The long pole gets longer

document id: N2893=09-0083

date: 2009-06-19

author: Martin Tasker (martin.tasker@nokia.com)

Acknowledgements: this document owes a lot to diverse opinions, ideas and comments on its subject matter, from Steve Adamczyk, Doug Gregor, Howard Hinnant, Alisdair Meredith, P J Plauger, Bjarne Stroustrup, Herb Sutter, James Widman, and the BSI C++ Panel meeting of 21/4/09.

1. Introduction

This is a non-technical paper, which makes the case that the "long pole" in the C++0x schedule, namely Concepts, is getting longer; and then presents options for a revision of the C++0x schedule agreed at Sophia Antipolis.

The purpose of the paper is to help consensus-building for a decision on this matter, at Frankfurt.

- by presenting the situation, and the likely consequence of no consideration of schedule at Frankfurt, to build consensus that there *is* a need to consider the schedule at Frankfurt
- by presenting a brief factual evaluation of a starter-set of options, to give such a discussion – not usually relished by Committee participants – something to go on

The paper starts by explaining the situation in terms of Concepts, along with the schedule decisions made in June 2008 at Sophia Antipolis, to do a CD in October 2008, an FCD in October 2009, and an FDIS in 2010. It then covers developments since, which have impacted the rationale behind those decisions, and which appear to make it infeasible to produce a standard to the anticipated quality on that schedule.

This makes it imperative to consider options for a new schedule. The next two sections of the paper work towards that:

- first, some assumptions are stated which have emerged during discussion of this issue, which have influenced the formulation of options, and which will (one way or another) influence consideration of them
- second, six schedule options are presented, each with pros and cons

There is no claim to exhaustiveness: the options presented here are a starter set.

There is no attempt to select between the options. The author's view is that it is better for individual readers to consider them, to discuss them, and to make up their mind based on their experience, and some reflection on the alternatives.

2. Current situation

2.1. Concepts in C++0x

Concepts is a mechanism for describing requirements on types and combinations of types, and is useful for early detection of errors in template code.

The ideas behind Concepts have been around in language research for some decades; they began to be investigated in the context of C++ soon after the 1998 standard, and have been a cornerstone of the intent of C++0x since the beginning of the project.

Concepts is a huge feature, affecting the language (~40 pages of new text in [concept] and [temp.constrained], along with tweaks throughout the language-related clauses) and the library (pervasive impact). To deliver technical consistency, practical usability, and backward compatibility is a non-trivial undertaking.

2.2. Sophia Antipolis

Herb Sutter, as then Convenor, published a public report of Sophia Antipolis (June 2008), at http://herbsutter.wordpress.com/2008/07/04/trip-report-june-2008-iso-c-standards-meeting/. He wrote:

> The biggest goal entering this meeting was to make C++0x feature-complete and stay on track to publish a complete public draft of C++0x this September for international review and comment — in

> ISO-speak, an official Committee Draft or CD. We are going to achieve that, so the world will know the shape of C++0x in good detail this fall.
>
> We're also now planning to have two rounds of international comment review instead of just one, to give the world a good look at the standard and two opportunities for national bodies to give their comments, the first round starting after our September 2008 meeting and the second round probably a year later. However, the September 2008 CD is "it", feature-complete C++0x. The only changes expected to be made between that CD and the final International Standard are bug fixes and clarifications. It's helpful to think of a CD as a feature-complete beta.
>
> Coming into the June meeting, we already had a nearly-complete C++0x internal working draft — most features that will be part of C++0x had already been "checked in." Only a few were still waiting to become stable enough to vote in, including initializer lists, range-based for loops, and concepts.
>
> Of these, concepts is the long-pole feature for C++0x, which isn't surprising given that it's the biggest new language feature we're adding to Standard C++. A primary goal of the June meeting, therefore, was to make as much progress on concepts as possible, and to see if it would be possible to vote that feature into the C++0x working draft at this meeting. We almost did that, thanks to a lot of work not only in France but also at smaller meetings throughout the winter and spring: For the first time, we ended a meeting with no known issues or controversial points in the concepts standardese wording, and we expect to "check in" concepts into the working draft at the next meeting in September, which both cleared the way for us to publish a complete draft then and motivated the plan to do two rounds of public review rather than one, just to make sure the standard got enough "bake time" in its complete form.

In short, Herb's posting reduces to,

- schedule: CD in October 2008, FCD October 2009, implied FDIS in 2010, C++0x == C++10 == C++0xa
- Concepts: the "long pole" feature in this schedule

2.3. Post-Sophia Antipolis

Concepts specification

The status Herb reported from June 2008, of "no known issues or controversial points in the concepts standardese wording", has unfortunately not proved stable: vigorous discussion of Concepts continues, on points in both language and library.

It is possible that this level of discussion is within the parameters expected at Sophia Antipolis; but we now have a year's data to indicate the pace at which these discussions are converging, and this should be taken seriously in any consideration of the achievability of the Sophia Antipolis schedule.

Concepts implementation

There has been significant slippage in expectations around Concepts-related implementations:

- GCC: the ConceptGCC branch, which was the only prototype language implementation, and therefore the only means by which practical experience could be gained in the library, has had no commits since 17th October 2008. No further GCC-related project work is in prospect, either on the ConceptGCC branch or any other. Note that the ConceptGCC branch, as a work-in-progress when development ceased, is buggy, and lags the draft specification of Concepts even at that time, so it's not suitable for any current Concepts-related experimentation in the library.
- Visual C++: Microsoft may have been assumed to be working on Concepts at Sophia Antipolis. But on 2009-04-01, in message c++std-lib-23525, Herb Sutter posted, "My own company is implementing a number of C++0x features in our next release, including rvalue refs and lambdas and such, but I have no ETA yet for looking at concepts, sorry."
- EDG: on 2009-05-22, in message c++std-lib-23866, Steve Adamczyk posted, "we're not likely to be able to start on concepts until mid-2010, which means we are unlikely to have a released implementation before mid-2011."
- other implementers: no data

A reasonable interpretation of these publicly available facts suggests – to the author – the following commitments, best-case scenarios, and most-likely scenarios, for Concepts implementations in compilers in line with draft standard wording, which can be used as the basis of experimentation in the library:

	committed date	***best-case scenario date***	***most-likely scenario date***
GCC	none	mid 2010: estimate based on (a) a well-funded expert individual/team which starts work immediately, and (b) a development duration similar to EDG's, implied by Steve's posting cited above	2011-2012 or later, since no such funding and individual/team is presently known to exist
Microsoft	none	mid-late 2011, since it seems reasonable to assume Herb would "have an ETA" if Microsoft had scheduled Concepts, even tentatively, for any 2010 products	no better than the best case
EDG	none	mid 2011, based directly on Steve's posting	no better than the best case

Although individual details in the table above may be debatable, it seems beyond doubt that this table represents slippage with respect to most parties' expectations at Sophia Antipolis. At Sophia Antipolis, it might have been assumed that ConceptGCC would keep up with the draft standard, and that it would be possible to continue library experimentation based on that. Now, useful library experience appears most unlikely before 2010 – a slippage of 18 months or more.

2.4. Summary

The assumptions which informed the scheduling decision at Sophia Antipolis appear to have been invalidated since. In particular, in relation to Concepts, already the long pole at Sophia Antipolis,

- specification work is converging slowly (though whether more slowly than anticipated, isn't clear to this author)
- implementation work has stalled
- optimistic expectation now is that Concepts implementations are ~18 months later than optimistic expectation might have suggested, at Sophia Antipolis

So it seems the long pole has got longer, and a re-visit of the schedule seems necessary.

3. Some considerations

The following considerations arose in the author's consultations which led to the schedule options presented later in this paper, and may be useful in readers' consideration of the schedule options.

3.1. Interoperability/defragmentation: standards vs cooperation

Innovation is good for users, so is interoperability. The point of a standard is to maximize both.

At this point in time there is no real concern about fragmentation in Concepts. Rather, the concern is about fragmentation in other C++0x features, caused by delay in standardization, caused in turn by waiting for Concepts.

In a simplistic perspective, standardization – of features which are ready for it – sooner is better, while standardization later is worse. But reality isn't quite that simple: the implementers in the C++ community cooperate on feature specification prior to ratification of the standard; and users can make their plans based on implementers' products and roadmaps.

So it's not quite black-and-white and abstract: Committee members must use their experience and subject knowledge to determine how much harmful fragmentation might result from a delay in C++0x standardization.

3.2. The role of implementations

For any software feature there is a chicken-and-egg question between specifying it, and getting a working implementation. There are strong arguments for getting an implementation before finalizing a specification, and strong arguments for doing a thorough design (which isn't far from a specification) before doing an implementation.

The bigger the feature, the less effective it is to take an extreme view: design and implementation need to work together to get something which both works, and is usable.

Concepts is a huge feature, and there's been no implementation in relation to Concepts for the better part of a year now. Committee members need to form a judgement about when the lack of current implementation and usage experience of Concepts will begin to inhibit further work on Concepts specification.

3.3. Bigger context

C++ as a language has competition – from high-level runtimes such as JavaScript, .NET etc; from hardware such as graphics accelerators; and from C and Objective C. Ideally, the pace of overall C++ feature development should be such as to maintain C++'s competitive position against such alternatives.

Among C++ user communities, C++ programming styles vary between C with classes (and perhaps a few templates), through OO, to GP and full-on meta-programming, with each significant C++ user community committed very strongly to one position or another on this spectrum. Ideally, the pace of feature development relevant to each C++ user community should not be held up by waiting for features more relevant to other C++ user communities.

4. Options

After consultation, then, the following options have arisen from discussions with a number of parties on this subject.

4.1. Issue now, fix later

Keep the Sophia Antipolis schedule and issue the C++0x FCD "now" (ie October 2009) with FDIS following in 2010.

Due to the lack of implementation availability within that schedule, there will be Concepts-related bugs and usability issues in the FCD, and still in the FDIS. Fix them in TCs: this will eventually converge onto something stable.

Main pros:

- C++0x delivered on schedule (==, by definition at the time of writing, the Sophia Antipolis schedule)
- C++0x includes Concepts, in line with long-set expectations

Main cons:

- the C++0x FDIS will be messy to an unprecedented degree due to bugs in Concepts
- NBs, and therefore ISO, may reject such a messy FDIS
- no implementer will be able to make [positive and crisp] conformance claims in relation to C++0x until the Concepts-related TC process has converged
- the effort expended on stabilizing Concepts during the 2009 and early 2010 may distract focus from non-Concepts features in C++0x and reduce their quality also
- it's impossible to see, at this time, when the Concepts TC process might converge

4.2. Bring Concepts in

Secure the earliest possible implementation of Concepts in one compiler or another, so that meaningful library experience can be gained.

Note that a production-quality compiler implementation is not necessary in order to gain meaningful language and library experience, as required to progress the specification. Therefore it's possible that the time taken to develop such a compiler may be less than the time given in the "best-case scenario date" column in the table above.

However, for this option to be really different from other options, it does require commitment (which probably implies funding) and capability (a suitably expert individual or team), to be secured (not merely talked about).

Adjust the timescale of C++0x to fit with the schedule expectations set by that individual or team.

Main pros:

- avoids the messiness and TC trail involved in "issue now, fix later"
- gives confidence that standard wording in library is based on experience
- enables a realistic deadline to be set for a C++0x which includes Concepts

Main cons:

- even in the most optimistic scenario this involves a year's delay to the current schedule, ie FCD October 2010, FDIS 2011, C++0x == C++11 == C++0xb
- there's no such commitment and capability in sight at the time of writing. Who will put up the money, and who will do it? Unless specific, high-confidence, answers to that question appear by the time a decision is made, this option cannot be included for serious consideration in the decision.

4.3. Wait on Concepts

Keep Concepts in C++0x. Don't stabilize until there is implementation experience and a reasonable confidence that the FDIS can be [reasonably] clean and bug-free.

Delay the FDIS until these conditions apply.

Main pros:

- C++0x FDIS will be of usual high quality
- C++0x includes Concepts, in line with long-set expectations

Main cons:

- in the most optimistic scenario this delays C++0x by two years, ie FCD October 2011, FDIS 2012, C++0x == C++12 == C++0xc
- there are no grounds, currently, for being confident that this optimistic scenario can be delivered
- one or both of the above criteria may cause ISO to push back on SC22/WG21 and require a re-appraisal of the C++0x standardization project
- no implementer will be able to make [positive and crisp] conformance claims in relation to C++0x until C++0x has stabilized

4.4. Decouple Concepts

Take Concepts out of C++0x – out of the document, and out of the process.

Maintain the schedule for C++0x and its current specification apart from Concepts.

Handle Concepts in a separate project (perhaps, but without loss of generality, in the Evolution Working Group), which can develop with its own pace and style until it's ready, with high confidence, for integration into a standardization project which could be (without loss of generality) C++1x.

Taking Concepts out of the WD could be done either by striking out from today's WD, or by rolling back to post-Sophia Antipolis WD and re-applying non-Concepts papers (or parts thereof) since then.

Main pros:

- C++0x FDIS will be of usual high quality
- C++0x will be delivered on schedule
- implementers will be able to make [positive and crisp] conformance claims in relation to C++0x, within timescales which they can predict
- C++0x will be feature-rich in language and library, even without Concepts
- Concepts can be developed without artificial deadline pressure from C++0x
- there will be more experience of Concepts before its specification is finalized

Main cons:

- C++0x will not include Concepts, which has been billed as a defining feature
- however it's done, it will require very significant effort to take Concepts out of the WD
- there will be additional language feature impact in the WD, eg features such as `for(:)` which depend for their usefulness on Concepts
- there are risks to community cohesion around the ISO C++ Committee, given the history of Concepts to date

4.5. Decouple language and library

Decouple the C++ language standard from the C++ library standard.

Exploit this decoupling to focus on Concepts in C++0x/language while omitting Concepts from C++0x/library.

Main pros:

- both C++0x/language and C++0x/library FDIS issued on time (theoretically)
- C++0x/language FDIS includes Concepts, therefore C++0x does, in line with long-set expectation
- C++0x/language and C++0x/library FDIS are of high quality: C++0x/language because Concepts is easier to do just in the language; and C++0x/library because Concepts is omitted
- separation of language and library has other beneficial effects (independent of Concepts)

Main cons:

- there is nothing remotely approaching consensus, in the ISO C++ Committee, that separation of language and library is the right thing to do
- it would require a very large effort to put this separation into effect: probably this will delay things so that, in practice, C++0x will be delayed
- separation of language and library wouldn't solve the problems with Concepts anyway, since with Concepts it seems of the essence of the problem, that the usability and utility of the language feature must be honed in conjunction with a library
- separation of language and library has other disbeneficial effects (independent of Concepts)

4.6. Semi-decouple Concepts

Take Concepts out of C++0x process: focus all issue handling, consideration of Nxxxx papers, and response to NB comments on ISO drafts, on features other than Concepts.

Maintain the schedule for C++0x and its current specification apart from Concepts.

Do not take Concepts out of the document: just mark Concepts areas as "here be dragons" (or something more formal to that effect) and advise implementers and users to ignore them for the time being.

Handle Concepts in a separate project (perhaps, but without loss of generality, in the Evolution Working Group), which can develop with its own pace and style until it's ready, with high confidence, for integration into a standardization project which delivers, in effect, a huge TC to the C++0x FDIS.

Main pros:

- C++0x FDIS will be of usual high quality, except for the marked here-be-dragons areas around Concepts
- C++0x will be delivered on schedule
- C++0x will be feature-rich in language and library
- Concepts can be finalized without artificial deadline pressure from C++0x
- the work avoided in removing Concepts altogether from the WD will not be needed
- reduces the risk to community cohesion involved in the stronger "decouple Concepts" option

Main cons:

- users and implementers will be confused by the "here-be-dragons" areas of the FDIS
- it's not clear that the ISO process supports "here-be-dragons" markings of any kind
- even if implementers are clear what their conformance claims mean, users may get confused
- although this option requires somewhat less work to the WD than the stronger "decouple Concepts" option, it does still require substantial effort to identify areas of the document as "here be dragons" and to mark them
- while working on C++0x, the Committee itself (and NBs) will be confused by this compromise and will drift instead into option 1 ("issue now, fix later") or option 3 ("wait for Concepts")

5. Summary

The options, in summary, are:

1. issue now, fix later
2. bring Concepts in
3. wait for Concepts
4. decouple Concepts
5. decouple language and library
6. semi-decouple Concepts

The purpose of a schedule discussion should be to choose a feasible schedule, either from the options above or another one (there is no claim made here to be exhaustive).

Option (5) is probably infeasible. If the time for (5) was ever right, or will ever be right, it is not now.

Option (2) is feasible only under a set of strict criteria, namely that a convincing commitment is made, publicly, and at latest before a schedule discussion is brought to consensus, to develop a Concepts implementation, at least in a form which allows experimentation at library level, in a timescale which makes a difference.

Option (2) is just the most obvious option which requires resourcing, though there are resourcing implications of all other options also. Any evaluation of these options, and of the status quo, must be made in the knowledge that the C++ Committee doesn't command resources: all contributions to the work of the Committee are voluntary.

The likely outcome of an attempt to stay with the status quo (the Sophia Antipolis schedule) is that we'll get the worst of (1) and (3) – ie, we'll attempt to issue now (shades of (1)), we'll decide the WD is too messy for an October 2009 FCD, and we'll make a late decision to delay by a year (shades of (3)); and this process may even iterate. This would surely be worse than a conscious choice of either (1) or (3) up-front.

Beyond these observations, it isn't straightforward in a paper such as this to rank the options, or to exclude any, or to nominate front-runners. The author's hope is that all options (and any others which might emerge) will be given equally serious consideration, in a context of open discussion, as an aid to reaching consensus at Frankfurt.