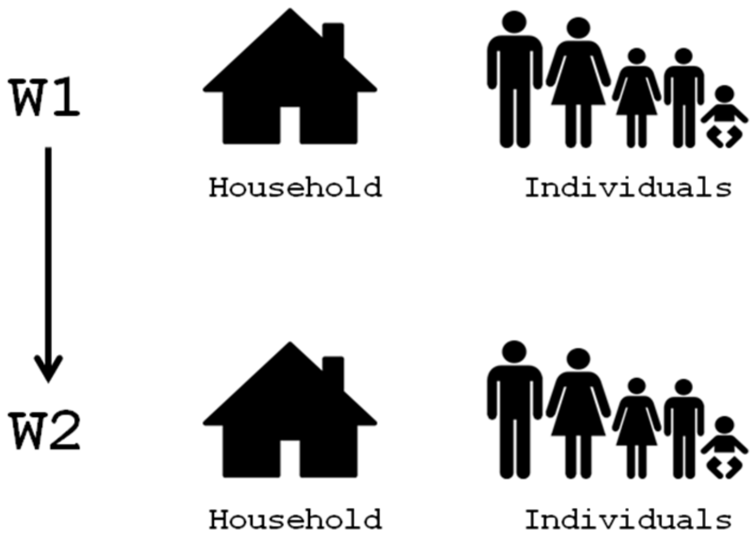
Developing and Testing A Complex Mixed Mode Questionnaire Instrument

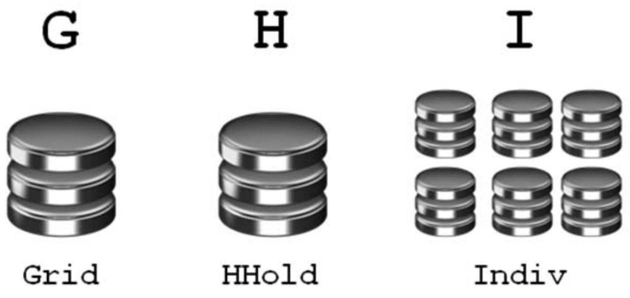
Richard Boreham, NatCen Social Research

Structure

Understanding Society is a longitudinal household survey. At Wave 1 households were randomly selected and interviewed, with every adult aged 16+ eligible for an individual interview. Anyone enumerated in an interviewed household at Wave 1 becomes an original sample member and they are included in the issued sample at all subsequent waves, and an attempt is made to interview them and all the people that they live with.

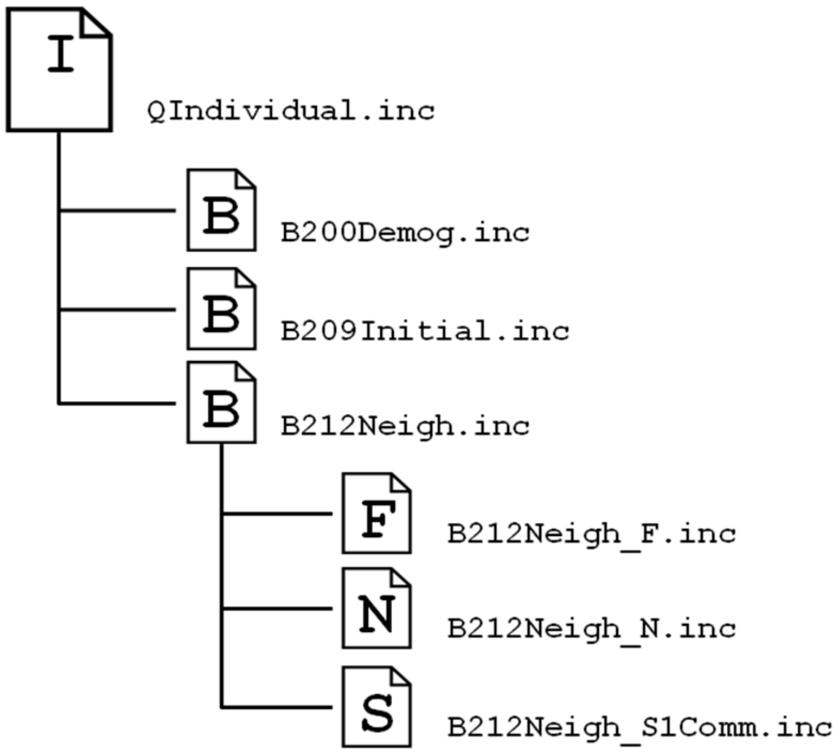


The Understanding Society questionnaire has 3 main components, a household grid, a household questionnaire, and then individual questionnaires for each adult aged 16+. These are programmed within one CAPI instrument as separate parallel blocks.



The household and individual instruments on Understanding Society consist of modules which are rotated on a wave by wave basis. Each module has its own separate block within the household or individual instrument, and each block is contained within a separate source file – all of which are named starting with a ‘B’, for example ‘B212Neigh.inc’ refers to the module on neighbourhood cohesion.

Within each block source file, the fields are specified within a separate field source file referenced from within the block, so B212Neigh_F.inc is called from the B212Neigh source file. At NatCen, researchers tend to program the fields, whereas programmers program the rules – therefore it makes sense to separate the fields from the rest of the block so that researchers can make amendments to question wording at the same time as programmers are working on the rules for a module. If there is a loop within a module, then the block containing the looped fields and rules is stored in a separate subfile and called from the main block. As an additional complication, the Understanding Society questionnaire is translated into 9 languages (other than English), so a distinction is made between translated and non-translated questions, where non-translated questions are either derived variables or questions which consist purely of interviewer instructions and with no question wording.



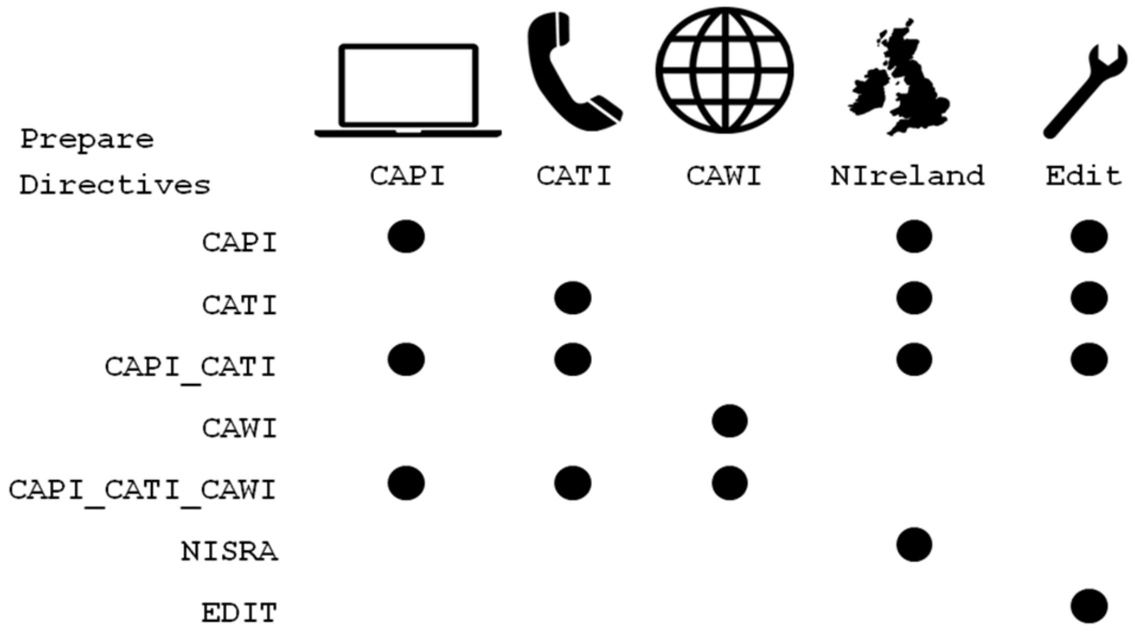
At Wave 1, the fieldwork was all face to face, so the questionnaire was programmed as a CAPI instrument. At Wave 2, Understanding Society incorporated the old British Household Panel Survey respondents, some of whom would only take part via a telephone interview, and so the questionnaire needed to work in both CAPI and CATI modes. For Wave 2, the questionnaire was developed initially in CAPI, then once the CAPI version was signed off as correct, it was frozen and a CATI version was developed using the CAPI version as the base. The situation was complicated, by having a separate version for the Northern Ireland sample as their fieldwork was carried out by NISRA (Northern Ireland Statistics and Research Agency), who use Blaise for their interviewing, but have a different sample management system to NatCen, which therefore required additional fields within the Blaise instrument. An edit version of the instrument was created to pull all the data from the separate instruments together.

The drawback of the approach taken at Wave 2 was that we had to maintain four separate versions of the Blaise instrument. Although the CAPI questionnaire had been signed off and frozen prior to creating any of the additional instruments, once fieldwork started, minor amendments had to be made to the CAPI and hence to all the other three versions, which made version control a nightmare. Therefore for Wave 3, one single instrument was created that used common code, primarily by using textfills to distinguish CAPI and CATI wording.

Understanding Society consists of a main survey wave and an innovation panel (IP) for methodological testing (such as the effect of different monetary incentives on response). The main methodological focus of IP5 was to test the use of mixed mode interviewing on overall response and to look at the mode effects associated with people's answers to individual questions. Therefore the Understanding Society interviewing instrument needed to work in CAWI as well as CAPI and CATI.

Our initial approach to re-programming the combined CAPI/CATI instrument was to continue to program using textfills for CAWI, and to use a ModeType variable to control routing of the different modes. However we quickly found that this was extremely time-consuming to program and was not a practical solution. The main problem we found was that although question wording could be similar for CAPI, CATI and CAWI, the Understanding Society has help screen for a large number of questions, and it was impossible to write help text that would make sense to both an interviewer and a respondent reading it, so we were having to create multiple textfills (as we are restricted to 255 characters for each textfill) for each question that had long help text. Textfills were also time consuming to create as they have to be referred to in 3 separate locations, namely specified under an Auxfield command, referred to in Fields and set in Rules. We realized that the best solution was to use Prepare Directives to identify the code that needed to be run in different modes.

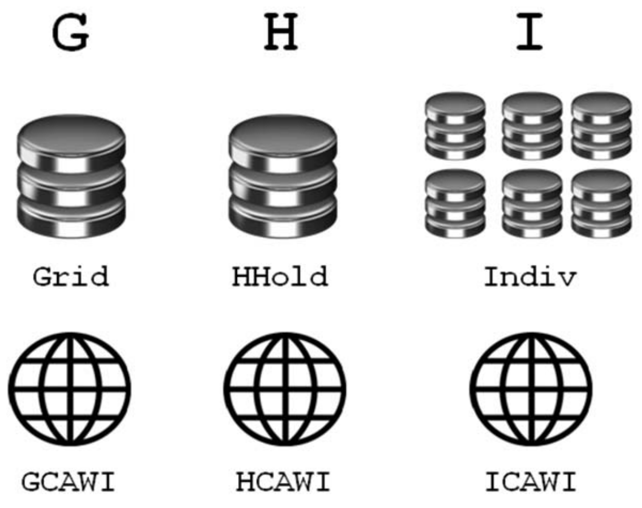
We identified that we needed to generate five different survey instruments. One instrument was needed for each mode, CAPI, CATI and CAWI. NISRA needed a separate instrument for Northern Ireland which was essentially the same as the CAPI instrument with some extra code, and a final Edit version was needed to bring all the different component versions together. We worked out that we needed 7 different Prepare Directives to be able to create the 5 different datamodels that were required.



Prepare Directives	CAPI	CATI	CAWI	NIreland	Edit
CAPI	●			●	●
CATI		●		●	●
CAPI_CATI	●	●		●	●
CAWI			●		
CAPI_CATI_CAWI	●	●	●		
NISRA				●	
EDIT					●

The Understanding Society instruments for IP5 were a 5 minute household grid, a 10 minute household questionnaire and a 32.5 minute individual questionnaire. Some routing within the household questionnaire and the individual questionnaire depends on variables within the household grid – for example parents were asked about childcare for each child within the household. We allow for up to 16 people to be enumerated in a household so there are numerous arrays within the household grid and individual questionnaire with 16 sets of the appropriate questions.

Due to the complexity of the existing CAPI/CATI instrument and concerns about navigating between parallel blocks in CAWI and possible performance issues, it was decided to have 3 separate CAWI instruments (Household Grid, Household Questionnaire and Individual Questionnaire) rather than one complete instrument.



As a result, separate Prepare Directives were needed for GCAWI, HCAWI and ICAWI, which meant that we needed 11 different Prepare Directives to generate 7 different datamodels.

Code within the body of the questionnaire

The W3 questionnaire has been developed as a joint CAPI/CATI instrument, so there were already some textfills in place to cope with differences between modes – mostly to do with the presence or lack of showcards in CAPI and CATI respectively.

```
FIELDS

NBRCOH1_A
   "^TF_SHOWCARD212A I am going to read out a set of statements
   that could be true about your neighbourhood. Please tell me
   how much you agree or disagree that each statement describes
   your neighbourhood.
   @/@/First, this is a close-knit neighbourhood. ^TF_READOUT"
  /"AGREE OR DISAGREE THAT NEIGHBOURHOOD IS CLOSE-KNIT" :TAgree
```

In this code there are textfills ^TF_SHOWCARD212A, and a global textfill ^TF_READOUT which are then set in the rules as follows.

```
RULES

TF_READOUT:=''
TF_SHOWCARD212A:=''
{$IFDEF CAPI}
      TF_SHOWCARD212A:='SHOWCARD 212A@/@/'
{$ENDIF}
{$IFDEF CATI}
      TF_READOUT:='@/@/INTERVIEWER: READ OUT'
{$ENDIF}
```

Previously the textfills had been set using routing based on a ModeType variable, but we altered the code so that every mode change was programmed using Prepare Directives.

The question NBRCOH1_A would actually work in CAWI as it stands because the textfills are set up correctly for CAWI. However, this question is actually the first question in a grid, therefore we need to split the question text up into the initial introductory sentence and sentence that is the question itself.

```
FIELDS

NBRCOH_CAWIIntro
"Here are a set of statements that could be true about your
neighbourhood. Please tell us how much you agree or disagree that
each statement describes your neighbourhood.": TCont

NBRCOH1_A (NC_NBRCOH1_A)
{$IFDEF CAPI_CATI}
   "^TF_SHOWCARD212A I am going to read out a set of statements
   that could be true about your neighbourhood. Please tell me
   how much you agree or disagree that each statement describes
   your neighbourhood.
   @/@/First, this is a close-knit neighbourhood. ^TF_READOUT"
{$ELSE}
   "@/@>This is a close-knit neighbourhood@>@/"
{$ENDIF}
/"AGREE OR DISAGREE THAT NEIGHBOURHOOD IS CLOSE-KNIT" :TAgree
```

We now have a NBRCOH _CAWIINTRO question variant, and because the NBRCOH1_A is part of a grid, we right-align the question text, and hence we do need to use Prepare Directives to create a CAWI version of the question enclosed with right align tags @> and with hard returns before and after the statement to space it out from other statements.

Throughout the questionnaire instrument we generally use the routing convention

```
{$IFDEF CAPI_CATI}
{$ELSE}
{$ENDIF}
```

Within the rules section we also use Prepare Directives to make sure that all datamodels are compatible.

```
RULES

{$IFDEF CAPI_CATI}
      NBRCOH_CAWIIntro.KEEP
{$ELSE}
      NBRCOH_CAWIIntro:=Cont
      NBRCOH_CAWIIntro
{$ENDIF}
NBRCOH1_A
NBRCOH2_A
NBRCOH3_A
NBRCOH4_A
```

So in the CAPI_CATI version NBRCOH_CAWIINTRO is kept, whereas in CAWI it is set to ‘cont’ (this is because it is displayed with just question text and no answer text so respondent can’t code an answer) and asked (effectively just the question text is displayed). The same effect could have been achieved by allowing NBRCOH_CAWIINTRO to be Empty.

Code at the Datamodel level

The Prepare Directives are set at the very top of the DataModel. The code below shows how the Prepare Directives have been set for the CAPI and ICAWI datamodels.

```
DATAMODEL USIP5CAPI "IP5 CAPI"

   {$DEFINE CAPI}
   {$DEFINE x-CATI}
   {$DEFINE CAPI_CATI}
   {$DEFINE CAPI_CATI_GCAWI}
   {$DEFINE CAPI_CATI_HCAWI}
   {$DEFINE CAPI_CATI_ICAWI}
   {$DEFINE x-GCAWI}
   {$DEFINE x-HCAWI}
   {$DEFINE x=ICAWI}
   {$DEFINE x-EDIT}
   {$DEFINE x-NISRA}

DATAMODEL USIP5CAWI "IP5 CAWI"

   {$DEFINE x-CAPI}
   {$DEFINE x-CATI}
   {$DEFINE x-CAPI_CATI}
   {$DEFINE x-CAPI_CATI_GCAWI}
   {$DEFINE x-CAPI_CATI_HCAWI}
   {$DEFINE CAPI_CATI_ICAWI}
   {$DEFINE x-GCAWI}
   {$DEFINE x-HCAWI}
   {$DEFINE ICAWI}
   {$DEFINE x-EDIT}
   {$DEFINE x-NISRA}
```

The code highlighted in red shows which Prepare Directives are set to be active for each Datamodel. If a Prepare Directive is not active in a Datamodel then it is prefixed with 'x-' to make it inactive. This works because for example there is no code in the entire questionnaire defined by {$IFDEF x-CATI}.

Once the different Datamodels have been defined, then it is simply a case of using Prepare Directives throughout the rest of the instrument. So different ModeLibs can be set using the code below.

```
{$IFDEF CAPI}
      {$MODELIB CAPI.bml}
{$ELSE}
      {$IFDEF CATI}
            {$MODELIB CATI.bml}
      {$ELSE}
            {$MODELIB CAWI.bml}
      {$ENDIF}
{$ENDIF}
```

Different numbers of Primary Keys are needed for different Datamodels. There is a separate Datamodel for the ICAWI instrument, so it needs a unique identifier for each person, whereas for all the other Datamodels the unique identifier is at the household level.

```
{$IFDEF ICAWI}
   PRIMARY
      QID.Area,
      QID.Address,
      QID.HHold,
      QID.PNo
{$ELSE}
   PRIMARY
      QID.Area,
      QID.Address,
      QID.HHold
{$ENDIF}
```

The ICAWI instrument needs to look up variables defined at the household level and so needs to define the HHGrid as an external file whereas other Datamodels don't need to do this.

```
{$IFDEF ICAWI}
      USES
         HGModel  'USIP5GCAWI'

      EXTERNALS
         GData : HGModel ('USIP5GCAWI.BOI', OLEDB)
{$ENDIF}
```

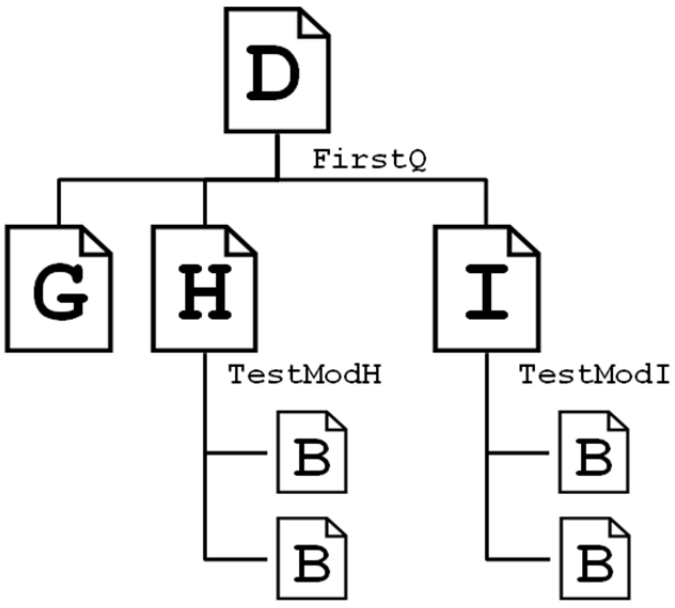
Ideally we would have liked to have put all the additional top level commands like defining the ModeLib, Externals and Primary Keys into a top level include file, but Blaise didn't like this, so we have had create a top level .BLA file with all these commands in. However, once the code has been set with routing using Prepare Directives, then the only code that need to be altered to create a different Datamodel is the initial definition of the different Prepare Directives.

Testing

One of the difficulties with testing a complex and long questionnaire is that in order to test a module of questions at the end of a questionnaire, the tester has to answer all the questions preceding this module to get to it in the first place. Even though testers can become familiar with a questionnaire and know the quickest route through, it can still be time consuming to even get to a module at the end of a questionnaire before even starting the testing. Testing is made even more time consuming if the routing in a module depends on the answers to earlier questions, because the tester has to move backwards and forwards through the questionnaire (sometimes having to go down a different route) to be able to check the routing properly.

We decided that we wanted to design the Understanding Society instrument in a way that would allow testers to skip huge chunks of the questionnaire and get directly to the module that they wanted to test. BUT we also did not want to create separate test and live versions of questionnaires, so we needed a mechanism to allow us to create one instrument which could be used as a testing and live version without any modifications.

The diagram below illustrates how we decided to implement a combined test/live instrument.



At the Datamodel level the first question that interviewers are asked is called FirstQ which on standard NatCen surveys is just a “Press 1 to continue” question.

```
FIELDS

FirstQ
  "INTERVIEWER: You are in the questionnaire for ^NCSerial
  @/@/To update admin details press ^CtrlEnter"
  :(Cont (1) "Interview Mode",
    Test (7) "Test Mode"), NODK, NORF

Password "Enter Test Mode PassCode":0..9999
```

We amended the standard FirstQ question to distinguish between interviewing and testing modes and had a follow up question for testers to enter a 4 digit passcode. The reason for this is that interviewers are inquisitive and someone would choose the test option when they were supposed to be doing live interviewing, so we needed a mechanism to stop them progressing beyond this point. We chose a memorable date as the passcode (note that we didn't actually choose 1939).

```
RULES

IF Password<>1939 THEN
   CHECK ERROR "Incorrect PassCode
   @/@/INTERVIEWER: PLEASE ENTER <1> AT <FirstQ> TO CONTINUE"
ENDIF
```

Then at the household and individual level we had questions TestModH and TestModI which listed the modules within the questionnaire. TestModI at the individual level is shown below.

```
FIELDS

TestModI
"Which modules do you want to test?":SET OF
  (Demog   (200) "Demographics",
   Initial (209) "Initial Conditions",
   Neigh   (212) "Neighbourhood Cohesion",
   …
```

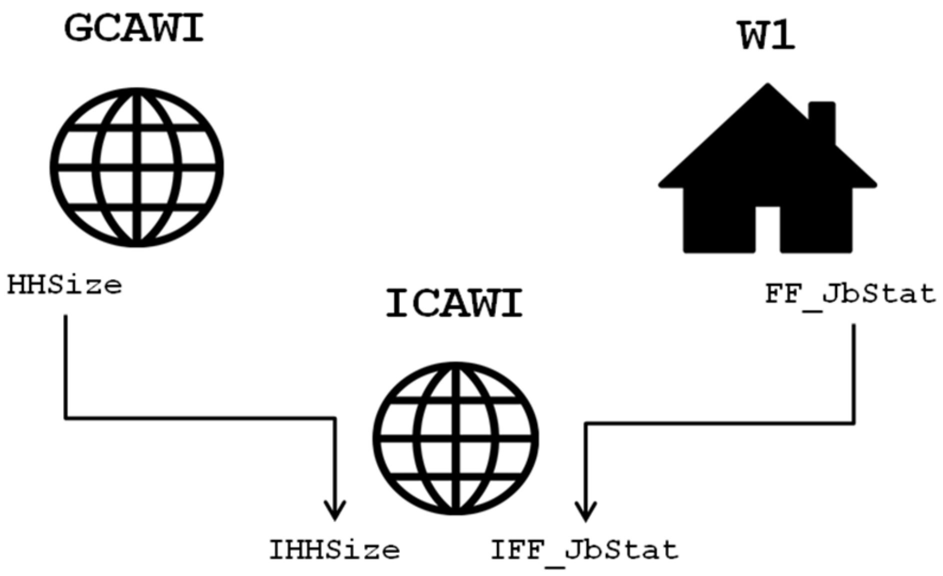
TestModI is only asked in test mode, and then each module within the questionnaire is on route if either FirstQ was the live version or if it has been marked in TestModI.

```
RULES

IF (FirstQ=Test) THEN
   TestModI
ENDIF

IF (Demog IN TestModI) OR (Neigh IN TestModI) OR (FirstQ=Cont) THEN
    Q200Demog
ENDIF
IF (Initial IN TestModI) OR (FirstQ=Cont) THEN
    Q209Initial
ENDIF
IF (Neigh IN TestModI) OR (FirstQ=Cont) THEN
    Q212Neigh
ENDIF
```

Where the routing in a later module depends on answers to a question in an earlier module, it is possible to force the earlier module to be on route if the later module is selected. In the example shown above, if module 212 Neighbourhood Cohesion is the only module selected at TestModI, then both 200Demog and 212Neigh modules are asked (because routing in 212Neigh depends on the answers to a question in 200Demog).

PROGRAMMING TIP: When naming modules use the same names to refer to the block and the module in TestMod. So the Block name is B200Demog (and is called using Q200Demog) and 'Demog' is the answer label in TestModI.

One final complication with the Understanding Society instrument is that routing within modules in the Individual questionnaire can depend on variables defined at the household grid level and on the feedforward data from a previous wave. The best mechanism we found to test dependent routing was to impute a copy of any variable that affected routing into the Individual level block.



In the diagram above household size is defined at the household grid level as HHSize and a copy is imputed into the individual levels as IHHSize. Similarly fed-forward data about the job status was stored in ff_JbStat and imputed into the individual level as IFF_JbStat.

We then needed to set up a mechanism so that the imputed feed forward or household grid variables could be altered at the individual level to test the individual level routing, shown below for FF_JbStat

```
RULES

IFF_JbStat.KEEP

IF (ManFF=No) THEN
   IFF_JbStat:=FF_JbStat
   ManFF:=Yes
ENDIF

IF (Demog IN TestMod) THEN
    IFF_JbStat
ENDIF
```

So IFF_JbStat is initially kept, and then set within routing which ensures that it is only set once and not recalculated. Then if the 200Demog module is selected IFF_JbStat is asked so that it can be amended.

This code then needed to be altered for ICAWI as the ICAWI instrument is separate from the household grid and needed to look-up the values via the external BOI file.

```
RULES

IF (ManFF=No) THEN
   {$IFDEF CAPI_CATI}
         IFF_JbStat:=FF_JbStat
   {$ELSE}
```

```
         IF HGData.Search(QID.Area,QID.Address,QID.Hhold) THEN
            HGData.READ
            IFF_JbStat:=HGData.QSignIn.FF_JbStat
         ENDIF
   {$ENDIF}
   ManFF:=Yes
ENDIF
```

We used this approach to test the IP5 mixed mode questionnaire and found it extremely useful as it enables testers to concentrate on the routing and question wording variants for individual modules, and minimises the time that they spend having to record answers to additional questions.

Summary

We used Prepare Directives to create different datamodels for each interviewing mode, and therefore could use common source code to ensure that the data collected in different modes is compatible. The advantage of this approach is that version control is much easier than having to update source in multiple places, and it minimises the resources needed to develop instruments for different modes as the same code is being re-used.

The questionnaire blocks are structured so that the fields are in a separate file from the rest of the block, which allows researchers to make minor working changes while programmers are working on rules and allows parallel working which is more efficient that serial working.

The key improvement to the development process was the introduction of a modular approach to development and testing. This meant that testing could start as soon as the first module has been completed, and that development and testing processes could progress in parallel which greatly reduces the total elapsed time to develop and test a complex multi-mode questionnaire instrument.