DISCUSSION CONCLUDING AAS 13-518

DENNIS MCCARTHY was interested in ANDREW MAIN's opinion that leap-second announcements would arise via alternative means if the IERS stopped this service, because of continuing demand for them. MCCARTHY asked MAIN "Who do you think would want to do that?" MAIN replied that at least one person from the leap-second mailing list* had already volunteered to provide this service—a person who insists that all of his clocks run on UT1 rather than UTC, although MAIN was not convinced that his clocks really do this. But MAIN's point was leap seconds will arise if there is any serious application for which they are found useful. If one looks at the history of UTC, there was an experimental "Stepped Atomic Time" (SAT) which was using 200 millisecond jumps. SAT was a good idea that was necessary to invent and from which the current UTC definition originated—and "we did not need a top-down authority telling us that."

JOHN SEAGO asked if MCCARTHY recalled a particular response of Mark Storz during the Exton Colloquium.[1] Storz was responsible for systems that relied upon leap seconds, which might fail if UT1 and UTC diverged. Storz contemplated that his organization might introduce its own system of leap seconds if leap seconds were officially abolished, but eventually realized that such an approach might cause compatibility issues because their systems are not self-contained and must deal with outside data standards. JIM KIESSLING believed "the horse has left the barn;" as long as the U.S. Naval Observatory and the IERS continue to publish *Bulletin A* "you can roll your own" leap seconds. MCCARTHY agreed, clarifying that he was actually curious to know who would continue using leap seconds if UTC was redefined without them, other than using it as a workaround as SEAGO mentioned. MCCARTHY said that so far during the meeting, he observed an impressive amount of time and effort working around the leap second.

ROB SEAMAN noted that the software issue is very much "Y2K-like". And like Y2K, one can either 'fix it right'—whatever that means by whatever standards are of the future—or, one can do what he expects will happen, which is "you will not be given enough money to 'fix it right' and you will work around it in innumerable ways", particularly within the thousands of systems within astronomy. SEAMAN supposed that mountaintop observatories would deploy clocks that mimic UTC as it currently exists in some fashion; they would not all adopt an explicit UT1 model but will continue to use 'UT' as meaning a general equivalent to 'GMT'.

MAIN brought out a related situation which he had not found discussed much. He noted that there have been various proposals that would involve loosening the |UT1−UTC| bound for all sorts of reasons, and there are certain users of UTC who would object to such loosening. So if there were an attempt to loosen the bound, we could see a split between a "looser, canonical UTC" versus some "service arising organically that provides a UTC-like time scale within the old bound." Yet there could also be a split the opposite way around: if UTC keeps its current definition, there are some users who would get so much benefit from a longer scheduling horizon that

* http://six.pairlist.net/mailman/listinfo/leapsecs

one could imagine the organic production of a UTC-like time scale that has a looser |UT1−UTC| bound.

RUSSELL REDMAN very much feared the proliferation of workarounds that imitate what UTC does now, but with different assumptions. Once people start doing that, everyone will make different assumptions depending on their immediate problems to be solved, and only manage the compatibility between different inputs later. This would become "an undiluted nightmare which will take another decade to solve." For such an intrinsically simple problem as this, it is far better to have a central authority that does it right, and have everyone else using that. KIESSLING likened this to a "rebellion growing up which discounts the ITU-R" redefinition.

If *status-quo* UTC is suppressed and an organic replacement comes about, MAIN thought "we might see multiple versions; we might see schisms." So any attempt to suppress UTC as currently defined would actually multiply the number of time scales that use leap seconds. SEAMAN agreed that "it would get more complex, not less complex." REDMAN added "…and less reliable." STEVE MALYS said that the availability of time via GPS would make for more possibilities; UTC is already there because GPS already provides some realization of UTC. MAIN said that GPS regards cæsium as the precise step, and one could build whatever one wanted on top of that.

SEAMAN had tried to recruit a database/SQL expert from the University of Arizona to speak at this colloquium. By SEAMAN's estimation, this expert had argued that the right way to normalize a database is to have two sets of times: the time native to the thing being described, and the time stamp of when it becomes valid. The expert's argument was couched in terms of financial databases and similar applications, but MAIN's illustration of the leap-second table was a perfect example that needed to be captured to make things even better. MAIN thought it would be slightly difficult to normalize that into SQL because SQL does not support ranges of values as keys. SEAMAN said that was the precise point of the expert, who was arguing for extending the Structured Query Language.

JOHN SEAGO wondered if the application programming interfaces (APIs) proposed by MAIN would be used by programmers. Because of their lack of familiarity with various time scales, they also lack familiarity with the additional complex inputs of the interfaces. MAIN said that this was a tricky question which touched on a point that MAIN would present later on behalf of STEPHEN COLEBOURNE. In the Java world, a different approach was taken for developing a time API. They considered an approach similar to that discussed by MAIN but rejected it on the grounds that it would be confusing. MAIN said that he therefore had two different answers for his two different talks…

MAIN's aim is really at skilled programmers. When writing real, substantive programs, one need not be concerned about all the underlying details. SEAGO agreed that when programmers call functions, they assume the experts behind the API are providing proper functionality. MAIN continued that one is rarely going to seek to have time labels 'unwrapped'; one only deals with a raw time label if one is implementing a time-scale conversion; anywhere else one deals with a time value in its 'wrapped up' form. Then the conversion operation is as simple as it could possibly be; there, one is using a 'wrapped up' time value. It is vitally important that anything built on top of that use the 'wrapped up' structures, otherwise we could end up with a profusion of bad and confusing APIs. MAIN thought this is the way to make computers handle the details automatically. And there is always the option of layering simpler APIs over the top of this to provide an implementation that does not provide so many sharp corners.

REDMAN liked this kind of approach, as it especially is one of the places where encapsulation is absolutely critical. An API is needed that is rich enough to do what true experts need, encapsu-

lated within a general-purpose library with appropriate defaults that will do the correct thing for the rest of the world. MAIN explained that is very much the intent behind the designs. Current time APIs have not been developed by subject-matter experts; even when developed by programming experts they have not captured the richness of time scales.[2, 3] SEAMAN remarked that programmers "may not use the APIs if they have them; they *will not* use them if they don't have them." MAIN and REDMAN agreed. MAIN added that there is much more implementation work to be done before these APIs can be used in earnest.

KIESSLING wondered about dealing with real-time issues, because there are very few real-time operating systems for system control. MAIN replied that real-time systems are a specialist community that he has not looked into very much. MAIN had designed this API to be used in a very-high-level language, which leads also to the possibility of analyzing a program to see which conversions it will need and then compile down to a program that has none of the flexibility of the source language but can run in bounded time. KIESSLING said he watched an application which accessed the processor clock and periodically looked at the number of cycles between different calls and got around all of the other operating system issues; that "is the joy of going through undocumented processor issues." MAIN said that presently, and in the last decade particularly, central processing units (CPUs) tend to provide direct access to a count of their own clock cycles, and that results in a fairly quality clock that can be accessed extremely quickly—in one instruction perhaps; they are very convenient. Spacecraft take time readings directly from a local oscillator; these days this kind of access is built into the processor for some purposes. KIESSLING noted that Windows® and many of the other operating systems are just not suitable for real-time. MAIN agreed that one needs an operating system specifically built for real-time work.

REFERENCES

[1] Seago, J.H., R.L. Seaman, S.L. Allen (2011), *Decoupling Civil Timekeeping from Earth Rotation—A Colloquium Exploring Implications of Redefining UTC*. American Astronautical Society Science and Technology Series, Vol. 113, Univelt, Inc., San Diego. p. 231.

[2] International Standards Organization (1990), International Standard ISO/IEC 9899:1990, *Programming languages — C*.

[3] Gosling, J., *et al*. (1996), *Java API Documentation*. Sun Microsystems, Inc.