Challenges of long term support of legacy software in the SIMBAD service

Anaïs Oberto, Grégory Mantelet

Challenges of long term support of legacy software in the SIMBAD service.

Anaïs Oberto[1] and Grégory Mantelet[1]

[1], *Université de Strasbourg, CNRS, Observatoire astronomique de Strasbourg, UMR 7550, F-67000 Strasbourg, France;* anais.oberto@astro.unistra.fr

Abstract.

SIMBAD is a service that provides reference data about celestial objects for astronomers that has been in operation for 50 years. Over this time astronomical data themselves have evolved because of the technological evolution of the observations, but also because of the evolution of software engineering. SIMBAD has gone through four major versions, each of which has been built using a different programming language, or other architecture, but with a long-term view to be scalable for future evolutions.

For instance, the very first interactive version of SIMBAD, in 1981, allowed queries for data with a constraint on the sky coordinates. Now, in the 4th version of SIMBAD, this service is still available, along with many other options. For each new version, in order to assure back-compatibility, the URLs and interfaces already used in the past are still available in parallel to the new version. The maintenance of such legacy software is necessary for the continuity of the service and to avoid problems for the many other community services that use SIMBAD. The support of such a software is challenging because it cannot be fully integrated with new developments. We will describe the range of different solutions that we use: specific redirections, hacks, interfaces, filters that are currently used for SIMBAD utilities.

1. Introduction - What is SIMBAD?

SIMBAD (Wenger & Ochsenbein 2000) is an astronomical object database containing (currently, in 2022) more than 13 million of astronomical objects, basic-data, cross-identifications, bibliographic references and measurements.

It was initially populated from few main catalogues (Henri-Draper, Smithsonian Astrophysical Observatory, General Catalog ...). Now, SIMBAD is a meta-compilation built from what is published in the literature and from our expertise on cross-identifications. It is continuously updated.

SIMBAD database software is developed and maintained by the Centre de Données astronomiques de Strasbourg, France. The first version was created in 1972. Today SIMBAD is in its forth release ((Wenger & Oberto 2007)). Now, fifty years after its creation, we are working on a new one (5th).

2. SIMBAD I/II - Hardware driven developments

Since the beginning, 1972, the database was built to store many objects, and this was a technical challenge on mainframes with low capacity of disks and memory. The data

were stored as flat files, and then the first developments (PL1 and IBM assembler languages) tried to optimize usage of processors and memory for easy access. A whole astronomical object should be stored on only 36 bytes containing the coordinates, magnitudes, a spectral type, 5 main identifiers and a set of flags defining the appartenance of the object to catalogs. At this time, astronomers were only able to get the data as punchcards, print outs on paper, or through a terminal directly connected to the server. Fast consultations of the database used a whole printout of the database on paper (10,000 pages).

In 1981, *"SIMBAD II"*, a new version was developed, because the hardware changed of location and type (Paris IBM to Strasbourg Univak). A new version of assembler code has been re-written according to the new processors.

3. SIMBAD III/IV - User interfaces

The arrival of the network brings the capacity to also develop some client software. This was the introduction of interactive queries into SIMBAD. In 1990, a new version was developed, fully re-written, and based on the same algorithms of the previous one. And then came the usage of Graphical User Interfaces for SIMBAD, first as standalone applications in C language, and then in interactive usage thanks to the World-Wide-Web.

In 2006, the new *"SIMBAD IV"* was released. New developments in Java language for client-side have emerged, accordingly to a new Java server (full re-write of the previous C-server). With this version, it starts to use a Database Management System: PostgreSQL. The usage of scripts through HTTP marked an important evolution of the astronomers behavior, as they were able to more intensively use SIMBAD. And now, SIMBAD is widely used (near 350K queries per day), thanks to multiple interfaces allowing queries by any sort of criteria, for instance:

- By name
- By position
- By magnitudes
- By bibliographical reference
- By any type of data, and by any combination of others ...

SIMBAD is also available through nice visualization and script interfaces, but also through many sort of Application Programming Interfaces (API).

4. Challenges of software evolution over years

Over the 50 years of the SIMBAD life, developers have followed the evolution of astronomy, and its usage. The technology itself evolved, and the SIMBAD code has been adjusted to take advantage of these new possibilities driven by scientific needs. Little by little, more functionalities have been added, and the more we support functionalities, the larger the development code is (which is a challenge in itself).

At the beginning, one of the difficulties for developers was to deal with the low space storage, and the solution was to store some date as binary. Nowadays, this volume limitation is not necessary anymore for SIMBAD, and PostgreSQL do the optimization

storage for us (currently 34Gb). Besides, it seems obvious to choose an hardware independent language. Unit testing is also quite a challenge on a 50 years old code, as none or very few was written at that time. The currently adopted strategy to improve this situation is to build them on needs for each new or updated feature.

The maintenance of APIs is very sensitive because of the use of SIMBAD inside other tools (as an external service that provides position from a name, a list of bibliography for an object, a known object name in a region, or maybe a link to a web page from a name, and many other usages).

We are taking into account these usages when updating SIMBAD in order to avoid breaking down what already exists. It should especially be guaranteed that new developments always returns same results as before. As long as a client is active, the server should answer (for instance, we still have C-clients), and we suggest and help users to change from their old usage to a modern one.

When we have to make some modifications on an old interface still in use, we propose a new solution in parallel for a smooth evolution for users; when possible the new feature will give the same response. In some cases, the usage of a specific proxy to reproduce the same behavior is made. As much as possible, before applying any modification we build non regression tests in order to reproduce the previous behavior that should pass successfully. We can not do that in all cases, some old usage has been removed little by little:

- HTML1 output has been removed smoothly between 2000-2005
- Mail queries has been fully stopped in 2006
- C-server not maintained up to date since 2010
- SOAP disappeared in 2015
- Corba was replaced by RMI, which disappeared in 2020

Example on cone-search facility

The feature to search by position have seen many evolutions in terms of options proposed to the user: refine search position/radius, input criteria on object type, bibliography etc...

It also evolved in term of output, regarding to new data:

- selection of names (for instance, only few names were given, and currently one object could have until 140 different identifiers)
- we added more magnitudes
- many formats to interact with other tools (Json, VoTable, XML, ASCII, HTML)
- or options to customize output (sort by bibliography, distance ...)

For this cone-search alone, we still have to maintain few C-client queries in order to avoid breaking old tools using SIMBAD (we do this as few as possible, due to the difficulty maintenance for us). HTTP queries also evolved, and even some, stopped. In this last case, users are warned thanks to newsletters or HTTP return code).

5. Timeless challenges

5.1. Security

The challenge on security needs a constant effort because there are always new risks. We use tools to check regularly vulnerabilities and libraries that are updated to prevent

attacks. The constraint of developping internal security controls internally is to heavy to maintain to be reactive enough.

5.2. R&D

Time spent on prototyping, and investment in Research and Development are necessary to be ready for future technological developments. Indirectly, it will help to maintain the full project code and make it evolutive.

5.3. Clean code

In order to keep a code "readable", and to avoid dead code, the first step is to explore the logs to determine what is still used. Then, we remove some features (not fully backward compatible) to avoid maintaining unused functions. SIMBAD data content and complexity has grown and will still continue to grow. The code must always be adapted to be generic as possible.

Whatever the language is, the code must be adapted in order to follow new major language releases. It must especially be restructured in a way it is easy to maintain. Code readability can also be improved thanks to new conventions and coding styles.

Finally, a good practice is to keep following standards evolutions (e.g. VO), as standards provide well defined specifications and documentation. This helps a lot in their implementation maintenance as well as keeping a clean code.

6. Conclusion

The software development life cycle should follow software design paradigms changing from one to an other (for instance: *"bring the data to the code"* or *"bring the code to the data"*). Recently, the new Green-IT development is an aspect to take into account. It especially encourages to optimize CPU time used for queries.

SIMBAD have seen many evolutions, and will continue to evolve. The interfaces, and capabilities are always moving, some new will appear and other will disappear. We encourage external developers using SIMBAD, to use as much as possible modern interfaces.

In its way to the 5^{th} version, we see some architectures of interest emerging. For instance, micro-services allow to separate developments and features, while allowing to use multiple technologies inside a same general application.

Acknowledgments. We acknowledge the support of CNRS-INSU, the Centre National d'Études Spatiales (CNES) and the University of Strasbourg.

References

Wenger, M., & Oberto, A. e. 2007, in Library and Information Services in Astronomy V, vol. 377 of ASPC Series, 197

Wenger, M., & Ochsenbein, e. 2000, A&AS, 143, 9. astro-ph/0002110