Converting HRS from Blaise 4 to Blaise 5 – the Example of Rosters

Jason Ostergren
Health and Retirement Study, University of Michigan Institute for Social Research

© 2018 Regents of the University of Michigan
Where Does HRS Use Rosters and Why?

- The HRS instrument includes separate rosters to collect arrays of children, household members, siblings, and pensions.
- The arrays of people and pensions collected in the rosters drive follow-up sequences which show up in different sections of the instrument; some are shared between respondents.
- Between waves, the people and pensions in the rosters are given identifiers for tracking and are preloaded and pre-filled into the roster of the following wave; this reduces interview length, since it’s quicker to verify than to collect from scratch.
A child roster in the Blaise 4 HRS instrument
Why Display As Roster Rather than Sequence?

- We believe that a roster presentation helps to reduce errors and duplication (though it does not eliminate them) by providing more visibly accessible information than a series of individual looped questions.
- A roster display can facilitate navigation to correct mistakes – a sequence of questions means backing up through many screens; and we have found that it is the case that a respondent and/or interviewer may not fully realize that they have mixed up or duplicated information until they are deep into the set of questions.
What Did HRS **Want** to Do for Blaise 5 Rosters?

- HRS initially wanted a dashboard-type screen:
  - Start with overview of all items (persons or pensions)
  - Each item represented by clickable button or similar which goes to a detail screen to add or edit data about item
  - Completing detail screen returns to dashboard, with some indicator (e.g. a green checkmark) for the item completed, and other indicators (e.g. a red ‘X’) for items not yet examined
  - Once all items completed, the button to allow movement past the dashboard would appear or be highlighted
What Did HRS Actually Do?

• Loop through detail pages first; roster summary afterwards
• Each detail page shows summary of person/pension if preloaded; asks to verify that info is correct
• If answer is no, detail page summary and verification question are hidden and replaced by detail questions on the same page
• After any preloaded persons/pensions, detail page question asks if there are new ones, and detail questions appear if ‘Yes’
• Templates provide visual cues about full list of people/pensions
Child detail summary screen for a preloaded child

Let's start with KEVIN. Is all the information we have listed below correct?

First name: KEVIN
Relationship to you: Son
Relationship to your wife: Son
Sex: Male
Alive or deceased: Alive

- Yes, this information is accurate
- No, this information needs updating

Note the names on the right are clickable navigation buttons and the current child is highlighted
Detail Summary Edit Questions

- If the detail summary question is answered “No”, after clicking “Next,” it is removed from the route and this group of detail questions appears in its place. If it had been answered “Yes” instead, the survey would have proceeded to the next child detail summary page.
Add New Item Page

After pages for any preloaded children have been verified or edited, the survey displays pages allowing new children to be added. Similar to the detail summary question, one answer ("Yes," in this case) opens up a group of questions; the other answer proceeds to the roster summary screen.
Blaise 5 Logic

Here is a simplified code snippet from the child detail summary page: X055 is a tracking number; A221 is the detail summary question; GROUP_EditChild_GROUPTEXT is the set of detail questions; A208 is the new item question

```
3359 IF X055APPN = EMPTY THEN
3360 A208ANewPerson
3361 ELSEIF A221AConfirm <> NO THEN
3362 IF SELFADMIN AND X058AFName <> MISSINGNAME AND X058AFName <> MISSINGNAME_SPN THEN
3363 A221AConfirm
3364 ELSE
3365 A221AConfirm := NO
3366 ENDIF
3367 ENDIF
3368 IF A221AConfirm = NO OR A208ANewPerson = yes THEN
3369 GROUP>EditChild_GROUPTEXT
3370 ENDIF
```
Once at the summary page, the buttons for “Add a child” and “Summary Page” appear in the list on the right (the buttons were hidden until the “IsVisited” property of the summary page was true). Once they appear, they will stay visible even on prior pages.
Split Roster

• Blaise 4 HRS roster had many columns mashed together (~17 questions for each child, for example)
• In Blaise 5, HRS wanted to minimize any scrolling for self-respondents, and also avoid difficult to read screens, which meant less questions per screen
• HRS ended up moving most of the original roster questions to an immediate follow-up series, thus splitting the roster in two parts
Split Roster, part 2

- The first part was the detail and summary sequence previously described; the second was a standard loop
- In the first part, we put variables that change more rarely or by mistake (such as relationship to respondent)
- These variables also form a sort of compound identifier
- In the second part, we put the remainder of variables (such as marital status and residency for people) which may vary and are less suitable as identifiers
- For example, “Stepson” may be more useful as part of an identifier for a person than “Married”
**Additional Programming Problems**

- Following is an overview of programming challenges HRS ran into outside of the basic logic and wording:
  - The IndexAreaList (the buttons on the right of the roster screens) as the main visual element of the roster
  - Handling layout, including questions per page
  - Handling gates in self-administered interviews, including different methods of page advancement
  - Adding a router to insert additional audit trail data for gates
The IndexAreaList master page template
The IndexItem button template
.layout file (edited in text editor)

• Entries for switching master page TemplateTarget and RoutelItemsPerPage

```xml
<RouteletLayoutInstructions RoutelItemName="SecA2_ChildTab">
  <Instructions>
    <InstructionLocator="Before" RoutelItemsPerPage="4" />
    <TemplateInstruction Locator="Before" TemplateName="IRS_SectionIndex" TemplateTarget="MasterPage" />
  </Instructions>
</RouteletLayoutInstructions>
```

© 2018 Regents of the University of Michigan
View from Layout Editor

• Even though we do not use it for editing, we can however use the layout view to see edits we make manually to the .layout file (namely switching to 3 questions per page in the detail sections) – this screenshot shows the detail summary and detail question set page for one child.
Simplified code snippet for gate

4550 A224_ChildSummary
4551 A224_ChildSummary.AlienActionEvent := '1'
4552 IF A224_ChildSummary <> 1 THEN
4553   A224_ChildSummary := FUNC_SetGateFieldsOnPreviousFieldIsVisited('A222_ChildGate', A224_ChildSummary.IsVisited)
4554 ENDIF
4555 A222_ChildGate
4556 ENDCGROUP
4557 FUNCTION FUNC_SetGateFieldsOnPreviousFieldIsVisited : INTEGER
4558 PARAMETERS
4559 IMPORT
4560   piCheckCurrentField: STRING
4561   piTriggerFieldIsVisited : TIsVisitedFieldProperty
4562 RULES
4563   IF POSITION(piCheckCurrentField, CurrentFieldName) > 0 AND piTriggerFieldIsVisited = Yes THEN
4564   ENDIF
4565 RESULT := 1
4566 ENDFUNCTION
“Router” Code

- Code snippet triggered by both events which can cause a forward page change from custom-action.service.ts, a new ActionService

```typescript
protected nextPage(): IActionKeyValuePair {
    this.signalsSuppressService.suppressSignals();
    let nextPageResult = super.nextPage();
    if (nextPageResult) {
        this.alienActionEventService.checkForAlienActionEvent(this);  
    }
    return nextPageResult;
}

protected nextField(actionObj: IAction, controlId: string): IActionKeyValuePair{
    let nextFieldResult = super.nextField(actionObj, controlId);
    if (nextFieldResult) {
        this.alienActionEventService.checkForAlienActionEvent(this);
    }
    return nextFieldResult;
}
```
“Router” Code, Continued...

- Code snippet writing an extra record to the audit trail service when triggered on a field with the correct property from alien-action-event.service.ts, another new class

```typescript
public checkForAlienActionEvent(kind: number) {
    if (kind === 0 || kind === 5) {
        const actionNameAssignField: string = 'AssignField';
        const fieldPropertyName: string = 'AlienActionEvent';
        const actionKind: string = (kind === 0) ? 'NextPage' : 'NextField';
        const detected: string[] = this.detectFieldProperty(fieldPropertyName);
        if (detected.length > 0) {
            this.auditTrailService.logActionEvent(actionNameAssignField, detected.join(','), AuditTrailLevel.None);
        }
    }
}
```