# Charm++ - Cleanup #1997

## Improve coverage of tests for ck.C

10/18/2018 01:12 PM - Eric Bohm

<table>
<thead>
<tr>
<th>Status:</th>
<th>New</th>
<th>Start date:</th>
<th>10/18/2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
<td>Date:</td>
<td></td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
<td>Due date:</td>
<td></td>
</tr>
<tr>
<td>Category:</td>
<td>Build &amp; Test Automation</td>
<td>% Done:</td>
<td>0%</td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
<td>Estimated time:</td>
<td>0.00 hour</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Spent time:</td>
<td>0.00 hour</td>
</tr>
</tbody>
</table>

### Description

ck.C has 46% function coverage from the current tests and examples in make test.

46% looks bad, but when the details are examined the uncovered code is all at the edges.

1. **CHARM_DEBUG** or other introspection infrastructure only used by the debugger
   1. given the gui nature of charmdebug it is outside the scope of this task to include that in make test.
2. **ERROR HANDLING**
   1. e.g., buffering immediate messages for nodegroups that don't exist yet. or detecting message resends
3. **default implementations**
   1. e.g., delegation default implementation that nothing ever uses because if you're using delegation you want it to actually do something
4. **unused features**
   1. e.g., cklocalchare, we don't have examples use singleton chares in performance sensitive ways that would lead someone to use this.
5. **_TRACE_BEGIN_APPWORK**
6. **fault tolerance and/or evacuation code and/or destruction code**
   1. we don't have tests for all of this infrastructure
7. **ext**
   1. the infrastructure for Charmpy isn't tested by the usual path through make test
8. **record replay**
   1. CkMessageReplay is not tested
   2. bigsim related sections not tested

Charm debug and introspection code probably requires a non-gui unit test framework of its own. We'll make a separate task for that at some point.

A substantial chunk of the total uncovered code is in the ext interface for charmpy and in record/replay. So those are the low hanging fruit.

The untested features should be handled by implementing new tests.

The never to be used default implementations should be given another look to see if we can implement it in a way that doesn't require never used code. If not, then we should tag that code to be outside of coverage because we don't care.

The exception case code should be instrumented with gcov exclusion because we're not going to try to write code to create bugs to test how we handle bugs until some more distant future.

### however, a suite of

### Subtasks:

- Cleanup # 1998: integrate charmpy tests into coverage **New**
- Cleanup # 1999: evaluate record/replay in make test **New**
- Cleanup # 2000: debug and introspection API coverage **New**