Coding Instructions for Topic Segmentation of the ICSI Meeting Corpus

(version 1.2)

Weiqun Xu*, Jean Carletta and Jonathan Kilgour

School of Informatics, University of Edinburgh
Email: {wxu,jeanc,jonathan}@inf.ed.ac.uk

July 21, 2004

ABSTRACT

This paper is a manual that instructs annotator how to work on topic segmentation in the ICSI meeting corpus. After introducing some general ideas of the task and the tool, it explains how to do the job step by step. Some advice is further given for external users.

Contents

*Contact person. All kinds of comment and feedback are warmly welcome.

1 Introduction

The ICSI Meeting Corpus is a collection of recordings of meetings from university research groups at ICSI, in California. Many people study these meetings because one of the big technology challenges at the moment is to build a meeting browser — that is, something that can be used to find out what happened at a meeting. A big part of meeting browsing is knowing what the people in a meeting were talking about — the topic — and when they changed topics. It's easier to make a machine understand topics and topic changes if someone tells it about topics and topic changes on some examples. Your job is to listen to some meeting recordings, divide the meetings up by topic into "segments", and briefly describe what the topic is for each one. We expect this to take 4-5 hours for new coders (but 2-3 hours for experienced coders) for every hour of recorded meeting.

2 The Task

2.1 Segmentation

You might be expecting us to tell you exactly how many topic segments each meeting should have, but the truth is, it varies considerably. We have in mind that a typical one-hour meeting might have something in the order of six to ten segments, but there could be meetings that discuss one topic extensively, and others that handle a very large number of topics, all briefly. For this reason, you should divide the meeting into segments in the way that you find most natural.

Everything that is said during a meeting should end up in some segment, but since some material in meetings isn't really about a topic we provide some standard kinds of segments to make this easier.

2.1.1 Sub-topics

Sometimes you won't be sure whether to mark part of a meeting as one segment or two, because there are really two segments but they are related to each other. For instance, if a group were talking about what they liked about Edinburgh, they might talk first about the free museums and then they might talk about the architecture. If they talked about these two things completely separately, without saying anything about why the two go together, then they would be separate topics. However, if they made clear that there were some connection, for instance, by saying that they were talking about what they liked about Edinburgh and then introducing these themes, the overall segment would be about "good things about Edinburgh", with the sub-segments about "free museums" and "architecture". You can mark sub-segments wherever you like, to cover part of the material in a segment.

If you feel that there is a clear subtopic happening, then mark it and describe it. Otherwise, don't bother to subdivide. In theory every time someone talks they're saying something different from the last person and therefore it should be possible to mark a new subtopic, but we don't want this level of detail. A subsegment should be something that the group is discussing, not just something one person threw into the discussion. We would expect some sub-topics in the meeting corpus, and possibly (but rarely) some sub-sub-topics, but if you

find highly nested structures (with lots of detailed sub-sub-topics), you should consider whether you might be subdividing topics too finely.

2.1.2 Hints for finding segment boundaries

There are some clues that should help you find segment boundaries.

The first is that people in meetings quite often announce topic changes. For example,

> *Ok, so the s the the next thing we had on the agenda was something about alignment* [BMR019-740]

You should be careful that the meeting actually moves on to the next topic at this point — sometimes someone will intervene with more material on the previous topic — but otherwise such utterances are pretty good indicators of a boundary. Note that as well as signalling a topic shift, these utterances often give some clear idea about the topic which you can put into topic description.

The second clue is that if the group discusses the agenda at the beginning, it can be useful to look for where the agenda items appear, even if groups don't always follow the agenda that they set. Again, the agenda can be a useful source of topic descriptions.

The last set of clues are words like "anyway" and "so", which can be used to indicate a topic shift, like in the example above. When you listen to the recordings, these indicators can sound quite distinctive from other ways of using the same words.

2.2 Topic Description

As well as saying where the discussion of a topic starts and ends, you need to give a short English description of the topic. This can usually be based on a few keywords from the discussion, and needs to be detailed enough that someone could figure out later on what the topic was, but not so detailed that writing the descriptions is a major part of the work. We have in mind short phrases for these, of perhaps six or seven words. Groups will quite often discuss a topic and then return to it later in the same meeting. When this happens, you should use exactly the same description. The software lets you do this without typing it in again. We'll know which topic a subtopic goes with, so it's OK if it is necessary to read both the topic and subtopic label to understand what the subtopic is (e.g., "why we like Doris" could have "her hairstyle" for a subtopic rather than "hairstyle as a reason for liking Doris").

Sometimes you won't be sure what words to use to describe a topic, especially if there are clear keywords in the text. It's better to provide some description than none, even if you have to just describe the segment in terms of how it contributes to the meeting (such as "clarification of technical issue affecting recognizer performance").

2.2.1 Standard "Topic" Descriptions

We provide five special, standardized topic descriptions to make coding easier and to ensure some consistency across the coded meetings: "digit task" for a peculiar task the participants do involving reading digits aloud; "opening" and

"closing" for the special kinds of things that happen at the beginning and ends of meetings; "chitchat" for social chatter; and "complex" for material that is just too difficult to code.

Digit task

Because the researchers involved in the meetings are themselves studying automatic meeting processing, they sometimes do things during their meetings particular to aid their research. Near the beginning or end of most of the meetings, they perform a task that involves reading a series of digits. This is to help their speech recognizer, and isn't really a part of the meeting business. If they do this task during the meeting you are coding, place these utterances in a segment of their own labelled "digit task".

Opening

The "opening" topic description can be used for all of the little things that groups tend to do at the beginning of a meeting: take attendance, review or set the agenda, say how long the meeting is, and so on. It's useful for us to know where the opening begins and ends, but the opening doesn't contain particular topics, or at least not technical ones, just discussion about how to conduct the meeting. We don't need opening material to be further segmented, except that if the group performs the digit task during the meeting opening, the digit task should be in its own topic segment or sub-segment. It doesn't matter whether you place the digit task within the opening as a sub-topic or next to it, so don't fuss about which structure to use.

Closing

Just as meetings can have openings that aren't about a real topic, they can also be closed in a similar way, for instance, with time dedicated to setting the next meeting date and reviewing who will do what in preparation for the next meeting (or deciding this if it hasn't been done during the meeting itself). Mark these as "closing", and again, do not segment further, except for the digit task.

Chitchat

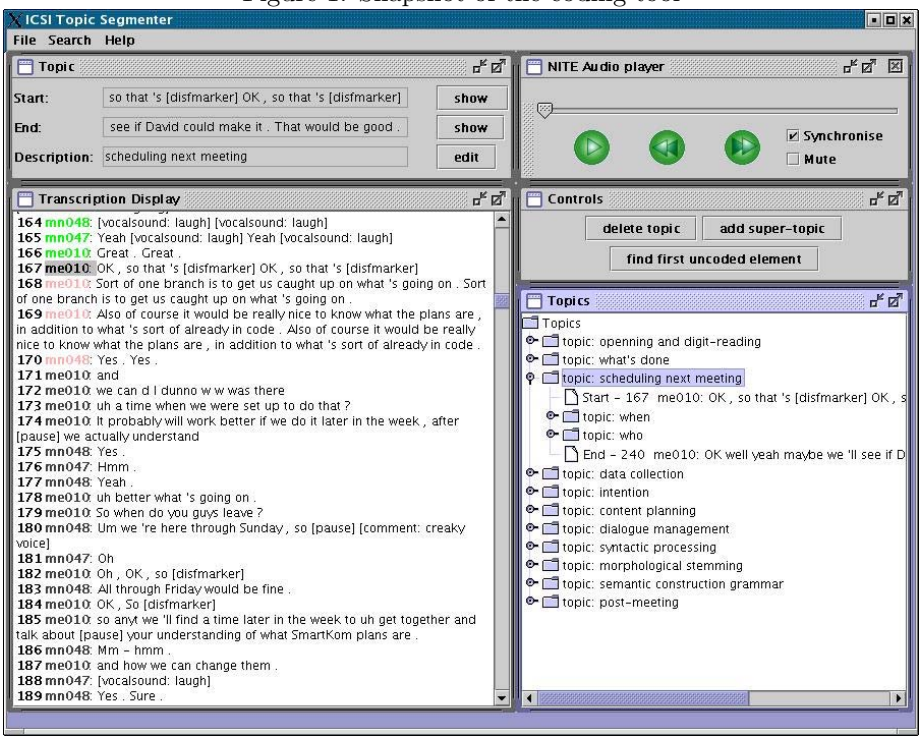
Sometimes during a meeting the participants just chat aimlessly, usually about social matters. This especially happens after the microphones have been switched on but before the beginning of the meeting "proper", and again at the end. It can also happen in the middle of a meeting, for instance, when a projector breaks, or simply if someone drags the group off-topic. When this happens, divide the meeting so that these areas form their own segments, but label them with the special topic description, "chitchat".

It can also happen that while the group is having a proper discussion of some topic, there will be one or two quick utterances, like jokes, that you might be tempted to code as "chitchat", but the group doesn't really get pulled off the topic they are discussing. Don't bother to segment around these cases — just leave the utterances within the wider segment.

Complex

It can be difficult to understand what is happening in these meetings for several reasons. The subject matter is pretty esoteric, and during informal meetings among people who know each other well, they tend to raise whatever issues they want to discuss whenever they spot an opportunity, rather than rigidly following a set agenda. Also, sometimes topics are so intertwined with each other it can be difficult to tell where one ends and another begins, especially if both need to be discussed before some decision can be made.

Figure 1: Snapshot of the coding tool



As long as you can find the boundaries between topics and pick out some keywords to put in the brief topic description, what you do will be OK. Segmenting the topics but failing to describe them is better than nothing. If you really can't make enough sense of some part of the meeting to segment it, then put that part in the special topic description, "complex".

3 The coding tool: ICSI topic segmenter

We've written software specifically for this task that will allow you to view a transcription, play the meeting recording, and segment the meeting and add the topic annotation. (The software comes with on-line help, so you can look at that as well as reading this document.) The coding tool will work on uncoded, coded, or partly coded meetings, so you can stop and restart at any time (but remember to save your work!) or just review meetings you coded earlier.

Opening a meeting usually results in five windows on a common desktop (see figure 1):

TOPIC: which shows some information about the current topic segment, including its START and END and its topic DESCRIPTION. You may quickly move to the corresponding utterances by clicking the SHOW button. You can also change the description by clicking the EDIT button. When you click the EDIT button, a DESCRIBE TOPIC window will pop up, in which you can either *Choose an existing topic* (predefined or previously coded)

from a drop down list or *add a new one* by inputting some free description.

NITE AUDIO PLAYER: which plays the audio of the opened meeting. [1] There are three buttons, two check boxes, and one progress bar.

Progress bar – which shows audio play progress. You may also slide forward or backward to your desired part.

Buttons ▷ = play or ‖ = pause, ▷▷ = fast forward, ◁◁ = fast rewind.

Check boxes

Synchronise when checked (default), audio is synchronised with text highlighted in green. This is very helpful when you browse the meeting for the first time, but might become annoying when you just want to skim the text later. This feature can be disabled by unchecking the box.

Mute turn off the sound while it's playing.

TRANSCRIPTION DISPLAY: which displays the transcription of the opened meeting. Every utterance is preceded by an automatically generated line number and a speaker ID label. The transcription window divides up the meeting by who said what. Beyond that, the way it divides something one person said into numbered lines is fairly meaningless. Most of the time, you will be able to say that an entire utterance belongs to one topic segment or another, placing the boundary between utterances.

If you want to add a topic segment that uses complete numbered lines, left click on the speaker label of the first utterance in the segment and then right click on the speaker label of the last utterance in the segment, which has to be after the first segment and not already in a topic segment. When there is a valid right click, the interface will pop up a window for you to describe its topic. You can do that either by choosing one from the drop-down list or by entering a new description. You can choose to split a numbered line (if you have to) by right clicking on a word instead of a speaker ID to end the topic segment. In the current version of the tool you can't start a topic by left clicking on a word, so you'll have to code the topic segment that covers the first half of the line before you code the one that covers the second half.

If you want to start play the audio from an arbitrary line, select the line, then press `Ctrl` and right click at the same time.

CONTROLS: which provides additional ways of changing the topic coding and moving around the transcript.

Delete Topic: which deletes the topic (i.e., a segment and its topic description) after you click a topic in the TOPICS window. (Note: any sub-structure will not be deleted, instead all the children of the deleted topic will be upgraded one level.) The interface is slow to adjust after you delete a topic, so be patient.

[1] If there are problems with the sound card or the coding tool can not locate the corresponding audio file, the player may not work properly or even not appear. Contact your coordinator if this happens.

Add Super-Topic: which adds a super/parent topic. First, select multiple consecutive topics in the TOPICS window. (You can do this by selecting the first topic in the series with the left mouse button, holding down the *shift* key, and selecting the last topic in the series.) Then click this button to make the selected topics sub-topics of some new parent topic. Of course, you need to give a description of the super-topic.

Find First Uncoded Element: which helps you quickly jump to the first uncoded utterance.

TOPICS: which displays all the coded topics in the meeting. A topic node is represented by its description with its start and end utterances and possible sub-topic(s). You can do nothing in this window but select one or more topic nodes.

4 How to do the work (for local coders)

Once you are sure that you have fully understood these instructions, the first thing to do is run the tool on an example meeting that we've already done. (It should be taken as a reference instead of a standard.) Here you should look carefully at the segmentation and topic description, since this is the easiest way to help you understand what's to be done.

To do the work, you will have to get the meeting data from CVS server, code a meeting, and put your coding back to the CVS server. Here's how to:

First, checkout CVS repository from our CVS server.

```
: mkdir -p ~/ami/cvs
: cd ~/ami/cvs  %this absolute path is noted as $AMI_CVS
% please note, most of the following operations
% are supposed to work under $AMI_CVS
: cvs -d $AMI_CVS init
: export CVSROOT=\
    :pserver:<cvs-username>@cvs.inf.ed.ac.uk:/disk/cvs/ami
: cvs login
% input your cvs password. If nothing wrong, go on
: /group/ltg/projects/NITE/nxt/starttopic -u $cvs-username
% it will take about 5 minutes to checkout the repository
```

Second, start the tool and open a meeting file (coded or uncoded) with this command

```
: sh topic.sh
```

If you are to look at an example coding, put it (including *. topic.xml and *.segs.xml) into your local directory `Data/ICSI/NXT-format/Main` before starting the tool. If you are to do segmentation and describe topic, following the instructions in sections 2 and 3. After you finish (or even if you have not finished), you can save your work by clicking `File/Save Corpus`.

Finally, check in your work to the CVS server (cvs login required):

```
: /group/ltg/projects/NITE/nxt/checkintopic \
    -u <cvs-username> -o <obs>
%  where <obs> is the coded meeting name, like 'Bmr018'
```

[Tips for coding] When you start on a new meeting, we suggest that you begin by listening, using the transcription to skip forward once you have a sense of what's going on. If you're in a room with other people please use headphones. You can code any part of the meeting at any time, but it's best to work in an orderly fashion, from beginning to end. If subtopics feature heavily in the meeting you're coding, you may find it easier to segment the entire meeting and then return to add the subtopics afterwards. You should keep a list of the meetings that you are finished with, as opposed to those that you've partially done, or those that you've segmented but wish to check — the software won't help you do this. If you feel it helpful to take some notes during coding, feel free to use a piece of scratch paper or two. Also, remember to keep track of the hours you spend.

5 Advice for users outside Edinburgh

The instructions are originally prepared for local coders, so there are some presumptions which make it difficult for external users to follow (esp. those in section 4. (But they might still be of some help. The two scripts used above are put in the appendix and can also be found with this document in the CVS repository under `DATA/ICSI/NXT-format/Contributions/TopicSegmentation` But some adaptations are necessary if you want to use them.) Some hints will be given below on how to setup the coding tool on your machine.

First you need to have a successful setup of The NITE XML Toolkit (NXT), which is publicly available through http://sourceforge.net/projects/nite/. For more information, please refer to the relevant documents or visit http://www.ltg.ed.ac.uk/NITE/.

Second the data. You will need the ICSI meeting transcription data in NXT format and the audio data (if your want to listen to). The transcription data are available to AMI project partners and "friends" through AMI CVS repository at the University of Edinburgh. Those who need AMI CVS access should choose "AMI" from the CVS web access application form at http://www.inf.ed.ac.uk/systems/cvs/new/. (For the transcription in NXT format, just checkout `DATA/ICSI/NXT-format/Main`. For our coding, checkout `DATA/ICSI/NXT-format/Contributions/TopicSegmentation`.) Regarding the audio data, you need to download the mixed wav files from IDIAP Multimodal Media File Server http://mmm.idiap.ch/ (under *protected area*). File name should be changed to something like `Bdb001.interaction.wav`. The audio should be put under `DATA/ICSI/NXT-format/Signals/` or put somewhere else but linked to this directory.

Finally, check out the segmentation tool under `Tools/NXTtools`. Before you could start the tool, there is still something to do with the localization. One is to change the NXT path in `Tools/NXTtools/topic.sh` to your NXT installation path and save it to your local cvs root directory. The other is to amend your matadata file `ICSI-metadata.xml` under `DATA/ICSI/NXT-format/Main/`, by adding the following to its "interaction-codings" section.

```
<!-- topic codings from the NXT topic Segmenter -->
<coding-file name="topic">
    <structural-layer name="topic-layer" recursive-points-to="segment-layer">
        <code name="topic">
            <attribute name="type" value-type="string"/>
        </code>
    </structural-layer>
</coding-file>
```

Now you will be able to start the tool by `sh Tools/NXTtools/topic.sh`. To view existing topic segmentations, copy the relevant XML files (*.segs.xml and *.topic.xml) with the same meeting name from (some directory under) `DATA/ICSI/NXT-format/Contributions/Topicsegmentation/` into your local main data directory `DATA/ICSI/NXT-format/Main/`.

NB: if your want to contribute, please make a directory of your own under `DATA/ICSI/NXT-format/Contributions/Topicsegmentation/` (preferred: same with your cvs username), copy your coding (including *.segs.xml and *.topic.xml) from your local main data directory `DATA/ICSI/NXT-format/Main/` to your own directory, and check in from there.

6 Appendix

6.1 Checkout Script: starttopic

```
#!/usr/bin/perl

# start a user off in topic segmenter. Simply creates a newdirectory
# for the user in CVS to make checking in simpler.
# Assumes you are CVS logged in!!

# Change these for non-local (to Edinburgh) setup

$CVS = "/usr/bin/cvs";
$CVSPATH="cvs.inf.ed.ac.uk:/disk/cvs/ami";
$NXT="/group/ltg/projects/NITE/nxt";
$SIGNALS="/group/project/ami1/ICSI/audiomix";
$base="Data/ICSI/NXT-format";
$shortbase="ICSI/NXT-format";
$toolbase="Tools/NXTtools";
$main="$base/Main";
$meta="$base/Main/ICSI-metadata.xml";
$topics="$base/Contributions/TopicSegmentation";
$shorttopics="$shortbase/Contributions/TopicSegmentation";
$topictool="$toolbase/topic.sh";
$CP = "/bin/cp";
$LN = "/bin/ln";
$metafragment="              <!-- topic codings from the NXT topic Segmenter -->
             <coding-file name=\"topic\">
                 <structural-layer name=\"topic-layer\" recursive-points-to=\"segment-
                     <code name=\"topic\">
                         <attribute name=\"type\" value-type=\"string\"/>
                     </code>
                 </structural-layer>
             </coding-file>
";

sub usage {
    print STDERR "Usage: starttopic -u <cvs-username> -d <checkout-directory>\n";
    exit 1;
}


## Start Here
$outdir=".";

# Get the args
while (@ARGV) {
#    print @ARGV;
    $arg = shift(@ARGV);
    if ($arg eq "-u") {
        $username = shift(@ARGV);
        if (!$username) { &usage; }
    } elsif ($arg eq "-d") {
        $outdir = shift(@ARGV);
```

```
51          if (!$outdir) { &usage; }
52      } elsif ($arg =~ /^-/) {
53          &usage;
54      }
55  }
56
57  if (!$username) {
58      &usage;
59  }
60
61  $path=":pserver:$username\@$CVSPATH";
62
63  if (-d "$outdir/$base") {
64      die "Decided not to check out as you already have data in $outdir/$base";
65  }
66
67  chdir $outdir || die "Can't change directory to $outdir";
68  # print "$CVS -d$path co $base\n";
69  `$CVS co $main`;
70  `$CVS co $topics`;
71  `$CVS co $toolbase`;
72
73  $coderdir="$outdir/$topics/$username";
74  if (!-d "$coderdir") {
75      mkdir "$coderdir" || die "Can't create coder directory $coderdir";
76      chdir "$outdir/Data" || die "Can't change directory to $outdir/Data";
77      `$CVS add $shorttopics/$username`;
78      chdir $outdir || die "Can't change directory to $outdir";
79  }
80
81  # edit the metadata
82
83  if (!(-f $meta)) {
84      print stdout "\nWaiting for metadata.\n";
85      $wt=1000000;
86      $maxwt=10;
87      $curwt=0;
88      while (!(-f $meta) && ($curwt<$maxwt)) {
89          for ($i=0; $i<$wt; $i++) { }
90          $curwt++;
91          print ".\n";
92      }
93  }
94
95  if (!(-f $meta)) {
96      print "ERROR: no metadata file '$meta' - please contact the script author jonatha
97  } else {
98      open (IN, "$meta") || die "can't open metadata file for read ($meta)";
99      $met="";
100     while (<IN>) {
101         $met.=$_;
102         if (/<interaction-codings>/) { $met.=$metafragment; }
103     }
104     close(IN);
```

```
105
106     open (OUT, ">$meta") || die "can't open metadata file for write ($meta)";
107     print OUT $met;
108     close OUT;
109 }
110
111 if (!(-f $topictool)) {
112     print "ERROR: no shell script '$topictool' - please contact the script author jon
113 } else {
114 # edit the shell script
115     open (IN, "$topictool") || die "can't open shell script for read ($topictool)";
116     open (OUT, ">$outdir/topic.sh") || die "can't open shell script for read ($outdir
117     while (<IN>) {
118         if (/ICSITopicSegmenter/) { s/-c\s+[^\s]*/-c $meta/; }
119         if (/NXT=/) { $_="NXT=\"$NXT\";\n"; }
120         print OUT $_;
121     }
122     close(IN);
123     close(OUT);
124 }
125
126 # link to a local copy of the signals directory
127 #print "$LN $SIGNALS $base/Signals\n";
128 `$LN -s $SIGNALS $base/Signals`;
129
130 # Copy any existing data to the local Main directory
131 `$CP $coderdir/*.xml $main`;
132
133 print "Completed checkout. \nUse 'sh topic.sh' to code and 'checkintopic' to check in
```

6.2 Checkin Script: checkintopic

```
1  #!/usr/bin/perl
2
3  # check in data produced using the topic segmenter.  Copies all the
4  # segment and topic data from the main directory to the user's
5  # directory and CVS checks it in.
6  # this assumes you're in the right directory (above the 'Data' dir)!!
7
8  # Change these for non-local (to Edinburgh) setup
9
10 $CVS = "/usr/bin/cvs";
11 $CVSPATH="cvs.inf.ed.ac.uk:/disk/cvs/ami";
12 $CP = "/bin/cp";
13 $base="Data/ICSI/NXT-format";
14 $main="$base/Main";
15 $meta="$base/Main/ICSI-metadata.xml";
16 $topics="$base/Contributions/TopicSegmentation";
17
18 sub usage {
19     print STDERR "Usage: checkintopic -u <cvs-username> -o <observation>\n";
20     exit 1;
21 }
22
```

```
23
24 ## Start Here
25
26 # Get the args
27 while (@ARGV) {
28 #     print @ARGV;
29     $arg = shift(@ARGV);
30     if ($arg eq "-u") {
31         $username = shift(@ARGV);
32         if (!$username) { &usage; }
33     } elsif ($arg eq "-o") {
34         $obs = shift(@ARGV);
35         if (!$obs) { &usage; }
36     } elsif ($arg =~ /^-/) {
37         &usage;
38     }
39 }
40
41 if (!$username || !$obs) {
42     &usage;
43 }
44
45 $coderdir="$topics/$username";
46
47 # copy to coder directory
48 `$CP $main/$obs.*.segs.xml $coderdir`;
49 `$CP $main/$obs.topic.xml $coderdir`;
50
51 # Note that we have to be in a CVS dir to successfully check in while
52 # cvs logged in.
53 chdir "Data" || die "Can't change directory to Data";
54 $coderdir=~ s/Data\///;
55 # Add and commit
56 print "$CVS add $coderdir/$obs.*.segs.xml\n";
57 `$CVS add $coderdir/$obs.*.segs.xml`;
58 print "$CVS add $coderdir/$obs.topic.xml\n";
59 `$CVS add $coderdir/$obs.topic.xml`;
60 `$CVS commit -m "auto checkin" $coderdir`;
61
62 print "Completed checkin.\n";
```