

Document: DIASER code re-factoring decision table

Project: DIASER <http://diaser.org.uk>

Date: 02/11/10

Prepared by: Damian L Brasher Interlinux Ltd <http://interlinux.co.uk>

About: To use Perl Modules or not?

Summary: A table to help evaluate the benefits of using Perl modules instead of the current single file with subroutines.

+	-
Modules can lower skill barriers and drop access time to core functionality.	On or around intermediate code difficulty reduces the skill entry level for new coders to the entire code base. Initial understanding may be slower, offset by good commenting and consistent style. Create a style sheet.
Reduced risk of unnecessary code branching, code errors breaking the application on commit and merge difficulties.	Distributed version control will offset this.
Simpler management of >10,000 lines code.	Main application will not grow beyond 15,000. *No need to eat my hat if it does.
Easier code navigation.	All subroutines are written in alphabetical order. Reasonable knowledge of tools like bash, vim, grep, sort, awk, perl -en, and less - offset navigation problems. In fact use of these tools as a programming environment dramatically reduce the cost of development as they are free, fast, consistent and reliable.
Code in smaller chunks to ease navigation. Easy to see the application structure using a single list command.	Increases packaging overheads. Simple to manage and work with a single file. Effective use of tools like multiple shells, bash, vim, grep, sort, awk, perl -en, and less - offset visibility obscurity.
Larger blocks of bigger abstraction allow easier manipulation of ideas.	Lower level of abstraction for faster idea manipulation.
Debugging improved.	Everyday maintenance occurs at a lower level of abstraction so authors are closer to potential issues and less bugs are likely to occur. Important for security applications.
Sharing modules is good for the community.	Modules alone will lack useful context without the specialised application.
Testing changes can be done without breaking the main branch.	Distributed version control allows change testing between coders using change-sets.
Reduce use of large subroutine parameter lists. To simplify parameter management.	Is this just an aesthetic improvement? The same be achieved using arrays.

Outcomes:

The table is not evenly weighted. In favour of not using Perl modules. The development environment, version control, type of application and to an extent; methodology affect the decision. These factors will not change. *The decision, therefore, for the foreseeable future is to work in a single file and make some internal improvements to the design, gradually move over to distributed version control, write a style sheet including commands to list subroutines, for example - and use more arrays for long parameter lists. Overall reduced interpreter load.