Project Note AP04-2011-01

# SessionDirector

Author: Thomas Hanke

DOI: https://doi.org/10.25592/uhhfdm.825

Version 1: 2011-09-01 for version 1.0 of SessionDirector used in the first data collection campaign of the DGS-Korpus project
Version 2: 2024-03-16 for version 1.1 of SessionDirector used in the second data collection campaign

## Introduction

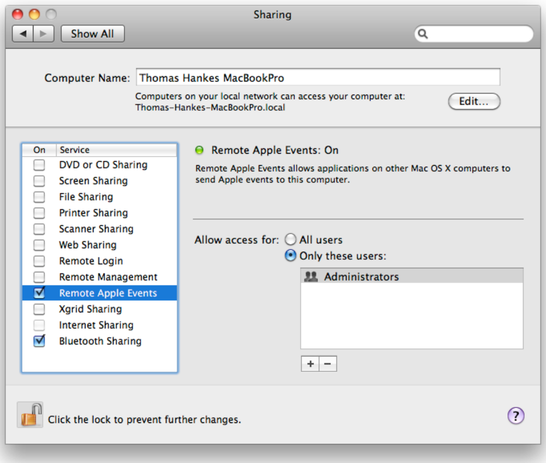
SessionDirector supports the moderator in managing a multi-task data collection session with one or two informants. It gives an overview of the tasks, guides the moderator through the pre-planned steps, remote-controls the informants' screens. In addition, it helps the moderator in keeping an eye on the time, and finally it writes a log file that can be used to pre-fill the transcript.
SessionDirector runs on macOS only as it uses AppleScript to remote-control Keynote[1] to present slides to the informants' screens. For moderators preparing their sessions, a Windows version is available that allows the moderator to review the session content.
At startup, SessionDirector reads an XML file detailing the session. An example of such a file is discussed in this document. The XML file refers to slide numbers in a Keynote document on the informants' machines.

## Setting the studio computers up for SessionDirector

The basic setup SessionDirector expects is that MacOS computers with Keynote installed are available for both informants as well as the moderator. All three machines need to be connected in a network. SessionDirector only runs on the moderator's computer, it remote-controls Keynote on the two other machines.
On the machines for the informants, you need to enable remote Apple Events:

- Open *System Preferences* from the Apple menu and select *Sharing*.

[1] Keynote from Apple is a program quite similar to Microsoft Powerpoint. In principle, it should be possible to adapt SessionDirector to control PowerPoint instead of Keynote. If you are interested in doing so, please request the SessionDirector source code in order to make the necessary adaptations to the AppleScript elements used.



Computer Name: Thomas Hankes MacBookPro

Computers on your local network can access your computer at: Thomas-Hankes-MacBookPro.local

| On | Service |
|---|---|
| ☐ | DVD or CD Sharing |
| ☐ | Screen Sharing |
| ☐ | File Sharing |
| ☐ | Printer Sharing |
| ☐ | Scanner Sharing |
| ☐ | Web Sharing |
| ☐ | Remote Login |
| ☐ | Remote Management |
| ✓ | Remote Apple Events |
| ☐ | Xgrid Sharing |
| ☐ | Internet Sharing |
| ✓ | Bluetooth Sharing |

● Remote Apple Events: On

Remote Apple Events allows applications on other Mac OS X computers to send Apple events to this computer.

Allow access for: All users [✓ Only these users:]

Administrators

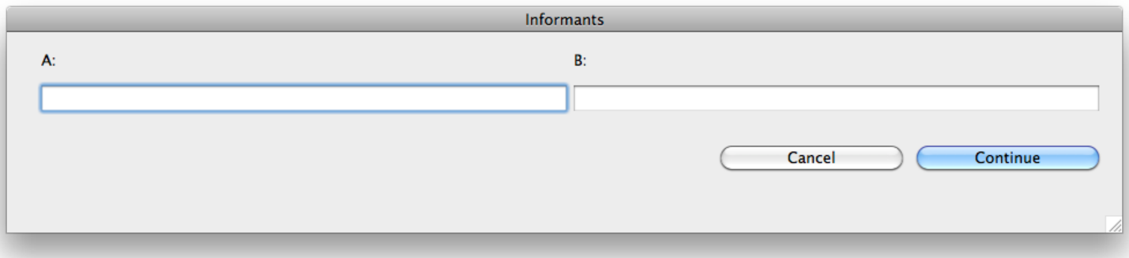
Click the lock to prevent further changes.

- Enable *Remote Apple Events*. (You might need to unlock the window with an admin password.) For security reasons, it is advisable to restrict access to a specific user name. (Switch to *Accounts* if you want to create a new user account for this specific use.) The user name (and the corresponding password) need to specified in the XML file Session Director reads in. (There, the password is not encrypted, it is therefore a good idea not to use a user account that is also used for other tasks, but to create a new one.)
- Close *System Preferences*.

Session Director expects Keynote to be launched and the slides document to be open (as the only document). Switch both Keynote installations to presentation mode where they will remain until Session Director sends its first command. (So it is a good idea to start the Keynote slides document with a “Welcome” screen.
If you want to test your setup with the demo documents provided, make sure you adapt the XML file not only to reflect the username and password you have identified, but also the IP addresses of the two other machines. The IP addresses go into the attributes *keynote_a* and *keynote_b*. They can either be internet machine names (such as *keynote_a.sign-lang.uni-hamburg.de*) or numerical addresses (such as *134.100.164.1*) or Bonjour addresses (machine name as set in *Sharing* plus *.local* such as *dgs-korpus-studio-a.local*).

## Using SessionDirector for the moderator[2]

- Launch SessionDirector and select *Open…* from the *File* menu.
- Select the XML file describing the session to be carried out.
- SessionDirector now asks you for the names or codes of the two informants involved. A is the person sitting left to you, B is on the right.



[2] A more in-detail user manual in German related to our tasks set is available as project note AP02-2009-03.

- Click *Continue* to start the session.

SessionDirector now opens its main window, displaying on the left all the tasks to be worked on during the session as defined in the XML file just read in. The session definition also defines the time available for the session. If for some reason there is more or less time available, you can set a time to close. This is useful if you have a late start, but one informant needs to catch a train, for example (*Define End Time…* from the *File* menu).
The standard approach is to work on one task after the other, from top to bottom. However, by clicking *Allow Jumping back & forth* in the lower left of the main window will give you the freedom to skip some items and later go back if necessary.
Within the current task, there may be several steps (subtasks) to go through. As you start one, the next item becomes available, but you can repeat individual steps, e.g. to have the informants watch the instructions once again. For each step, a certain time has been allocated. This is displayed right to the subtask. On the top of the window, you find the timeline for the current task. While it happens quite often that the time allocated for one subtask is not sufficient or too long, they might average on the task level. If not, you see an extra time bar to the right of the standard task bar. If that is fully filled, it definitely is time to have the informants move to the next task.
If delays add up during the session, the time window (on the left of the screen) may change its color from green to yellow to red. If red, you should consider dropping not so important tasks, SessionDirector suggests what tasks to skip by placing a scissors symbol in front of the task name.
The time window displays five pieces of information, all in hours and minutes:

- the current local time
- the time into the session
- the delta to the plan: If the number is negative, the current session is ahead of time. If the number is positive, it is behind schedule.
- the maximum time to the next pause (i.e. the time still available respecting the maximum duration of the current subsession – not displayed if no subsession is active)
- the defined end time of the session, either set by the user or defaulting to the start time plus the maximum duration of the session as defined in the XML file.

In the right part of the main window, you may find additional hints for you to consider if something goes wrong. For example, on a certain topic discussion might not really take off. Then hopefully you find here some suggestions on other topics you could bring into the discussion.
If necessary, you can write down notes in the lower right part of the main window concerning individual tasks such as comments or interesting observations.
Probably you have planned in some breaks that show as tasks in the tasks list. If an informant goes to see the restroom in between, just select *Pause* from the *File* menu. That way, this pause is recorded in the log file and can automatically be transferred to the transcript later on. At the end of such a pause, recording is resumed where you left off.

## Developing SessionDirector XML files

Certainly the best way to develop a SessionDirector file is to start with an example file provided by us. Then change and add as necessary and test the results. If SessionDirector refuses to open a file, it most certainly is not a valid XML file. Unfortunately, there is no indication where the error is. So you might want to use an XML-aware editor to write the file.
The top-level element is the *session*. Nested into that are *recording*, *task*, and *pause* elements. All of these elements should have an id (that is logged with the timestamp when the element is called up, so it should be unique). They can (and should) have duration specifiers:

*minimum_duration*, *average_duration*, and *maximum_duration*. The value is either a number (minutes) or integer:integer (minutes:seconds).
Optionally, *recording*, *task*, and *pause* elements can be grouped under a *subsession* element instead of being direct children to the *session*. The only role of a *subsession* to obey a *maximum_duration* attribute for each *subsession*.

### The *session* element

The session element should have a name attribute describing its contents, e.g.

```
<session name="DGS-Korpus Berlin"…
```

If you want to rank your tasks to determine what to skip in case the session runs out of time, you should provide a *standard_rank*, i.e. the rank up to which all tasks should be included under normal circumstances:

```
standard_rank="5"
```

The examples suggests that all tasks with ranks up to 5 form the standard set, lower numbers being more important. Tasks with numbers higher than the standard rank will be included if time is left over. See below for more comments on ranking.

We recommend that the session has a *maximum_duration* attribute. (Do not forget the pauses when calculating the length of your sessions!) An *average_duration* is useful as well, a *minimum_duration* has no effect on this level. If the average duration is not specified, it is calculated as the sum of the average durations of the contained elements. A maximum duration, however, need not be the sum of the max durations of the contained elements, it should be somewhere between the sum of the averages and the sum of the maxima.
With the attributes *green_threshold*, *yellow_threshold* and *red_threshold* you determine when the time window changes color depending how well the session is in time. With

```
green_threshold="-30"
yellow_threshold="0"
red_threshold="20"
```

you define that the window is yellow if you are behind schedule, or red if more than 20 minutes behind. It is green if you are max. 30 minutes ahead of time, and white if more than 30 minutes.

For the attributes *keynote_a*, *keynote_b*, *username* and *password* see above.

## The *subsession* element

The *subsession* element optionally groups the *recording*, *pause*, and *task* elements into blocks that are assigned a *maximum_duration*. This duration is taken into account in time-keeping. E.g. if there is a limit for recording in one go (as your record to media with limited capacity and then need to switch media before continuing), use *subsession*s with *maximum_duration* set to a bit less than your maximum recording time so that the moderator can bring all elements grouped into the *subsession* to be finished before the maximum recording time is reached.

### The *recording* element

The recording element asks the moderator to confirm that a certain action has been completed, such as using the clapperboard:

```
<recording id="clapperboard-1" name="Clapperboard" average_duration="0:30">
       <checklist text="Important: Show clapperboard to all cameras (to help movie sync)!"/>
</recording>
```

## The *task* element

Tasks mainly consist of subelements, with at minimum one *subtask*: Subtasks are the elements where the informants are actually expected to sign.
In addition to subtasks, tasks may also contain *presentation* and *decision* elements. Typically, a task starts with a presentation explaining what to do, followed by one or more subtasks.

## The *decision* element

If the informants have the choice between some options, build in a *decision* element so that the decision taken can be recorded in the log file. Here is an example:

```
       <task id="6" name="Jokes" average_duration="5" maximum_duration="10" rank="1" >
              <presentation id="6.1.1" name="Let's tell jokes" source="8" duration="0:29" />
              <decision name="Who is first?" average_duration="0:10">
                     <option id="OptionAB" name="Person A is first, then person B" />
                     <option id="OptionBA" name="Person B is first, then person A" />
              </decision>
              <subtask id="6.2" name="" >
                     <narration id="6.2.1" name="Joke 1" source="2" average_duration="2"
                                                                  maximum_duration="7" />
              </subtask>
              <subtask id="6.3" name="" >
                     <narration id="6.3.1" name="Joke 2" source="2" average_duration="2"
                                                                  maximum_duration="7" />
              </subtask>
       </task>
```

The *decision* element's *name* attributes are shown in the moderator's user interface, and the moderator needs to document the decision taken by the informants before being able to proceed with the next subtask. The *source* attribute means the slide number where to start the presentation if both informants shall see the same set of slides. If you want them to view different slides, use the attributes *source_a* and *source_b*.

## The *presentation* element

Presentations are expected to have a fixed duration, i.e. here you specify a *duration* attribute instead of *average_duration* etc. If you have a task intro movie explaining the task and then the actual elicitation material, make them two *presentation* elements. By that, you can replay the explanation if necessary, or only the elicitation.

## The *subtask* element

Subtasks consist of any combination of *presentation*, *narration* and *narration_with_stimulus* elements. In addition, subtasks should carry a *target* attribute, logging which informant is expected to be the active one, possible values are A, B, or AB.

## The *narration* element

During a *narration* element, one of the informants is expected to sign something to the other informant (subtask's *target* being A or B) or the two informants to discuss something (*target* AB). In most cases, you do not want to distract the informants by displaying something one the screens, so you specify *source* to point to a black screen.

## The *narration_with_stimulus* element

Here, the signer comments different stimuli being shown one after the other. The moderator clicks the button to proceed to the next stimulus if one contribution is finished. With the next stimulus, the signer is expected to start again as the task is well known by then. For that to work, specify different slides sets in source, separated with semicolons. Here is an example where both informants see the same elicitation data:

```
<task id="15" name="Traffic signs" average_duration="20" maximum_duration="25" rank="3" >
        <presentation id="15.1.1" name="Explanation" source="112" duration="1:05" />
        <subtask id="15.2" name="" >
           <narration_with_stimulus id="15.2.1" name="Traffic signs #1"
               source="114;116;118;120;122;124" average_duration="7" maximum_duration="12.5" />
           <narration_with_stimulus id="15.2.2" name=" Traffic signs #2"
               source="126;128;130;132;134;136" average_duration="7" maximum_duration="12.5" />
        </subtask>
</task>
```

Here is an example where the informants change roles for each picture shown:

```
<subtask id="26.2" name="Picture naming" >
        <presentation id="26.2.1" name="Explanation" source="222" duration="0:39" />
        <narration_with_stimulus id="26.2.2" name="Pictures #1"
                          source_a="226;224;230;224;234;224;238;224;242"
                          source_b="224;228;224;232;224;236;224;240;224"
                          average_duration="3:10" maximum_duration="5" />
        <narration_with_stimulus id="26.2.3" name="Pictures #2"
                          source_a="224;246;224;250;224;254;224;258;224"
                          source_b="244;224;248;224;252;224;256;224;260"
                          average_duration="3:10" maximum_duration="5" />
</subtask>
```

(Slide 224 in this example is a “black” slide.)

### The *pause* element

The pause element should be self-explanatory by now:

```
<pause id="Pause 1" name="Coffee break" minimum_duration="10" average_duration="15"
        maximum_duration="20"/>
```

### The *rank* attribute

The lower the rank attribute, the more important a task is. Tasks with *standard_rank* may be left out if the session is much behind, tasks with ranks higher than *standard_rank* are considered optional and will be skipped unless there is time left. See below for more comments on ranking.

### The *comment* element

The comment element contains text to be displayed on the right side of the main window, e.g. to suggest extra topics if the informants are not happy with the suggestions made. The comments are linked to the task by having the same id. That way they can be written stand-off at the end of the file.

## Building the Keynote slides

Keynote gets the command from SessionDirector to jump to a specific slide that is identified by its number. However, Keynote only displays the next slide. This results in a recommended composition of the slides in a Keynote document as follows:

| | |
|---|---|
| 1 | – Welcome slide |
| 2 | – Jump target for black screen |
| 3 | – Black screen |
| 4 | – for 1st slideshow |
| 5…n | – Slideshow with automatic transitions from slide to slide, no transition for last |
| n+1 | – Jump target for 2nd slideshow |
| n+2…m | – Slideshow with automatic transitions from slide to slide, no transition for last |
| … | |

If you later on want to replace one slides set with another, do not do an in-situ replacement. Instead, append the new slides to the end of the document, with a preceding jump target, and change the slide number for this presentation in the XML format. That way, all following presentations will not be affected by replacing a slideshow with another containing more or less slides.

## Ranking

Ranking determines which tasks should be skipped if time requires. Tasks with a rank higher than the session's *standard_rank* are considered optional and will only be included if time permits. (They are marked with a + symbol in SessionDirector's tasks list.) Tasks with a rank equal to the session's *standard_rank* may be left out the session is behind schedule. Currently, any ranking below *standard_rank* has now influence, but for future compatibility we suggest that you actually rank your tasks by importance, i.e. how bad it is if they have to be skipped.

## Testing the setup

Even if you have produced a correct XML file, there probably are a couple of glitches you will only discover if you actually play through a session (probably ignoring the time constraints in the first case). Also have a look at the log file and make sure you are happy with what is logged how. Are the ids unique so that you know what was going on?
Finally, you need to make sure that the timings are all correct. This is best done by doing some life tests – it is then not only your SessionDirector file being tested, but also the elicitation material as well.

## Formats not covered

If you find that some elicitation task cannot be modelled with the available elements, talk to us. We might either have a solution within the existing framework, or be able to extend the program to handle a new element.
The program might also be scaled up to handle three or four informants. There is a limit, however, with the current synchronous communication mechanism between SessionDirector and remote Keynotes: SessionDirector waits for Keynote to acknowledge (not execute) the command. This results in some delay between identical performances for informants A and B. In case of more informants, this delay might actually cause a major problem.

## SessionDirector log files

SessionDirector creates a log file for each run which looks as shown here:

```
descriptor_file=LEI15_Set_ZH.xml program_version=1.0.15 informantA= XXXXXXXXXXXXXXXXXX
informantB=XXXXXXXXXXXXXXXXXX 09:57:17 Session started 09:57:17 Task start 1
09:58:32 Task flash 1 09:58:47 Task 1
09:59:06 1.1.1
10:00:47 1.2.1
10:00:49 Done with 1.2 10:00:49 Task 2 10:00:50 2.2.1
10:05:55 Done with 2.2 10:05:55 Task 3 10:06:46 3.1.1
10:07:49 3.2.1
10:07:58 Done with 3.2 10:07:58 Task 4 10:08:01 4.2.1
10:16:00 Done with 4.2 10:16:01 Task 35 10:20:53 Done with 35.2 10:20:53 Task 5 10:21:06 5.1.1
10:21:43 OptionBA
10:21:45 5.2.1
10:21:55 Done with 5.2
10:21:57 5.3.1
10:21:59 Done with 5.3
10:21:59 Task 6
10:22:25 6.1.1
10:22:59 OptionAB
10:23:01 6.2.1
10:24:04 Done with 6.2
10:24:06 6.3.1
10:27:01 Done with 6.3
```

```
10:27:01 Notes: XXXXXXXXXXXXXXXXXXXXXXXXXX 10:27:01 Task 46
10:27:52 OptionBA 10:27:55 46.2.1 10:38:50 Done with 46.2 10:38:53 46.3.1 10:49:28 Done with
46.3 10:49:28 Task 11 10:49:50 11.1.1 10:51:26 11.2.1#1 10:58:11 Done with 11.2 10:58:11 Task
9 10:58:24 9.1.1
10:59:36 9.2.1 11:05:33 9.2.2 11:08:49 Done with 9.2 11:09:02 9.3.1 11:12:06 9.3.2 11:14:47
Done with 9.3 11:14:47 Task flash 5 11:14:56 Task stop 3 11:15:51 Task Pause 1 ...
```

So this file tells you what task was being worked on during which time interval. By converting this into annotation, you can prefill a transcript which might help structuring the subsequent annotation process. For import into iLex we use Perl scripts to convert the log, these are available upon request. For import into ELAN, a similar script is straightforward to write.