INRIA Evaluation Committee
Criteria for Software Self-Assessment
Version 1, August 2011

The goal of this document is to provide INRIA teams and candidates with a self-assessment mechanism for their software developments, to be used by the INRIA Evaluation Committee (EC) for evaluation seminars and internal or external recruitments/promotions. The EC considers software a major research outcome of the Institute. However, experience has shown that software descriptions in the team descriptions and the candidate application files are very hard to use, because team members and applicants are much less used to self-assessment of software than to self-assessment of theoretical contributions and publications.

In the sequel, "software" can mean an original stand-alone program, an element of a library, or a part of a larger system.

For each of the criteria listed below, please specify the self-assessment of the actual state and expected future evolution of the software. For evolution, no info = no change, "up" = should go to an upper level, "down" = should go to a lower level (in the numerical classification of each item). If you hesitate between one level and the level below, always choose the level below. If the source code development level is 4 while the documentation level is 4, choose 2 for the global software level.

The numbers associated to the different criteria in what follows don't hide a value ranking. The software with all 5's isn't inherently superior to other software. What the Evaluation Committee wants is a *fair self-assessment of the real state, evolution, and impact of the software*, which in principle should prove exact in a subsequent in-depth evaluation.

1. Characterize the software

Using the acronyms, please describe the real situation of your software. Examples:

- **A-2** means Audience = 2, "to be used by people in the team or close to the team (including contractual partners)";
- **SM-2-up** means Software Maturity = 2, "basic usage should work, terse but usable documentation, some software engineering, basic bug fixes done from time to time" and this is expected to go up;
- **SDL-3-up4** means Software Distribution and Licensing = 3, "distribution to an industrial partner in a contractual setting and where the software is actually used" and it is expected to be raised to 4, "public source or binary distribution on the web, organized by the development team."

You can also give a 1-line comment, if need be.

1.1. Audience (A)

1. personal or internal team prototype (to experiment an idea);
2. to be used by people in the team or close to the team (including contractual partners);
3. ambitious software, usable by people inside and outside the team but without a clear and strong dissemination and support action plan;
4. large audience software, usable by people inside and outside the field with a clear and strong dissemination, validation, and support action plan;
5. wide-audience software (aims to be usable by a wide public, to become the reference software in its area, etc.).

1.2. Software Originality (SO)

Here by ideas we mean algorithms, programming techniques, GUI, interfaces, etc.

1. none;
2. minor contribution to existing software, reusing known ideas;
3. original software reusing known ideas and introducing a few new ideas;
4. original software implementing a fair number of original ideas.

1.3. Software Maturity (SM)

1. demos work, rest not guaranteed, loose documentation, no real software engineering;
2. basic usage should work, terse but usable documentation, some software engineering, basic bug fixes done from time to time;
3. well-developed software, fairly extensive documentation, reasonable software engineering and testing, attention to usability, dissemination, bug fixes, and user feedback;
4. major software project, strong attention to functionality and usability, extensive documentation, strong software engineering, systematic bug chasing, and regression testing;
5. high-assurance software, certified by an evaluation agency (Common Criteria, DO-178, etc.) or formally verified.

1.4. Evolution and Maintenance (EM)

1. no real future plans;
2. basic maintenance to keep the software alive;
3. good quality middle-term maintenance, with persistent attention to users;
4. well-defined and implemented plan for future maintenance and evolution, making it possible for users to use the software without risk for important projects, organized users group.

1.5. Software Distribution and Licensing (SDL)

1. none;
2. basic source or binary distribution to the team or close community;
3. distribution to an industrial partner in a contractual setting and where the software is actually used;
4. public source or binary distribution on the web, organized by the development team;
5. External packaging and distribution: either as part of a popular open source distribution (e.g. a Linux distribution, an algorithmic or scientific library) or packaged within a commercially distributed product (Matlab, etc.).

2. Characterize your Own Contribution (OC)

For each of these categories:

a) design and architecture (DA)
b) coding and debugging (CD)
c) maintenance and support (MS)
d) team/project management (TPM)

Specify if you are:

1. not involved
2. an occasional contributor
3. a regular contributor

4. a main contributor

Examples: DA-2, CD-3, MS-1-up2, TPM-1, etc.

3. Additional information (free format, please be concise)

- Software's web site (if available).
- Objective of the software: domain addressed and precise technical goals; type of the software (autonomous, part of a library, part of a large system); performance goals, functionalities/algorithms, dissemination, etc.
- Users community: type (research, education, commercial/industrial, large open source community, etc.), size, how the community is animated, etc.
- Your assessment of the software's impact in your research community and your user community.
- Description of the state of the art, placement of the software w.r.t. the competition.
- Size of the software (lines of code), languages used, size of the development team, development effort, list of patents and main publications related to the software.
- Your contribution to the software (as a team or as an individual).
- Any meaningful additional information