Document Number: N3861
Date: 2014-01-20
Authors: Michael Wong
Project: Programming Language C++, EWG, SG5 Transactional Memory
Reply to: Michael Wong <michaelw@ca.ibm.com>

SG5: Transactional Memory (TM) Meeting Minutes 2013/09/09-2014/01/20

Contents

Minutes for 2013/09/09 SG5 Conference Call

Minutes by Victor

> Agenda:
>
> 1. Opening and introductions
>
> 1.1 Roll call of participants
> 1.2 Adopt agenda

Michael Scott: add discussion of abort vs. terminate

> 1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org

APPROVED

> 1.4 Review action items from previous meeting (5 min)
> 1.4.1 Victor, will update the issue on 1.13 Example illustrating function-local static initialization in atomic transactions

> 1.4.2 Jens adjust it with a red pen (adding clause references) and send it to Victor

DONE (and references were added to paper)

> 1.4.3. please motivate lawyers to publish Legal Draft Release wording with original document

DONE

>
> 1.4.4. Considering going to Chicago C++ Standard meeting, Hotel Booking deadline Sept 6, 2014:
> http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3577.pdf

Michael Wong, Jens, Zhihao, Hans, Victor, Tatiana, Torvald will go.

> 2. Main issues (50 min)
>
> 2.1 SG5 TM Post Mortem
> 2.2 SG5 Slide prep for Chicago

One-hour presentation, with more time for Q&A
- probably won't look carefully at wording at this meeting

AI (Victor): make slides, presentation
AI (Jens): help Victor, send out draft of slides

Jens: Touch on everything, but also explain *why* we need all these changes.
- why should transaction-safety be statically checkable

Victor: To what extent should we talk about stuff we dropped
Jens: Probably not, except as back-up slides.
Jens: Edit: Should have one or two slides saying what changed from last time.
Victor: Yes, and we didn't change that much since Bristol
Jens: Yes, only do diffs from Bristol
Jens: I looked at slides from Bristol, and large font is good.

Zhihao: Question about Clause 15, first bullet ("std::terminate is called"): should this be abort?
Jens: We don't know how to call std::terminate within a transaction, but we want to be able to do so. We changed this to abort in the new places, but some regular exception handling specifies that std::terminate is called, and we haven't come to a conclusion about what we should do yet here.
Michael Scott: Unless I'm confused, I know what the behavior I want at implementation-level, but don't know how to describe it at language-level. The thread that will call terminate takes whatever steps it can to stop other threads, and then calls terminate.
Torvald: [something I didn't understand]
Victor: [repeats what Michael said, to make sure he understood]
Hans: Stop other threads, or make sure they don't see any updates.
Mike Spear: [poses some situation I didn't catch]
Michael Scott: We don't need to guarantee good behavior in that case.
Tatiana: What are the guarantees of terminate?
Michael Scott: Can infinite loop, not sure whether it can throw an exception.
[... more discussion I couldn't capture...]

Torvald: Seems like we're running into a problem with terminate that already exists, and we should figure out what is supposed to happen for nontransactional code first, and then figure out what we should do for transactions.

Victor: Can we cause terminating thread to stop other threads?
various people: no.

Zhihao: First bullet can't happen because of limited kinds of exceptions that can cancel on escape.
Jens: std::terminate might be called by exception handling even if the exception is caught within the transaction.

Michael Scott: [missed this]
Hans: abort has the advantage that it can "cut off the power" and leave things in inconsistent state.

Mark Moir: There are some alternatives: we don't need to terminate other threads, only prevent them from executing user code based on having seem some inconsistent state due to canceled transaction.

Zhihao: Why bother canceling, since we are aborting anyway? We could just leave things in an inconsistent state.
Jens: The objection to that approach: the instant it takes to call the abort, other CPUs will continue to execute and so will see the inconsistent state.

Michael: Maybe say that if you called std::terminate within a transaction, instead call abort.
Mike Spear: [missed this]
Jens: Might call terminate in destructor called during unwinding.

Jens: Let's write it up and look at all the options, may be more of a research project.

Mark: If we analyze the cases, we might discover we don't need a general solution. If the code we care about is all in atomic transactions and relaxed transactions that haven't done anything irrevocable, then we have that code under control.

AI: Everyone who has an approach should write an email to the list, then we can dissect this on the list.

> 3. Any other business

Next meeting is at Chicago
- suggest that we convene meeting on Tuesday.

AI (everyone): Send precise dates to Michael Wong
AI (Michael): Set up SG5 wiki for Chicago

> 4. Review
> 4.1 Review and approve resolutions and issues [e.g., changes to SG's working draft]
> 4.2 Review action items (5 min)

SUMMARY OF ACTION ITEMS:
Victor: Make slides, presentation
Jens: Help Victor, send out draft of slides
Michael Wong: Schedule SG5 presentation to EWG for Tuesday or Wednesday; set up SG5 wiki
Everyone: Send approaches to terminate problem to mailing list
Those attending Chicago meeting: send dates to Michael

> 5. Closing process
> 5.1 Establish next agenda
> 5.2 Future meetings: Sept 23 C++Std meeting, Oct 14 teleconference
>
>

> April 29: Post Bristol report (DONE)
> May 13: Discuss memory model wording, atomicity wording (Michael Wong at C++Now) (DONE)
> June 3: Discuss transaction expressions, function transaction blocks, memory model wording and other standardese topics (Michael Wong at OpenMP and Innovate) (DONE)
> June 10: Discuss function transaction blocks, clause 6, clause 15 (Victor away)
> June 24: Discuss Outstanding questions about Victor's minimalist proposal, Review recent TM talks at TRANSACT, ACCU, ADC++, C++Now. Goal of SG5 (Justin at HotPar, Hans at HotPar).
> July 8: Discuss Outstanding questions 5 onwards about Victor's minimalist proposal,
> July 22: Continue discussion of outstanding questions 6, 7
> Aug 5: Continue discussion of outstanding 6, 7. Review recent TM talks at TRANSACT, ACCU, ADC++, C++Now, Hot PAR.
> Aug 19:Discussion of Std writeup. Write up of proposal for Pre-meeting mailing deadline.
> Aug 26: Review Write up
> Sept 9: (Michael Wong at IWOMP, Justin at PACT) will work on presentation for C++ Std meeting
>
> Sept 23: Chicago C++ Std Meeting
>
> 5.3 Adjourn

Minutes for 2013/10/14 SG5 Conference Call

Minutes by Mike Spear

Agenda:

1. Opening and introductions

1.1 Roll call of participants

Attending: Justin, Tatiana, Michael Wong, Victor, Mike Spear, Paul McKenney, Hans, Herb Sutter, Maged

1.2 Adopt agenda

(Unanimously Approved)

1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org

(Unanimously Approved)

1.4 Review action items from previous meeting (5 min)

(N/A)

1.4.1. Considering going to Issaquah C++ Standard meeting, Hotel Booking deadline Jan 20, 2014:
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3707.pdf

Michael Wong: this is going to be an important meeting, we'll go to full plenary at the next meeting, up to 100 people, and the decision about going to a TS will be made here. Note that following meeting after that is in Switzerland. If we can have a TS by Issaquah, we could be approved in 2014. Note there are 8 meetings until Issaquah.

2. Main issues (50 min)

2.1 SG5 TM Feedback from Herb

(Herb arrived late, so we did 2.1 before this. Then Herb arrived and we switched)

(Herb) Overarching thing from Chicago was that the current design conflates two things that should be independent: synchronization is one, convenient way to write transactions in general is the other. Synchronization is when there is shared memory that we have to protect with locks. This is extremely useful to solve, in Herb's opinion. The second point, which he thinks people want to do at the same time, is writing transactions in general (i.e., taking program from one consistent state to the next). Herb has long been an advocate of this. It is useful, and we have some constructs for this (try/catch), but this is different and orthogonal to managing synchronization on shared memory. There are overlaps, and we might want to have both, but we need transactions even in sequential code, whereas we only need synchronization for multithreaded code; and we need synchronization even when we don't need transactions.

Even though "transaction" is in the name, he sees TM as being about synchronization first and foremost. That's why he was pushing for "atomic { }", or "synchronized {}", and if we want transactional semantics too, we should do it orthogonally. If we do it right, it will be good for sequential code too. Encourages we tease them apart, possibly in separate specifications. Otherwise, we don't have hope of doing better than mutexes.

Herb felt that this view was shared by some but not all.

(Maged) Agrees with this separation. Do "relaxed transactions" / "single global lock semantics" fit this bill exactly?

(Herb) Yes.

(Maged) Guarantee of deadlock freedom is only if we don't compose with other locks. (Herb agrees)

(Justin) This allows a lot of implementation flexibility, since we can just use a single lock under the hood. Is that your goal? Is the fundamental idea that we must abstract away the mechanism?

(Herb) Recall that the word "transaction" is overloaded. For synchronization, his ideal has long been for TM to let me write something simple (atomic {}) and it's as if a single global lock covers all the memory that is accessed within that region. It should protect memory accesses, and nothing else. No rollback, no undo, etc. If it was better than try/catch, he'd advocate it everywhere, but that's a different problem. For example, one could imagine "synchronized transaction { ... }".

(Victor) One worry is that Michael Scott and Mark Moir aren't here. Their input would be invaluable. Question about what "atomic" means in this regard. For example, is it a race to access memory inside and outside block at same time.

(Herb) It should be a race. It's unsynchronized!

(Maged) What about atomic variables?

(Herb) They should be allowed. Races are only if the variables are not synchronized in any way.

(Maged) Should synchronized blocks be able to handshake via atomic<> vars?

(Hans) Regarding why they've been combined: one argument is that they aren't quite separable, because failure atomicity must be sure the incomplete state doesn't escape to another thread before the atomic section completes. The usual way is to combine with synchronization property, so that nothing is visible until it is known no exception will be thrown. So you can't probably do exception safety without synchronization.

(Tatiana) (reviewed distinction between relaxed and atomic transaction) One issue is letting the compiler know what can be run efficiently.

(Herb) A similar question in SG1 relates to vector semantics for stuff that is buggy when we use vectors. Question is if we should have annotations to help know if you've got it right. The annotations are viral, of course, and the sense seems to be "no annotations". Expects same sentiment here. But the question has come up often, for many domains, and there will be an informational talk about it at Issaquah to cover alternatives.

(Mike Spear) What about performance expectations of code? Why do we need a construct at all?

(Herb) Would need lots of compiler assist, of course. Would need to teach people that it's magical, elidable, etc.

(Michael Wong) It would still need to be recursive, of course

(Justin) Would it compose with exception safety?

(Victor) Of course, we don't have that now, and may never, so worrying about composability with it seems premature. But even a "magic" lock still doesn't compose with other mutexes.

(Herb) The "all of memory" issue is important, of course. And it seems appealing.

(Victor) But we still have a race if we use it and other locks to protect the same data from different constructs.

(Tatiana) But a mutex carries a lot more than just semantics, when we program with it.

(Victor) If it's just semantics, then we shouldn't think of it in those other ways.

(Herb) Ok, I'm starting to see how it's not a mutex: mutexes are not composable. This is more like a recursive mutex, since it is safe to take repeatedly, but it goes better because you don't have to teach about reentrancy. Maybe reentrant mutex is the right model. But are there the traditional reentrancy problems if I have a nested transaction?

(Hans) From implementation side, it is useful to identify regions in which this sort of block is held, which is hard for a general mutex.

2.2 SG5 TM post mortem

(Michael Wong Speaking / before 2.1, in case Herb arrives late)

Had a major presentation to Evolution group, started with 20 people, grew to ~40. Jens did the presentation, Jens and Victor had prepared the slides. Second presentation at an evening session for people unfamiliar with TM. At Evolution, emphasis was to get feedback from Bjarne, Herb, and other key decision makers. Three major points:

1 - (esp Herb, also Bjarne) Need some replacement constructs for locking, which only does that... don't conflate protection of invariants from synchronization... just composable, safe, but without handling invariants. That should be separate.

2 - Need to add TM to the standard library, or else people will dive into weird programming practices.

3 - (esp Bjarne) People like making their own little corners of C++ for their own features, but we need to be careful not to let TM do that. We might have to give up some aspect that we care about to make the rest general. Michael thinks this applies to invariant preservation. I.e., could we make something else behave like transaction_safe, since transaction_safe is complicated and TM-specific.

Minor points:
Be sure to cover memory new and delete, and a few other non-controversial points.

(Victor)
Agrees with high-level. Big message is that there is concern that we are trying to do too much... exceptions are a challenge. Victor spoke with Herb afterwards, and it seems like Herb favors global elidable lock. But Herb also suggested that we might just want to keep exception safety smaller.

Bjarne's feedback, and Chandler and others feel that the most urgent aspect is handling synchronization to avoid problems with current locking mechanisms.

Herb wants simple syntax (i.e., "atomic { ... }"); Bjarne likewise said disagreements about syntax are often symptoms of not understanding something deeper.

(Michael Wong): Bjarne said names convey exactly what your underlying thing is. If we aren't sure, or if we are conflating underlying meanings, then it is hard to come up with a good name. The guidance appears to be "make sure you have your underlying meaning clear".

3. Any other business

4. Review
4.1 Review and approve resolutions and issues [e.g., changes to SG's working draft]
4.2 Review action items (5 min)

Considering going to Issaquah C++ Standard meeting, Hotel Booking deadline Jan 20, 2014:
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3707.pdf

Next meeting Oct 28th.

(Note: meeting adjourned abruptly)

5. Closing process
5.1 Establish next agenda
5.2 Future meetings: Oct 28 teleconference

Oct 14: Feedback from Herb;
Oct 28: Review Chicago action items
Nov 11:
Nov 25:
Dec 9:
Dec 23: Christmas Eve
Jan 6:
Jan 20: deadline for Issaquah hotel
Feb 3: pre-C++ meeting prep
Feb 10: C++ Std meeting Issaquah

5.3 Adjourn

Minutes for 2013/10/28 SG5 Conference Call

Minutes by Tatiana Shpeisman

Agenda:

1. Opening and introductions

1.1 Roll call of participants

ATTENDING: Hans, Tatiana, Maged, Justin, Torvald, Michael Wong, Victor, Jens, Michael Spear, Michael Scott, Herb, Mark.

1.2 Adopt agenda

APPROVED

1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org

APPROVED

1.4 Review action items from previous meeting (5 min)

N/A

1.4.1. Considering going to Issaquah C++ Standard meeting, Hotel Booking deadline Jan 20, 2014:
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3707.pdf

2. Main issues (50 min)

2.1 SG5 TM Feedback from Herb

Michael Scott: Asks for clarification on Herb's proposal.

Herb: The essence is separation of two concerns – synchronization and transactions/atomicity. The second aspect is the ability to roll back the code due to exception, even for single-threaded code.

Michael Scott. This has been proposed in the past but have not received much attention. In your mind, what happens when an exception reaches the outer boundary of a region that does atomic synchronization?

Herb: The execution just leaves the protected region

Michael Scott. This is what we call commit-on-escape. Is this the same as relaxed transaction?

Michael Scott, Victor: Yes

Jens: No, it's more like atomic transactions with commit-on-escape.

Tatiana: Let me try to summarize the differences between atomic and relaxed transactions. Relaxed transactions have only synchronization aspect, the same as single global lock. Relaxed transactions can contain any code, but might have serialization bottleneck due to unsafe code. Atomic transactions can contain only safe code (that is, the code that can be rolled back), allow for static checking, and can be rolled back when an exception escapes the transaction

Herb: The key problem is synchronization. Transactions is the only solution that we have that is better than locks. Atomic transactions might also be useful but this is orthogonal.

Michael Scott: You might be chasing what you cannot get. To get reasonable performance with simple programming model you need atomic transactions. If we allow anything inside transactions we might not get performance. Another way to think about transactions – if atomic transactions is really what you want then there is a problem that they don't play nicely with legacy code. So relaxed transactions exist to interface with legacy code and to handle rarely executed code paths.

Hans: We are now getting into the issues that are controversial

Herb: If I explain what relaxed transactions are, can people use them and get scalability?

Michael Scott: Not with HTM.

Tatiana: Concurrent execution with HTM may have restrictions that go beyond unsafe code. HTM can abort for many different reasons. It depends on implementation.

Herb: We can just teach people not to include inappropriate code in the transactions

Many people: Static checks have a lot of value.

Maged: Relaxed transactions is a lower hanging fruit. I am worried about the complexity added by atomic transactions.

Victor: I think Herb said that static checking is nice but not crucial. Consider several possible designs. 1) we have synchronized blocks (aka relaxed transactions) and a code that cannot be rolled back is a race. 2) Unsafe actions are OK but it's a performance problem. …

Herb: Static annotations came up within the context of other proposals too. So far, people prefer just telling programmers not to do it. In C++, we trust the programmer. On the other hand, there was non-zero support for some checking. Static checking has value – for example, AMP C++ has it. Considering static checks is a valid question but it could be a more general conversation.

Torvald: It'd be useful to split Herb's proposal into two parts – split atomic transactions and their cancel and escape flavor. We'd have just relaxed transactions and commit-on-escape atomic transactions.

Victor: Are these three things that we could discuss separately? 1) Relaxed transactions 2) atomic transactions with commit-on-escape 3) cancel-on-escape atomic transactions. I heard Herb saying that he wants just relaxed transactions and everything else can wait.

Michael Wong: Is there a straight line from relaxed transactions to the whole thing?

Justin: Some of us feel that relaxed transactions could be a good first step. Other people feel that it would be a disservice.

Michael Scott: Relaxed transactions by themselves could give transactions bad rep because they won't scale

Michael Spears: I'd only agree with it if relaxed transactions always came with the static checking. In particular, STL is a big concern. Not having annotations on standard libraries could make it very difficult for the programmer. We did memcached – we had to get rid of all serialization bottlenecks – we could not get any performance till we did that.

Justin: Do you think scalability limitations might be tied to the implementation that was used? Do you think relaxed transactions would be more viable if you built everything with transactions?

Michael Spear: First question - serialization is mostly independent of implementation. Second question – I don't think anybody starts from scratch.

Tatiana: Can we define a tier of features that all fit together?

Michael Scott: It's an appealing vision, but if the first starting point is bad it can put away people from TM.

Michael Spear: Three steps – relaxed, relaxed with warnings, atomic with static checking

Michael Wong: This is such an important issue that we need to give everybody a chance to express their opinion.

Jens: I agree with Herb that relaxed transactions approach of shoot yourself in the foot fits with C++, there are a lot of things we need to teach people already. But, we have unique opportunity to help programmers to write exception safe code using atomic transactions with cancel-on-escape. That's a very appealing prospect – HW implementation of roll-back will implicitly do cross-thread synchronization. So, although the aspects might be conceptually different, from the implementation point of view the aspects are dependent.

Maged: The ability to rollback makes a difference in implementation.

Herb: So far, I don't hear any new guidance to programmers than what we already have with locks – don't call unknown code, make it as small as possible. Can I take every published lock-free algorithm and rewrite it in terms of relaxed transactions?

Mark: Some bullet points, more in the e-mail. If we talk about the guidance for relaxed transactions being similar to guidance for locks it's not surprising because relaxed transactions are just locks. And, if we need only relaxed transactions we do not need any language support so we can as well dismiss the group.

Victor: Collaborating Mark's point.

Herb: I might have caused unnecessary confusion. When I talk about rollback being separable, this is for the user. I hope that under the covers we use rollback.

Michael Wong: This was good feedback.

<skipping to item 5.2>

5. Closing process
5.1 Establish next agenda
5.2 Future meetings.

Next meeting will be on November, 11. Anybody for whom it does not work because of US Veteran's Day please let Michael Wong know.

5.3 Adjourn

APPROVED

Minutes for 2013/11/11 SG5 Conference Call

Minutes by Hans Boehm

Agenda:

1. Opening and introductions

1.1 Roll call of participants
** Michael Wong
** Michael Scott
** Michael Spear
** Maged Michael
** Jens Maurer
** Victor Luchangco
** Hans Boehm
** Justin Gottschlich
** Mark Moir

1.2 Adopt agenda

** Unanimous

1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org

** Unanimous

1.4 Review action items from previous meeting (5 min)

1.4.1. Considering going to Issaquah C++ Standard meeting, Hotel Booking deadline Jan 20, 2014:
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3707.pdf

2. Main issues (50 min)

2.2 Proposal for action items based on feedback

Discuss procedures for updating base document.

** Michael Wong: Require concrete papers to modify baseline proposal
** Michael Scott: How formal does a proposal have to be.
** Michael Wong: Can just be a list
** Michael Scott: Does it need to be LaTeXed? Does it need a number?
** Victor: A lot of value to informal discussion. How much overhead?

** Michael Wong Administer numbers within SG5. Publish with numbers.
** Jens: Want email identified by number?
** Michael Wong: yes, e.g. SG5-2
** Jens: seems overly bureacratic
** Michael Scott: Agrees. Just invent unique tag.
** Victor: Encourage distillation into coherent write-up.
** Michael Wong: Consistent with what they had in mind.
** Mark: Already referring people back to specific emails. Need primarily self-contained summaries. People need to consciously summarize/distill.
** Victor: Requires vigilance.
** Jens: Could add links to agenda document

2.1 SG5 TM Chicago Action items
Summary of Feedback from EWG Presentation Wednesday 13:30 - 15:40 hrs
1. user-replaced new and delete (link-time) and calls to set_new_handler()
** Jens: Does a user-define new/delete drop out of transactions? Does this require tm_waiver?
** Justin: new() easier. Delete() harder.
** Michael Spear: use different allocator method to delete?
** Mark: Already handled in current proposal?
** Jens: At least add sentence about forbidding user defined new.
** Michael Wong: Need a proposal that Torvald can see.
** AI: Jens to write initial proposal.

2. rename "cancel_on_escape" and friends; "escape" is a bad name (Clark)
** Victor: Everyone found "escape" confusing. Should use standard terminology, whatever that is. Postpone discussion.

3. overload functions on transaction-safety; might provide a way to make containers in the standard library tx-safe;

** Jens: transaction-safety, part of type, not signature. Can't currently be used for overload resolution.
** Jens: Kind of know whether we're in a transaction, due to safe-by-default. But that's currently under the covers.
** Michael Wong: Combine with 5 and 6?
** Jens: 3 is a longer term issue.
** Conclusion: Group 3,5,6, and 9 as major infrastructure change dependent on Herb's suggestion.
** Justin: What does this have to do with standar container transaction-safety?
** Jens: Allows debug standard library impl, which avoids things like print calls in transactions.

4. provide tx-safe std library containers in the first round (Bjarne)
** All: Should be transaction-safe, just didn't get there yet.

** Justin - Someone has done something along these lines.
** AI: Justin to track this down.
** Michael Scott: std::string should be included.
** Michael Spear: Also consider libc.
** Jens: Partially already covered libc.
** Victor: Make transaction safe wherever possible.

5. create tx-safety region akin extern "C"

6. have constexpr imply transaction-safe?

** Some interest in this. Might help.

7. integrate more (Bjarne)
8. exception escape behavior
9. spelling: "atomic { ... }"

10. splicing the concepts right to achieve the right names; "relaxed transactions"? (Though the example given by Tony was actually confusion about "atomic transaction"!)

** AI: Victor to generate a proposal.

3. Any other business

4. Review
4.1 Review and approve resolutions and issues [e.g., changes to SG's working draft]
4.2 Review action items (5 min)

5. Closing process
5.1 Establish next agenda

** Michael Wong: Discuss "splicing concepts" proposal asap.

5.2 Future meetings: Nov 25 teleconference

Oct 14: Feedback from Herb;
Oct 28: Continue feedback from Herb. Review Chicago action items
Nov 11; Proposal for action items
Nov 25:
Dec 9:
Dec 23: Christmas Eve? No Call.
Jan 6:
Jan 20: deadline for Issaquah hotel
Feb 3: pre-C++ meeting prep
Feb 10: C++ Std meeting Issaquah

5.3 Adjourn

** Unanimous

Minutes for 2013/11/25 SG5 Conference Call

Minutes by Maged Michael

Agenda:

1. Opening and introductions

1.1 Roll call of participants

Attendees: **Maged, Michael Scott, Michael Wong, Victor, Tatiana, Mark, Hans, Michael Spear, Torvald**

Secretary rota: **Justin, Michael Scott, Mark, Torvald, Michael Wong, Victor, Jens Maurer, Mike Spear, Tatiana, Hans, Maged**

1.2 Adopt agenda

Adopted

1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org

Approved

1.4 Review action items from previous meeting (5 min)

1.4.1 Jens: user-replaced new and delete (link-time) and calls to set_new_handler()
https://groups.google.com/a/isocpp.org/forum/?hl=en&fromgroups#!topic/tm/n2liuc2sVg8

DONE

1.4.2. Justin: track down tx-safe container paper

spent about an hour or so looking for the website I had previously seen, but was unable to find it. I will do some more digging, but it may be that the project is now defunct and is no longer reachable.

CONTINUE

1.4.3. Victor: splicing the concepts right to achieve the right names proposal
Victor is making progress. He expects to have a draft soon.

CONTINUE

1.4.4. Considering going to Issaquah C++ Standard meeting, Hotel Booking deadline Jan 20, 2014:
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3707.pdf

CONTINUE

2. Main issues (50 min)

2.1. Review Jen's proposal:
https://groups.google.com/a/isocpp.org/forum/?hl=en&fromgroups#!topic/tm/n2liuc2sVg8

Hans: Not convinced the prohibition of user-defined new and delete is needed

Mark: Jens' point is that this is for now but later user-defined new and delete may be allowed.

Hans: Why specifically new and delete

Torvald: Need constraints on allocation behavior

Tatiana: New and delete are not safe inside an atomic transaction unless the user provides safe new and delete.

Torvald: Memory allocation is separate of the rest of the program

Tatiana: What is a user-defined allocator has side effects

Michael Scott: The runtime system may call new and delete in circumstances more than called for in the user code

Torvald: Yes. The compiler cannot assume that user-defined new/delete are safe

Hans: You need to delay deallocations.

Hans: We need a statement that the standard new/delete are transaction safe.

Maged: We don't need a prohibition if we say that standard new/delete are safe and a user-defined allocator happens to be safe

Hans: It is not clear why we need specific prohibition for new/delete

Michael Wong: This is based on a question by Chandler

Michael Wong: Let's wait for more responses from other

2.2 Michael Scott proposes discussing splicing the concepts.

(I apologize in advance for missing some important points)

Michael Scott: Can't see a clear path separating speculation from atomicity

Victor: I responded to your email

MLS: I sent an email 30 min before the call

Victor: Not advocating separating speculation and atomicity but he can see how they can be separated. Speculation can commit without caring about being atomic.

MLS: Maybe it is OK for other states to see my speculation non-atomically. But half atomicity is needed that speculation does not observe inconsistent state.

Victor: There are some uses like requested by Gregg Stefan that speculation without atomicity

Maged: Sometimes atomicity is provided by other means or not needed because the data is private. I would like to see three options: speculation only, atomicity only, and a combination of both only when needed.

Tatiana: Herb was suggesting this separation between speculation and isolation

Victor: Speculation is different from failure atomicity.

Tatiana: I meant failure-atomicity. Herb's proposal is to separate failure-atomicity from synchronization and provide the user with the means to combine them.

MLS: I understand the difference between the two but does the difference matter to language constructs.

Victor: Speculation can be seen piecemeal by other threads but failure-atomicity you want other threads to see your updates atomically.

MLS: Not sure the distinction is necessary. Speculation requires half of isolation (the speculative section sees consistent state). Other threads may see intermediate speculative state.

Tatiana: If speculation can observe inconsistent state then there is a data race and all bests are off.

Victor: Not necessarily. Maybe there are locks.

Tatiana: The original proposal was the ability to speculate and then decide not to commit

Victor: For a program with interaction with other threads speculation has a stronger notion that failure-atomicity.

3. Any other business

4. Review
4.1 Review and approve resolutions and issues [e.g., changes to SG's working draft]
4.2 Review action items (5 min)

5. Closing process
5.1 Establish next agenda
5.2 Future meetings: Dec 9 teleconference

Minutes for 2013/12/09 SG5 Conference Call

Minutes by Justin Gottschlich

Agenda:

1. Opening and introductions

1.1 Roll call of participants

Hans Boehm

Justin Gottschlich

Victor Luchangco

Paul McKenney

Maged Michael

Mark Moir

Torvald Riegel

Michael Scott

Tatiana Shpeisman

Michael Spear

Michael Wong

1.2 Adopt agenda
Slight modification, approved.

1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org
Approved.

1.4 Review action items from previous meeting (5 min)

1.4.1. Justin: track down STL containers implementation using TM.
Couldn't find it; removed.

1.4.2. Victor: splicing the concepts right to achieve the right names proposal
Done.

1.4.3. Considering going to Issaquah C++ Standard meeting, Hotel Booking deadline Jan 20, 2014: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3707.pdf
Ongoing.

2. Main issues (50 min)

(Swapped to first) 2.1 Continue discussing the Separation of concerns issue. Victor currently working on AI. Others are encouraged to also write it up.

Begin discussion.

Victor: Proposal splices old proposal into two parts: relaxed transactions -> synchronized blocks, clean piece separate from atomic transactions. Leave all interactions with atomic transactions and synchronized blocks in another proposal.

Victor: There are currently four key issues to discuss –

1. Unsafe code: whether or not code should be considered unsafe in synchronized blocks? Certain kinds of code cannot be rolled back.

2. Performance: will there be too much synchronization which will degrade performance? One possible solution is synchronized blocks with domains.

3. Support for all parallelism constructs: do we need support for condition synchronization with synchronization blocks?

4. Handlers: should we add support for commit handlers, abort handlers, etc.?

Mike Spear: What about C++ atomics?

Michael Scott: We should support all basic forms of parallelism and state that performance may degrade based on usage.

Victor: Right.

Justin: Does this mean we should support condition synchronization?

Victor: Well, the current construct doesn't really provide such a mechanism because we have no way to bind the synchronization block to such a condition variables / mutex. Mike Spear has written something about condition synchronization.

Paul: I've read Mike's paper and it still seems somewhat complex. Perhaps the first pass is to proceed without it.

Tatiana: I have big concerns with proposal as a whole. As I understand it, we have renamed relaxed transactions to synchronization blocks and we have dropped everything else. I would like to understand why we have taken this direction. Does this mean the C++ committee has shot down everything else? Also, if we do accept it, what kind of damage do we do to TM by introducing this? Why is synchronized block the best option today?

Paul: I'm not sure why this is considered a nuclear option. Were relaxed transactions better than synchronized blocks?

Tatiana: The prior proposal included three options. 1. Relaxed transactions. 2. Atomic transactions with transaction safety. 3. Transactions with cancel to achieve failure atomicity. The new proposal does not include how atomic transactions will work with relaxed transactions. Do we really need to drop atomic transactions?

Mark: Agrees with Tatiana. Synchronized blocks are pretty much relaxed transactions. The solution with relaxed transactions pretty much doesn't require any language support. Atomic transactions are needed for performance. I sort of feel like we are starting to get to the point where atomic transactions are being dropped and that makes me, and I think most others, pretty nervous.

Paul: Why wouldn't that give you comfort knowing atomic transactions will be able to improve performance?

Mark: I don't know that we need language support for synchronized blocks.

Michael Scott: Couldn't we just have a magic global lock and then all the problems with condition synchronization go away?

Hans: I think this proposal does allow us the potential for speculative execution. Also, Mike Spear convinced me that in order to use relaxed transactions we need to have a better understanding of << sorry, I missed your point here, Hans >>

Michael Scott: Who is arguing for this new proposal?

Mike Spear: I'll take the bait. We could produce something like "synchronized {}" and then we could later add "synchronized transaction {}".

Tatiana: I understand Mike Spear's point, but perhaps our job is to identify the complete solution and then let the C++ committee decide what to do with it.

Michael Scott: It sounds like we have growing consensus to combine both synchronized blocks and atomic transactions in the same proposal.

Tatiana: As I understand it, if what we want conceptually is synchronized blocks then we will serialize many programs. We need some mechanism to improve performance to identify performance bottlenecks, as Mike Spear has demonstrated.

Paul: If you can have bit-sized pieces it might be easier to get accepted by the committee. But, I won't block this proposal if we decide to put both synchronized blocks and atomic transactions in it.

Hans: As Tatiana pointed out, this is destined for technical specification. So there is nothing that prevents people from piecemealing it.

Mark: Almost exactly what I was going to say. The way that Victor has split this out, people may be able to more easily see how to separate out the pieces that they want to implement.

Maged: I support that also and what Hans said. However, I do think the splitting is important and helps people understand things more clearly. I think it's important that we don't throw away atomic transactions.

Torvald: A lot of agreement with what was said already. It seems nobody wants to drop atomic transactions. If the goal is to get things accepted piece-by-piece, does it make sense to break things into two pieces?

Mark: Perhaps one way to handle this is to have one proposal. Perhaps have the first part mentions just a stand-alone piece (synchronized blocks) and then the second part includes (atomic) transactions.

Victor: Okay, so I propose I do what Mark just suggested, where I start the first part of the proposal that includes the synchronized block proposal and then the second part includes atomic transactions.

AI: Victor to write-up modified proposal to include both synchronized blocks and atomic transactions.

Victor: One reason why I split the proposal out into two pieces is because I don't think the C++ committee will accept a proposal that includes both synchronized blocks and (atomic) transactions.

Mark: I appreciate you volunteering and that what you have done already is important. 1. I think it's worth reconsidering reorganizing the previous proposal. 2. We should definitely get Jens on board so he can understand what changes are necessary to the actual C++ Standards document.

Michael Wong: Jens does need to be onboard, however, if we decide that we need to alter the proposal again, then we probably will need him to rewrite the standards modifications again. I have also seen that we can have the proposal accepted in pieces like "part A" or "part A and B" or "part A and part C" and so forth.

Victor: How do people feel about static checking for transaction safety? We would specify that there are unsafe operations but not include all the extra stuff for transaction safety annotations.

Mark: I have tried to come around to see the benefit of run-time checking of transaction safety, but I still feel that compile-time and link-time checking of transaction safety is the right path.

Mike Spear: Transaction callable is still important. If you like synchronized blocks, then you essentially should support things for explicit transaction safety.

Hans: It seems that much of the transaction safety restriction is based on how the standard library behaves.

Tatiana: I think you'll throw away a lot of useful work if we eliminate explicit annotations for transaction safety.

Victor: Okay. The general consensus seems to be that we include explicit transaction safety.

Michael Wong: Is it possible to tone down transaction safety to an attribute?

Victor: I don't think so because transaction safety needs to be part of the type system.

Mark: I think we are already trying to nailing down too many moving parts.

End discussion.

AI: Everyone to read Victor's new proposal.

3. Any other business

4. Review
4.1 Review and approve resolutions and issues [e.g., changes to SG's working draft]
4.2 Review action items (5 min)

AI: Michael Wong to find someone who can lead the effort to identify what is necessary to modify the standard library to support our TM proposal. Michael Spear has volunteered to help that effort, but not to lead it.

AI: Victor to write-up modified proposal to include both synchronized blocks and atomic transactions.

AI: Everyone to read Victor's new proposal.

5. Closing process

Next meeting: January 6. Mailing deadline for next C++ meeting is January 17.

Jan 6: Mailing deadline is Friday, Jan 17
Jan 20: deadline for Issaquah hotel
Feb 3: pre-C++ meeting prep
Feb 10: C++ Std meeting Issaquah

5.3 Adjourn

Minutes for 2014/01/06 SG5 Conference Call

Minutes by Michael Scott

> Agenda:
>
> 1. Opening and introductions
>
> 1.1 Roll call of participants

Hans Boehm, Justin Gottschlich, Victor Luchangco, Maged Michael, Mark Moir, Torvald Riegel, Michael Scott, Tatiana Shpeisman, Mike Spear, Michael Wong.

> 1.2 Adopt agenda

Swapping 2.2 and 2.3 to accommodate Victor, who had to join a few minutes late.

> 1.3 Approve minutes from previous meeting, and approve publishing
> previously approved minutes to ISOCPP.org

Done.

> 1.4 Review action items from previous meeting (5 min)
>
> 1.4.1. Victor: splicing the concepts right to achieve the right names
> proposal

Ongoing; see below.

> 1.4.2. Considering going to Issaquah C++ Standard meeting, Hotel Booking deadline Jan 20, 2014:
> http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3707.pdf

> 2. Main issues (50 min)
>
> 2.1 Add call for next Monday Jan 13 proposal

Mailing deadline for Issaquah is 17 Jan., or shortly thereafter.
Adding an extra conf. call next week to accommodate.

> 2.3. Transactionalizing STL

Mike Spear taking point (so to speak :-)
Looked at list<> template. What would it take to make it txn-safe?

Issues encountered:

Size() method required to take constant time in C++'11.
So every op needs to modify a counter. Would seem to imply a lot of aborts.
Justin: perhaps we should push back on guarantee of constant time in a txn context?
Mike Spear: don't think it helps: txns compromise the ability of subsequent non-txnal uses to be constant-time.
Michael Scott: perhaps size() could be log-time?
Mark: note that we're taking a non-scalable abstration and worrying about the fact that we can't make it scale. Maybe we need different abstractions for scalable programs.
Hans: probably the only option in the long run.

Size() is also const noexcept
If called from STM, need to allocate new entry in read set, which might run out of memory.
This is a general problem, for multiple functions; prob. means we have to give up on guaranteed progress.
Michael Wong: lobby committee to get rid of noexcept?
Hans: real problem is not the declaration, but the fact that the TM impl may throw unexpected exceptions. Not convinced that one can cleanly catch all out-of-memory conditions in any event.
Michael Scott: can we hope to do all TM memory allocation "outside" -- at allowable points in the code?
Mark: maybe abort, pre-allocate a larger amount, and start over?
Hans: would have to explicitly allow txn_begin to throw "out of memory", which might cause its own problems.
Mike Spear: worst case, can't we go inevitable? (Doesn't handle self-abort.)
Justin: Self-abort might still be possible, at least with certain programmer discipline.
Hans: But you can't get around the fact that stack is limited. Out-of-memory is a deep, fundamental problem.
Michael Scott: perhaps we need to highlight the larger issue, which may need addressing in the standard more generally.
Mike Spear: and perhaps add a note that pre-allocation of metadata may be wise in an implementation that wants to reduce the likelihood of problems.

templated fns -- Should these be annotated as txn-safe?
"Yes" becomes problematic if instantiated for certain types;
"No" becomes problematic if need to use inside other txns.

Would seem to lend weight to arguments for safe-by-default and against annotations.
Torvald: gcc currently does safe-by-default w/in each compilation unit.
Mark: i.e., "implicitly declared safe".
Mike Spear: doesn't seem to quite work that way in his experience... Bug? Runs into problems in List_Base.
Mike and Torvald will explore off-line.
Justin: work with Hans on templates & TM encountered trouble trying to figure out which things needed to be relaxed txns, and which could be atomic.

Justin: maybe we need a two-step process: (1) do the best we can with the current STL; (2) work on better, future, STLs.
Michael Wong: people are working on better, scalable, STLs.
Hans: those aren't nec. exactly what we'd need: not considering transactions.
Mike Spear: if interested in getting involved, all sources of his work are on github; will post the link.

> 2.2 Continue discussing the Separation of concerns issue. Victor
> currently working on AI. Others are encouraged to also write it up.
> https://groups.google.com/a/isocpp.org/forum/?hl=en&fromgroups#!topic/tm/r6YHQ-Uqz08

Victor sent around an updated draft during the meeting.
No major changes from last time (reorganized some of the content).
Needs an intro that explains how it all fits into the larger context.
One medium change: synchronized blocks no longer automatically transaction-unsafe (though of course some of the stuff you can put in them might be).
Transactions remain truly atomic / isolated; did not try to separate out just "failure atomicity".
So the new proposal is basically a renaming of "transaction relaxed" and "transaction atomic" to be "synchronized" and "transaction".

Torvald: suppose we allowed locks inside transactions. Doubt it would still be ok to say that synchronized blocks are transaction safe. Victor: think it _would_ be ok.
Mark: synchronized blocks would behave differently inside and outside transactions. Would still work, but maybe confusing?
Michael Scott: if your synchronized block handshakes with another thread, it won't work inside a transaction, but that makes perfect sense...
Victor: right.
Torvald: maybe...

<to be continued>

> 2.4 Continue discussing user replaced new and delete
> Original:
https://groups.google.com/a/isocpp.org/forum/?hl=en&fromgroups#!topic/tm/n2liuc2sVg8
> More discussion:
https://groups.google.com/a/isocpp.org/forum/?hl=en&fromgroups#!topic/tm/WUTH7ETBprk

Deferred.

> 3. Any other business

> 4. Review

> 4.1 Review and approve resolutions and issues [e.g., changes to SG's working draft]

> 4.2 Review action items (5 min)

- Mike Spear to post link to STL work.
- Mike Spear and Torvald to discuss gcc list_base operator= issue.
- Group as a whole to continue on-line discussion of Victor's proposal.
- Michael Wong to send notice of next week's meeting time.

> 5. Closing process
> 5.1 Establish next agenda
> 5.2 Future meetings:

Jan 13?: extra call for Issaquah mailing prep
Maybe one hour late, to accommodate Victor and Mark;
watch the reflector for final notice.
- hide quoted text -

Jan 20: deadline for Issaquah hotel
Feb 3: pre-C++ meeting prep
Feb 10: C++ Std meeting Issaquah

> 5.3 Adjourn

Minutes for 2014/01/13 SG5 Conference Call

Minutes by Mark Moir

Agenda:

1. Opening and introductions

1.1 Roll call of participants

=> Attendees: Jens, Mark, Michaels (Maged, Scott, Spear, Wong), Victor.

1.2 Adopt agenda

=> DONE

1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org

=> DONE

1.4 Review action items from previous meeting (5 min)

1.4.1. Mike Spear to post link to STL work. - DONE

https://groups.google.com/a/isocpp.org/forum/?hl=en&fromgroups#!topic/tm/W1PBkMJLGJ4

1.4.2. Mike Spear and Torvald to discuss gcc list_base operator= issue.

=> DONE. Wait until a few more collections are converted, then decide on concrete recommentations. Options: move all code into header, or mark declarations as txsafe, not sure which will be more palatable.

1.4.3. Michael Wong to send notice of next week's meeting time. DONE

1.4.4. Victor: splicing the concepts right to achieve the right names proposal

=> DONE.

Group as a whole to continue on-line discussion of Victor's proposal.

=> ONGOING.

https://groups.google.com/a/isocpp.org/forum/?hl=en&fromgroups#!topic/tm/r6YHQ-Uqz08

1.4.5. Considering going to Issaquah C++ Standard meeting, Hotel Booking deadline Jan 20, 2014:
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3707.pdf

2. Main issues (50 min)

2.1. Transactionalizing STL

https://groups.google.com/a/isocpp.org/forum/?hl=en&fromgroups#!topic/tm/W1PBkMJLGJ4

=> Michael Wong has sent out numbers for two papers:
- Transactionalizing STL
- Revised proposal

Throw bad alloc and built-in abort need to be txsafe.
Things like assert should be txsafe.

Michael W: Agree, next step is to write paper (3-4 pages) about it, give rationale, discuss hurdles. Can add more containers later.

AI: Michael W and Mike S to draft paper.

Michael S: what about issue of whether existing API can be made to scale? To what extent are we trying to make scalable versions of what's there and to what extent should new APIs be proposed that can be implemented in scalable ways.

Mike S: Initial position and longer term view to add more. What we're doing would make a staple, e.g., txl list allows collection of txl lists.

Michael W: analogy to STAPL (https://parasol.tamu.edu/stapl/).

Michael S: So initial step is to at least make existing funtionality useable for low-contention cases?

MW/Mike S: yes.

MW: Issue not closed, but want to get a sense on initial positions. MW will present. Maybe MS can call in.

AI: MW to send github repo to MS.

2.2 Continue discussing the Separation of concerns issue. Victor currently working on AI. Others are encouraged to also write it up.

https://groups.google.com/a/isocpp.org/forum/?hl=en&fromgroups#!topic/tm/r6YHQ-Uqz08

=> Victor: no substantive change except allowing synchronized blocks within (atomic) transactions, just changing exposition to make clear that synchronized blocks can exist independently of transactions.

Michael S: Terminology question. Hope for final names for transactions. Could we call transactions "atomic" blocks?

Jens: yes, question is whether sufficiently unambiguous in grammar.

Michael S: If so, is it desirable?

Jens: Seems likely that it is sufficiently unambiguous.

Mark: I always thought so, but was shouted down before due to potential confusion with C++ atomics.

Victor: Herb said possible and desirable, I am ambivalent. Worry that it might be confusing with atomics, but some might consider that good.

Jens: I agree too, if you squint the right way it makes sense. We've had feedback that we should improve keywords. Changing to atomic would be fine.

Michael S: Can we make a concrete decision?

Mark: Others who are not here may have concerns, e.g., some people think atomic should mean "at the bit level atomic", not "programmer thinks as if atomic".

AI: Michael Wong to send out note and make sure people have a chance to respond.

Mike S: One concern, future "on commit" funtionality might not make sense if "transaction" connotation is lost.

Mark: More immediately, things like "commit_on_escape", but people didn't like
"escape", maybe we should fix everything together?

Jens: there are a few bullets to bite for cancel_on_escape:
- doesn't work with all exceptions
- are we comfortable with current limited position
- e.g., should there be a fcility for programmer to extend functionality

Victor: Great idea, but no time for this round.

Victor: At Issaquah, people are likely to want to remove atomic blocks/transactions.

Mark: We need to make clear that we are trying to structure the proposal so that easier pieces are independent and can be adopted earlier.

MW: How should I handle this?

Jens: We should not suggest splitting, if question comes up, discuss what there is to lose if we don't split? Answer should be nothing.

Maged: Keep implementors aware of what else may be coming, so they don't close off implementation options inadvertently.

MW: This is a Technical Specification, implementors can choose to implement a subset.

Jens: Are you in contact with Bjarne to discuss next iteration?

AI: Michael Wong to let Bjarne know we want a slot (perhaps one each for transactionalizing STL and proposal).

Some discussion of relationship between the two.

Jens: We don't need a large library evolution group.

MW/Jens/Victor: Agree we don't need to update wording changes for papers for this round.

Jens: Perhaps try to have it ready for meeting in case people want to see it.

MW: Any further questions, discussions?

None heard.

AI: Victor to send out updated document with all pieces present, keyword changes, though some discussion may need to be reworked.

MW: What about characterising responses to previous feedback?

AI: Victor will draft material. Mark and Jens agree it should come first.

Jens: What about replacing "escape" wording?

Michael S: No objection to "noexcept", we've already bitten the bullet on being imprecise, maybe extend it? cancel_except and commit_except would be parallel to noexcept.

Discussion of whether to include underscores. Tradeoff between consistency and cringeworthiness. Keep them in for now.

Jens: Is commit_except sufficiently different from synchronized block to warrant its existence?

Mark: Yes, see old arguments about atomic vs. relaxed. (Synchronized block does not document/enforce programmer's intention that the block is atomic; may also affect semantics in future if locks/atomics are allowed in atomic blocks.)

Jens: Recent proposal said compiler-generated constructor is implicitly txsafe if all constructors are txsafe, what about tx-safety of members' constructors? Those should contribute to txsafety. Need to get this right in formal wording, ok if informal not precisely right.

2.3 Continue discussing user replaced new and delete
Original:
https://groups.google.com/a/isocpp.org/forum/?hl=en&fromgroups#!topic/tm/n2liuc2sVg8

More discussion:
https://groups.google.com/a/isocpp.org/forum/?hl=en&fromgroups#!topic/tm/WUTH7ETBprk

Jens: Most conservative way is to prohibit user-replaced new and delete for now. Design space that needs to be explored.

MW: We should write this up in main proposal.

Victor: I am not qualified to write this.

AI: Jens to provide paragraph for document (and to make up for inadequacy of meeting minutes :-)).

3. Any other business

4. Review
4.1 Review and approve resolutions and issues [e.g., changes to SG's working draft]
4.2 Review action items (5 min)

=> Summary of AIs from above:

AI: Michael W and Mike S to draft paper on transactionalizing STL
AI: Michael W to send github repo to Mike Spear
AI: Michael Wong to send out note about changing transaction to atomic and make sure people have a chance to respond. (DONE)
AI: Michael Wong to let Bjarne know we want a slot (perhaps one each for transactionalizing STL and proposal).
AI: Victor to send out updated proposal document. To include discussion of how we're responding to previous feedback at beginning.
AI: Jens to provide paragraph for Victor's document addressing issues related to user-replaced new and delete.o

5. Closing process
5.1 Establish next agenda
5.2 Future meetings: Jan 20 teleconference

Keep next week's meeting.

Oct 14: Feedback from Herb;
Oct 28: Continue feedback from Herb. Review Chicago action items
Nov 11; Proposal for action items
Nov 25: Reviewed Jens user new and delete proposal
Dec 9: Review Victor's proposal
Dec 23: No Call.
Jan 6: Reviewed transactionalizing STL std:list. Mailing deadline is Monday, Jan 20
Jan 13: Continue discussion transactionalizing STL, spliting concerns proposal. Writeup
Jan 20: deadline for Issaquah hotel, mailing deadline
Feb 3: pre-C++ meeting prep
Feb 10: C++ Std meeting Issaquah

5.3 Adjourn

DONE.