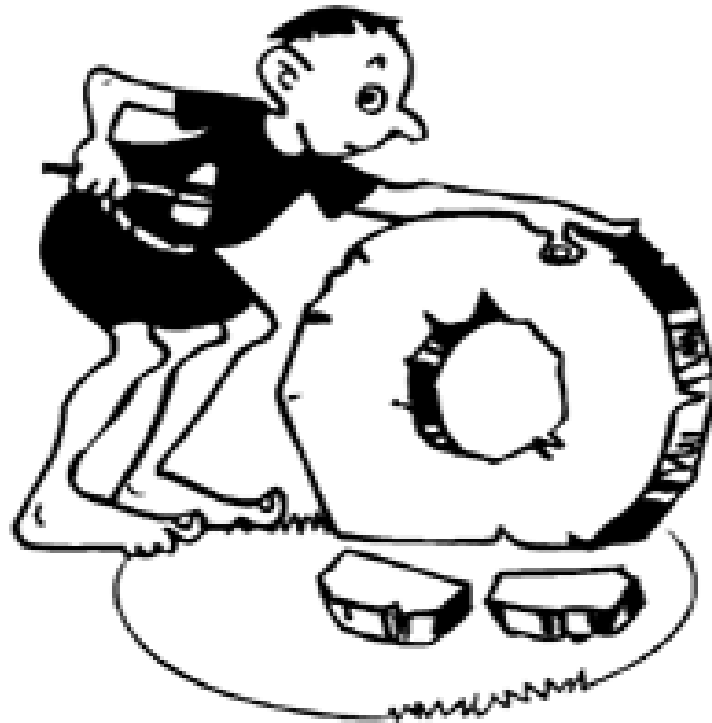


# Software Evaluation



Don't Reinvent the Wheel.  
(Does the software tool you want  
already exist?)



# What Do You Want?

What hardware is available for the software to run on?

What OS does it need to run on?

What does the software need to do?

- Can you split this functionality among multiple programs?
- Does it work with your data formats?

How much money can you spend?

# Where to Search

Expect to spend a couple of days or more googling for the tool of your dreams.

Several reliable sources exist:

- NITRC
- NIH
- Sourceforge
- Universities

# First pass Evaluation

NITRC organizes tools by type, posts summaries and user reviews of tools.

For any given tool, there should be:

- Reviews (Do people like it?)
- Update history (Is it active?)
- Documentation (Is it well written?)
- Support (Do users post? Do they get responses?)

If the tool does what you want and is documented and active, it is time to download it.

# Second Pass Evaluation

Install the tool and try it.

- Can you get compiled code?
- Will it run on your machine?
  - It may require Matlab, IDL, some special library
- Are the installation instructions correct? Well written?

Can you get your data in and out?

Does it crash?

Do you like it better than the alternatives?

# Bugs

An error that causes unexpected results.

Sometimes a characteristic of the program is intentional, although you don't expect it (a feature not a bug).

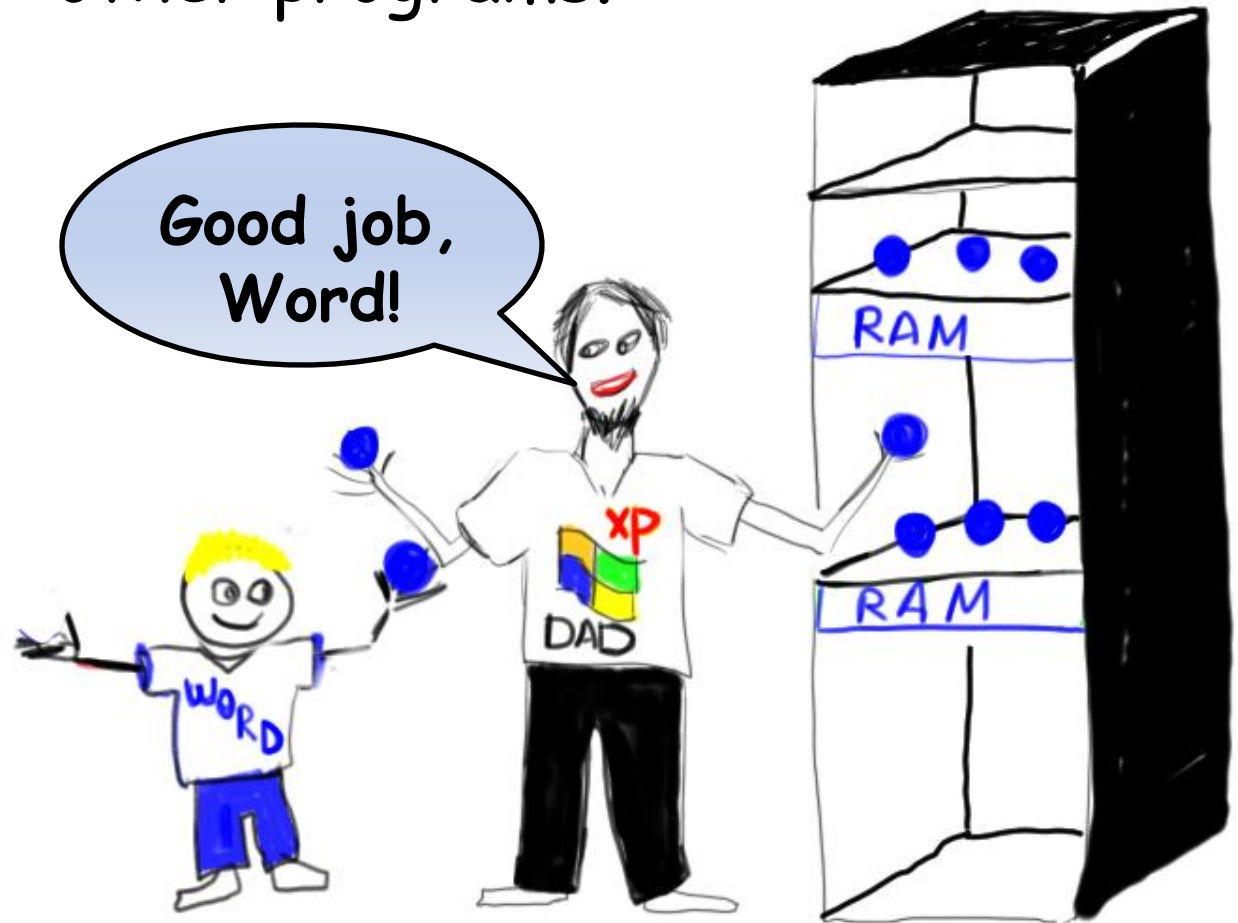
Some bugs are obvious, such as those that cause the program to crash or lock.

Other bugs, such as errors in the underlying math, are difficult to discover because everything seems to be working.



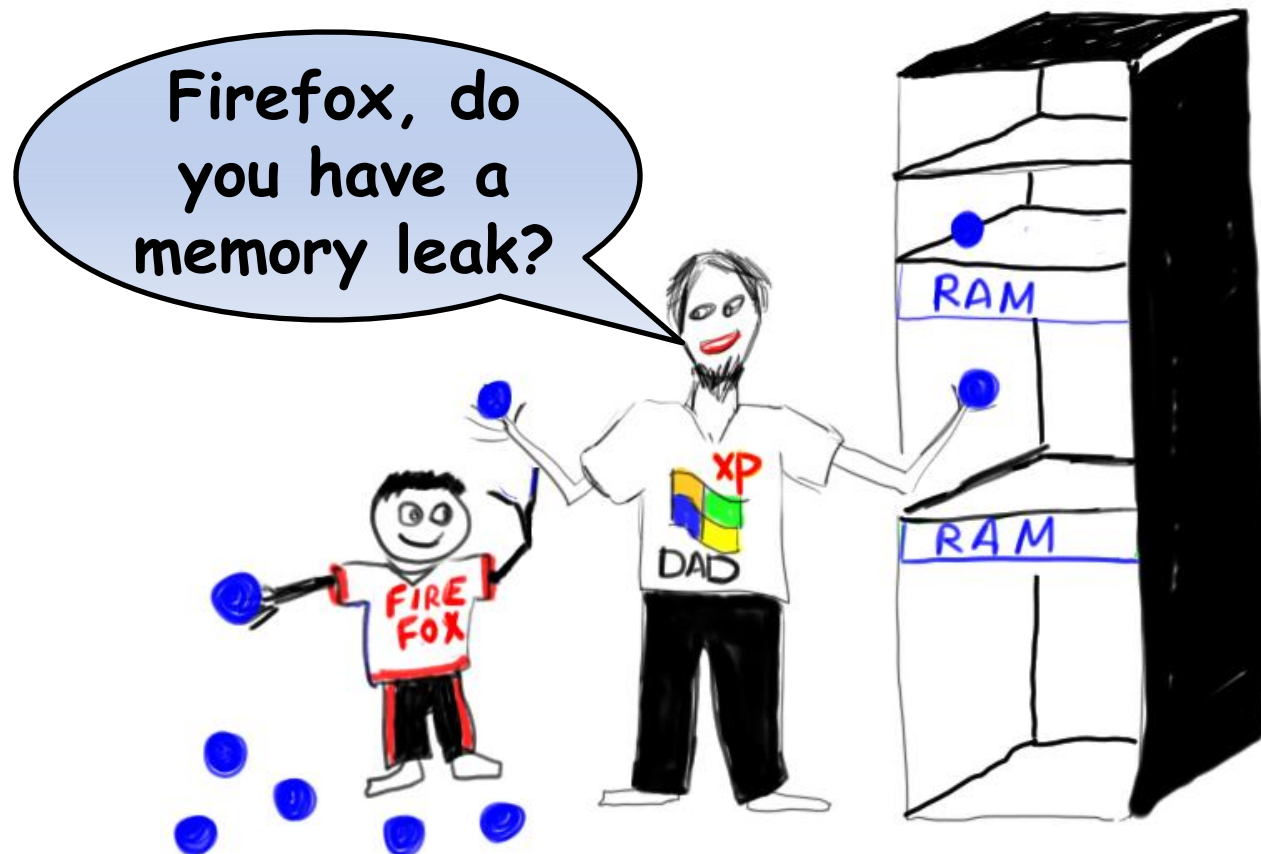
# Memory Leaks

Well behaved applications manage **RAM** they aren't using. That RAM is then available for other programs.



Applications that don't correctly manage their  
**RAM** have "**memory leaks**".

A memory leak slows or locks the system because  
no more **RAM** can be allocated to the programs.



# The Black Box

Neuroimagers want software to do complex mathematical tasks.

Most of us are not able to check the code to make sure we like the algorithms and that they are correctly implemented.

So, how can we be sure the software is right?

# The Black Box

Look for software that has

- Longevity
- An active development group
- An active user group
- Good documentation
- Good behavior (doesn't crash, functions actually produce something)

Software that has all or most of the above characteristics is less likely to do the wrong thing than "in-house" software or someone's dissertation project.

In short, if the software has

- withstood the test of time and
- heavy use by sophisticated users,
- a lot of bugs will have been worked out.



It is thus said to be mature  
and stable.



That doesn't mean it is  
perfect.




Watch the forums and do  
the updates.


# Troubleshooting

# Before Requesting Help

Reread the instructions and try them several times.




Get the most recent version of their software and confirm that you still get the problem.



Check Google




Look at online forums or FAQs that might answer your question.




Look at the documentation.

# Before Requesting Help


Is the software still supported  
(meaning, they are willing to spend time  
trying to help you)



Could the problem result from  
interaction with another running  
program?



Can you reproduce the problem on  
another machine?



You may be asking for a lot of time and  
effort, be sure you've made a thorough  
effort to solve the problem yourself.



# Talk to the Teddy Bear

Review the problem carefully, out loud and or in writing.

Can you think of anything you have not checked?

Try additional approaches and talk through the problem again.

If you can't figure it out, but you can reproduce the problem, you are ready to send that help request email.



# Requesting Help

Explain issues very carefully:

Be sure to  
compliment  
the  
software.

List  
software  
version,  
OS and  
hardware.

Describe  
**exactly**  
what  
conditions  
result in  
the  
problem.

If there  
is a log or  
error  
message  
generated  
... send  
that along  
too  
(verbatim)

Tell them  
what you  
tried  
(include  
updates,  
searching  
documenta  
tion and  
forums)


# Be Humble: Half the time, you are the Problem

At least half the time, I find the problem while I'm writing the email, because it is something I was doing wrong.


Of the emails I send, about half the time, the problem is something I was doing wrong.

# Summary

It is better to find an existing tool, than to have to create one.




Good tools are well documented, actively developed and well supported.



Install, open an image. Does it seem to work?



Try to accomplish the task you set out to do. Does it work?



Request help politely, and only after making a thorough effort. Document thoroughly.