Software Engineering For Astronomy

SUMMARY.
Software Engineering for Astronomy (SEFA) provides students with the background and techniques required for software development for astronomy. It provides a bridge between the branch of astronomy that teaches students theoretical, practical and instrumental observing skills and the field of computer science, which teaches students how to develop reliable and user-friendly software systems. SEFA teaches students the basics they need to learn software engineering from scratch: concepts, requirements and specifications, design, implementation and testing. Students will learn the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software. Experimental systems and software hands-on will be made possible. Students will implement their software designs and apply the software systems they have created to real-life problems in astronomy.

OBJECTIVES

We expect students taking this METEOR to acquire knowledge in both theoretical and practical aspects of software engineering. Students will learn (1) terminology from the computational and astrophysical sciences, (2) main concepts from software requirements analysis and specification to design and analysis (software design strategies, development life-cycle models, graphical user interface developments - GUI), (3) data and analysis software (FITS format, data reduction software, image display tools), (4) implementation and testing methodology (choice of languages, coding, testing, debugging) as well as software scalability, maintenance, and reliability models. Students learn to conduct a literature search and how to conduct software design and implementation phases that require reviewing documents. Students face telescope and instrument control systems (field rotation, active optics and adaptive optics) and data reduction problems and applications. In the lab, they learn the skills needed to implement their software designs created for real-life needs in astronomy.

PREREQUISITES

Natural interest in numerical modeling, practices, and astronomical instrumentation is expected. Basic programming experience is advised.

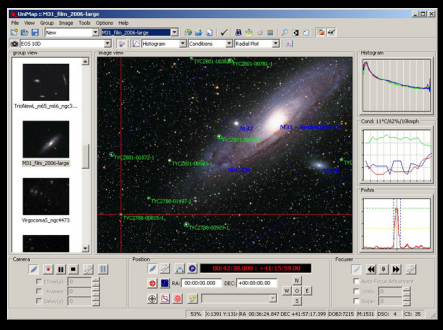
THEORY

by P. Martinez

Software is way more than just a program code. Software engineering is an engineering branch associated with software product development using well-defined scientific principles, methods and procedures, where the outcome is an efficient and reliable product. Software engineering is a systematic collection of past experiences arranged in methodologies and guidelines.

The theoretical part of the METEOR will provide an overview of the concepts, methods and principles necessary to achieve a good quality software development accommodating multiple functions.

Applications to astronomy will be provided. This part includes lectures, exercises, discussions of examples, and literature research.



Example of GUI developed for astrophotography (GUI: Graphical User Interface)

APPLICATIONS

by P. Martinez

How's the debugging going?
Credits: PhD Comics

Student projects will help students use the skills needed to implement their software designs. They will apply the software systems they have created to real-life astronomical problems. Projects include a re-use analysis phase to assess what is already available and to check if the problem their software intends to solve has already been proposed. Students conduct a literature search which requires web-based tools (e.g., NASA/ADS). Because any project has several designs and implementation phases that require reviewing documents, students learn how to conduct the required work for each of these, including requirements definition, conceptual design, functional decomposition, and final design. Student projects rely on real data or lab data from the SPEED facility (Segmented Pupil Experiment for Exoplanet Detection) at the Lagrange Laboratory (Valrose) or with available data from on-sky instruments (JWST, SPHERE, etc.).

MAIN PROGRESSION STEPS

The METEOR program is structured into 4 modules: (1) data and data analysis, (2) software design, (3) Implementation and testing, (4) telescope and instrument control systems, and (5) student projects with the following progression steps:

- First half of the period : theoretical courses (exam at middle or end term, tbc).
- Second half of the period : student projects, final report at end term.
- Last week: preparation of the final oral presentation and term project report.

The METEOR program is based on various pedagogic structures:

- (1) Focus lectures that are opening lectures on a single and specific topic (e..g., SAO DS9, NASA ADS, Class diagrams, Graphical User Interface design process, SPEED control infrastructure),
- (2) Computer practicum that are numerical practical work (e.g., Image and data reduction quality, Strehl ratio and image quality metrics, Zernike PSF and MTF, App designer, Conceptual diagram),
- (3) Labs hands-on that are practical work in lab environment (e.g., Camera control software, Deformable mirror control software, SPEED interface control system),
- (4) Reading assignments that are active learning based on scientific articles,
- and (5) Mini-project that represent immersive work on software development, one to choose from the list (WebbPSF$^{[edu]}$ software, ABISM$^{[edu]}$ software, Camera controller software, dOTF wavefront sensor software, Fourier Optics GUI). Students can propose their own mini-project topic (subject to validation).

EVALUATION

Active learning is expected and evaluations will be proposed in the form of:

- reading assignments (written/oral questions may be asked);
- homework assignments (oral presentation may be asked);
- part way through the METEOR a numerical project will be proposed and will be coached and facilitated;
- hands-on experience with the hardware and software components will be made possible;
- a final exam (conceptual essay and/or questions and quantitative problems).

The final grade is based on 40% defense evaluation at the end of the METEOR, 30% evaluation (exam 20% + homework/reading assignments evaluations 10%), and for 30% evaluation with the student projects.

Student projects consist of four components, which comprise about 10 pages of report in total: (1) a page project proposal to be approved by the instructor before work begins, (2) a two-page write-up at the end describing the projects end functionality and any insights gained and/or difficulties encountered, (3) the software itself, usually about few hundreds of lines of code.

BIBLIOGRAPHY & RESSOURCES

- SPEED system control and software infrastructure
- Fundamental of software engineering, PHI Learning - 5th Edition
- Software systems for Astronomy, Springer - 2014e Edition

CONTACT

☎ +33 (0)4 92 07 63 39
✉ patrice.martinez@oca.eu