SEP #1: Procedures for Scilab Enhancement Proposals

SEP: #1
Title: Procedure for Scilab Enhancement Proposals
Version: `1.0`
Author: Sylvestre Ledru <sylvestre.ledru@scilab.org>
State: Draft V4
Type: Procedure
Scilab-Version: All
Vote:"No: discussions and suggestions"
Created: Monday, 01 december 2008

Abstract

The subject of this SEP is to explain the principles behind SEPs (Scilab Enhancement Proposals), how to write them, and answer typical questions about rules, especially about adding or removing features in the Scilab version distributed by the consortium (FAQ)...

Rationale

Before introducing the SEP idea, there was no clear and transparent process for adding or removing features in Scilab. It was done internally in the Scilab consortium in an unformal way.

Because more and more contributors are involved into the project and also for transparency reasons, there is a need to define better processes explaining how the consortium and the community can make the software evolve in a common direction.

Scilab is also a programming language. Thousands of functions are available to users.

Evolutions are just standard in the lifecycle of a language/software like Scilab. The consortium and contributors feel the need to clearly explicit these evolutions, how they happen, how to suggest a new one...

Scilab Enchancement Proposals (SEPs) are introduced in order to formalize this process in an easy and efficient way.

Changes requiring a SEP to be written:

- Addition of new functions
- Addition of new features (a set of functions for example)
- Change of the type of an input/output argument
- Addition/Removal of an input argument from an existing function
- Addition/Removal of an output argument from an existing function
- Deletion of an existing function from the public interface (the public interface is the list of functions which are documented, i.e. for which there exist a help file)
- Change of the behaviour of a function, for instance improvement of the underlying algorithm, modification of tolerances on results, change in the output format of any value,

etc.

- Addition of new non-optional input argument to a function
- Any change that make an existing test (non regression, bug fixing) fail and in need of an update.
- Any change concerning the rules described in the present SEP#1

A SEP is not mandatory in any other situation. It is however recommended to write a SEP whenever the change in sight is more than trivial. Assessing where the exact limit is is left to common sense.

SEP use cases

A SEP can be used in two ways:

1) A Scilab developer/contributor (this includes any member of the operational team) needs to introduce a new function into Scilab.
2) A lambda user needs a new feature to be introduced into Scilab. He will write a SEP to explain exactly what he wants. The feature could be developed either by the Scilab consortium, or by a member of the Scilab consortium, or by any other contributor.

Steps

1. Send an email to the Scilab developer mailing list explaining what would be the feature (to know if anyone is already working on it, if there is not already a solution...).
2. Write your SEP (see the appropriate section below about this point)
3. Send the SEP to the Scilab developer mailing list
4. Wait for feedback/updates
5. Discuss
6. Iterate to the point 3 until a consensus is reached on the SEP.
   Definition of a consensus: consensus is reached on the SEP when someone proposes on the dev list that consensus has been reached, and no one contradicts the assertion within three working days.
   In case it becomes clear that no consensus can be reached, call for a vote (see the section dedicated to votes below)
7. Start the development (if not already done and included in the SEP as an implementation proposal)
8. Update the SEP in order to state that it has been accepted and implemented (in which version of Scilab, if changes to the specification have been needed)

Writing a SEP

A SEP is a specification of a proposed change or new feature in Scilab. The language used to write SEPs must be English.

OpenOffice format is recommended. Other open formats are accepted. SEPs cannot be stored under any closed format (MS Word for instance).

It is pretty easy and straightforward to write a SEP. A SEP template is available in the SEP directory

of the Scilab repository and shall mandatorily be used: just fill in the blanks. Additional sections not necessarily present in the template may be added whenever it is felt necessary.

Examples of optional sections in SEPs:

- Implementation

  Say a few words about how you intend to implement the feature. This section should be understandable by someone having a minimum knowledge of the internals. Don't explain everything from basics but put focus on the important points dealing with HOW the change will be implemented in the pre-existing codebase.

- Use case

  Typical use cases of a new feature

- Alternatives

  Explain here what other possible paths have been explored, why they have been rejected, what were the arguments for the decision. State what were the alternate options, either in the feature itself, or the way it is implemented.

- Compatibility

  Stress here any compatibility issue with respect of Scilab versions. For instance if a new feature uses function *readmymind* available only from Scilab 6, then this section should state that the change proposed by the SEP is not compatible with Scilab 5.x. or below.

  Also, the compatibility section should analyze any impact on the user's existing scripts. Non regression and backwards compatibility should be the rule. If this rule is broken by the proposed change, this MUST be said clearly in the SEP (and the "Compatibility" section is therefore no longer optional).

- Limitations and drawbacks

  Clearly state any limitation introduced by the changes summed up in the SEP. For instance, a new function *readmymind* will be able to read the developer's mind, but not the users's mind. This must be explained.

  Also drawbacks should be highlighted, such as any caveat resulting from a new feature, any risk of breaking user's existing scripts, etc.

- Still <TODO>
  List here any topics not yet dealt with that the SEP need to clarify.

- Detailed reference implementation
  If the feature does not require too many lines of code, it is possible to put the full implementation inside the SEP. Unicode diffs are preferred.

  A reference implementation of the proposal is in most of the cases a good idea to provide at SEP release time since it will allow other contributors to try the idea. The implementation may however be provided separately.

- Unitary tests
  Some unitary tests of the code would be appreciated

About SEPs contents:

- Profile consistency
  When you are writing a SEP about a feature similar to others already available in Scilab, please try to keep as much as possible a strong consistency.

String or plotting functions are good examples. In both functions *strstr* and *strsplit*, the haystack is the first input argument.

Voting

Voting is the last resort action when no consensus can be reached.

In such a case, voting is called on the developers mailing list. The voting period starts at the moment the call for votes is sent to this list. The duration of the voting period must be at least a full calendar week. The person calling for the vote may however select a longer voting period in his mail calling for the vote.

People that have the right to vote are exactly those that have write access to the Scilab repository. Among these people, no vote is more important than any other vote. There is no possible veto by anybody. One person has exactly one voice, no more, no less.

Votes are expressed publicly on the developer's mailing list.

Valid votes are: YES, NO, BLANK. (BLANK is the same as not voting at all).

A SEP is accepted whenever it receives at least 2/3 of the votes cast (not votes from the 2/3 of the people that can vote). Otherwise the SEP is rejected.

SEP workflow

Draft	(pending discussion, no consensus reached, or currently subject to voting – Increment SEP version any time it receives any change reflecting the ongoing discussion)
	|
Accepted	(consensus reached, or positively voted, possibly no implementation yet)
or Rejected	(consensus for NO, or negatively voted)
	|
Final	(spec. update done reflecting comments and changes discussed on the dev list, possibly still no implementation yet)
	|
Integrated	(the Scilab version including the feature must be mentioned)

How is the proposed change going to be incorporated in the standard distribution of Scilab ?

When consensus could be reached, or when no consensus could be reached but the vote was successful, the change can be integrated the way it has been specified in the SEP in the standard Scilab distribution. For legal reasons, the author of the implementation in Scilab must have signed the developer agreement. The originator of a SEP however may be another person, not subject to the terms of this agreement.

In no consensus could be reached, or in case of a negative vote, the change proposed by the SEP can not be integrated in the standard Scilab distribution. In such a case, there are other possibilities for making the feature available, such as creating a toolbox or use a (future) packaging system which will enable the diffusion of the features which have not been included in the standard Scilab distribution.

The procedure concerning the inclusion of the code into the official Scilab has been determined yet and should be the object of a new SEP in the future.

Storage of SEPs and version control

SEPs are stored in the same GIT repository as Scilab in the directory git.repository/SEP/. If you don't have write permission on the repository, please send an email to the developer mailing list, someone will push it for you.

Changelog

1.0 – Sylvestre Ledru <sylvestre.ledru@scilab.org> Initial version

Copyright