ESRC Microsimulation Seminar Series

Seminar 1: 'Scaling up; scaling down'
London, 2nd April 2009

Developing & Maintaining Dynamic Microsimulation Models at DWP

Sally Edwards

[EDITED TRANSCRIPT]

I'm Sally Edwards, I work for the Department for Work and Pensions and I was originally book, planned to be presenting with my colleague Simon Gault and unfortunately Simon was unable to attend today, so I'll be presenting Simon's slides as well. Unfortunately for you, you won't get to see Simon but also he was going to be tackling the questions at the end, so if you've got any questions at the end, you can write them down, I'll take them home for Simon!

But anyway, moving on, I'm going to give you an overview of what this presentation's going to be about which is primarily rather than looking at the specific details of the models that we have, it's more focused on the development and maintenance techniques that we have adopted. I'm going to look at the objectives that we had when we initially started development of our first dynamic microsimulation model, the standards that we used and why we used them, look at the methods of simplifying complexity that we have used throughout our models and then how we maintain our models, and finishing off with what we feel went well and what didn't and what lessons we learnt from the development and maintenance that we've done so far.

So well I think I might skip over Simon, he's a really nice man(!) but obviously you don't get to meet him, I should have brought a photograph shouldn't I? He could have been a cardboard cut-out beside me, but anyway Simon has been and gone, who am I? I have probably a different background to most of you here, I'm a systems analyst or business analyst, so my background is very much in developing and designing, maintaining large scale IT systems, I've worked predominantly in private sector, so retail, airline industries, banking and life and pensions. So I came to the Department as a short term contractor actually to be involved in this particular model in development and then I've actually stayed on for many years working with it, so I come at it from an IT development background, so that what I've really brought to the project. My current role is I manage the GENESIS simulation model and GENESIS is our underlying dynamic microsimulation model tool. I'm a member of the, Simon and I are both members of the model development unit along with our colleague Howard who has just nipped off, but any questions actually I could pass on to Howard as well, if any questions come through that are specifically a detailed statistics or economics based questions I usually forward them on. But I'm a member of the MDU and we are responsible for microsimulation at the D, department and we have both static and dynamic models and I'll be talking to you today about dynamic models that we use. Our unit consists of around twenty people and that's not entirely working on the models, it's also work on other areas, a demography unit and poverty modelling.

So specifically the dynamic microsimulations models that we have, as I said they're based on our standard architecture which is called GENESIS and the first one we developed was PENSIM2 which simulates a private, both private and state pension income up to 2100, so it's obviously a very long term model and it also includes demographics, labour market, it has got tax benefit in there as well but the primary function is to model the pension system within the United Kingdom.

We also have used the architecture to develop INFORM which an integrated forecasting model for working age benefit claimants, that's a shorter term model, I think it runs to 2020, that's the model that Simon has developed and manages.

We also have some individual benefit forecasting models for different UK benefits.

And finally we have an employment model which is modelling characteristics of employment and it's just recently finished and is currently in validation stage.

We have a couple of static microsimulation models that I won't be talking, a policy simulation model is widely used and an employment and hours model.

One of the key things I want to cover is why we do what we do in the way we do it and we're probably very different to a lot of you in the style of organisation and the fact is that we had to consider when developing, although it is different to what you do, some of what, the techniques we use you may also find useful, I'm hoping that you would think so. But what we decided, when we decide do start the development of PENSIM2, one of the key things was to make sure that the model itself was as simple as possible, so we took a modular approach, so each process or event within the simulation is a separate module and that's been really useful to develop in that way because we can then re-use some of those modules for other simulation models which we have done.

The other factors that are important, that we do have quite a lot of developers that work on each model at the same time within the model development unit and we have a very large staff turnover. We worked primarily with the system economists who stay in post for only one year, so we need to keep the models as simple as possible because during that one year they've got to learn what they're doing, do it, get something out of them and then they move on, so we can't take the approach that someone's going to come in and stay for ten years because generally that's just not the way it works. With the PENSIM2 model specifically we intend, well with all our models really, we intend them for long term use, so looking at PENSIM2 we hope it's going to last for fifteen to twenty years. Now based on staff turnover I would anticipate we're going to have probably twenty five, perhaps to thirty developers during that time working on PENSIM2, maybe 200 to 300 users, so it's really important that actually the code and the parameters are easy for people to understand very quickly. So in a similar way to EUROMOD you know we've tried to make things very simple for people to, so they can pick it up quickly. We have very clear documentation process, training materials and user guides so we find those are absolutely essential because of our large turnover of staff.

The other key factor that's taken into consideration at the outset was that the front end had to be Excel and the code that underpinned the model was going to be written in SAS. Now the reason we went with this decision was we have extensive expertise in SAS at the Department and although I know commonly these models are written in C++, we do not have C++ expertise so we would have had to have brought it in, then we would have run the risk of those people leaving and not having the experience of SAS. So Excel and SAS both widely used and well understood at DWP, so that's the reason we went with that decision.

OK, now I'm going to look at the original objectives when PENSIM2 was built, that's our first dynamic microsimulation model. When I joined, on my first day I was handed a brief feasibility study, it was a nine page document, I just looked it up actually and found it, it was fantastic and it listed these fourteen points that the model had to have. I won't run through all of them but I looked at it and I thought ooh maybe I'll leave now(!) it seemed a little bit of a tall order to me at the time, I felt that to try to include flexibility and efficiency and ease of this and ease of that and speed and user friendly, the whole lot within something that was actually quite complex was going to be quite a tall order. So what we did was we, using this list of objectives as our basis, and actually they're a very fine list of objectives and I think any developer that's starting out with a new model you could take this little list and say this is what I'm aiming towards because I think they're all pretty good. But we felt was that we could not necessarily achieve all of them, so we picked out some of the key things and then looked at how we were going to achieve those.

So we realised that it was actually going to be impossible to satisfy all of the objectives within a single model solution, so we decided to split the model into two separate parts. The first part was the Excel based front end, very similar idea to the EUROMOD model which was a user friendly, transparent code, I put in green the points that came straight from my objectives list, and so we have the code in a very standard format which is easy to understand and flexible, that enables analysts to change parameters and to change the structure of the model without having to understand the underlying code, makes it easy to maintain and also to hand over, I use the term 'easy' perhaps a little flippantly there, it's not necessarily simple easy, but it's easier than if we'd handed over perhaps a standard complicated piece of code. We also allow independent base data, the variables that are input into the model are very easy to change how they're defined, the table structures are all, can be changed very easily as well.

The second half of the model then is a SAS code generator, this is the part that's called GENESIS and this was developed with our own expertise and own resources and it's proved to be a very robust generator and produces very reliable outputs, it was tested very thoroughly up front and then it's been used for all the different dynamic microsimulation models I've mentioned. And that was a very key point, the decision to split it into the code generator and the excavated front end.

So the structure, the diagram I'm showing you is very similar to probably ones you've seen before, the code on the, green boxes on the left hand side are the simulation model that users and developers see, these Excel sheets. We also have the capability of, capacity to input some what we call static codes, so this is code that doesn't sit neatly into the generic Excel sheet format, so users can code separate Excel modules if they need to. These are not used very often, most models, all of our models primarily consist of this Excel parameter sheets, but there is the capacity for the SAS code. Then we have the GENESIS model engine, this takes the parameter sheets and the static code and generates a separate SAS programme, so it's two stages to the yellow box and then it runs to simulation, pulling in the base data which is in SAS and creating SAS post simulation data tables.

Now we're going to move on to looking at the development standards that we used. As we had two separate sections to the development, we had the GENESIS model engine and we had the PENSIM2 analytical time, they were staffed by two separate groups of people, so we actually used different development standards. Now this was a reasonable approach because the GENESIS model engine was developed by IT staff and so we used standard, what I'd call standard IT project development procedures and they're very structured procedures that we use during that development period. The PENSIM2 analytical team was staffed by economists and statisticians and they used a far less structured approach, probably one that you guys are perhaps more familiar with and it was more appropriate to that team because it was a more free flowing environment of ideas passing through and uncertainty as to what to include and not to include and lots of trial and error, that would be how I would term it.

And so I'm going to concentrate on the GENESIS engine development standards that I feel worked very well for us and I think if you're in the development stage of a model, these are things that you really should consider including. We did take a highly structured approach, we used a project management protocol which is based on PRINCE2. Now we didn't use all the aspects of that and that's to do with project planning, analysing the risks, making sure that your scope is very clear and when I talk about planning, each of the components of the model engine would have been estimated and determined whether or not they were worth developing, if something was going to take three months to develop, do we actually really need it? And that enabled us to be very careful with what we did include and didn't include.

We use two types of documentation, I'm quite strong on documentation, I do quite like to have documentation, Howard's smiling away here, Howard's the kind of opposite and I'm drawing him into my little documentation world. We have development documentation which is produced at each stage of the project, so it clearly is used, it explains what the next stage is about, it's really important for people that are IT based when they don't really have a clue of what the application, some of the terminology used in the application means. So it defines what the terms are and what the calculations are, so what is polychotomous alignment or that sort of thing, they wouldn't understand. So it's very clear definitions of what the coders are supposed to be produced. And we, with that development documentation it wouldn't be updated, you wouldn't go back and update it further down the line because it's used purely during the development. Then there's key project documentation that as I run through I'll show you the things that we felt were key, that we've maintained and we've kept up to date with each new release of the model we still have this documentation in place.

We used development processes that were simple process diagrams that provided clear guidelines that are particularly useful for new staff that come in, they don't know quite what they're supposed to be doing and we've got diagrams to show them what you know, the procedures that we're using.

We wrote a detailed requirement specification to explain the functionality of the model engine and this document effectively translated what I felt was economists' language into an IT language or a language that IT developers could understand. That detailed requirements document has been maintained through the different releases, so it's a living document. And that was signed off so that the developers actually knew what they were developing and changes weren't just thrown in during the course of the development.

We had clear design diagrams and documentation, these were just used in development stage, we had a strict trench control process so that any changes that were thought up during the development were impacted and assessed or you know accepted/rejected, we continue with that change request process and we find that really useful so that we don't just go in fiddling about with the code basically, we're quite strict about it. We also have a problem log process as problems are picked up, they are recorded, assessed, again sometimes they're not actually included, if the problem is seen as being trivial but it takes a long time to fix then we would perhaps just say OK we've got that problem we'll live with it.

The validation process we use a detailed, we developed a detailed test plan which was based on the requirements document and that was used to ensure that the model engine satisfied all the requirements that had been specified.

We also have a test pack so that whenever we make changes to the engine we run through that test pack, we run through to make sure that we haven't adversely affected something else. And we found that particularly useful because it's really easy to tweak something and cause something else to stop working.

We have a user guide, it's possibly not as good as Holly's good bedtime reading(!) but it's something you refer to occasionally, it's very important to have. We have quite extensive training materials, because we have such a large staff turnover we run, I probably three or four, well probably more training courses each year for groups of staff coming through.

OK, well the pit and chain (?) model development did not follow that structured approach, they followed generally, so what have I said here? Yeah it wasn't possible to develop the detailed requirements up front and so the effort during that stage was predominantly spent on regression analysis to try to work out what the appropriate parameters were to put into the model and then those results from the regression analysis were reviewed to try and determine what to include and what to use for assumptions. And the results of, what have I said, yes because we used these standard GENESIS templates for actually inputting all the parameters, the PENSIM2 analysis team were able to quite easily code up their parameter sheets rather than if they had to build a traditional dynamic microsimulation model.

Simplifying complexity, I wanted to talk a little bit about the techniques that we use to simplify complexity and I could see a lot of similarity between what we do and what EUROMOD has done. We have a very strict no hard coding rule and that has been adhered to very thoroughly, so we have complete flexibility, because it's a code generator, the GENESIS engine is a code generator, we have all the structure within the parameter sheets, so we, the order of the events that are processed or simulated, it can be, models can be taken out and put in easily. The data structures, the tables and the relationships between the data, so it's easy to set up say you can define partnership or parent/child or, well any relationship you want to set up you can set up within this structure. So we could set up say company or various different, what I'm trying to say is the simulation model doesn't have to be around people or benefit units or households, it could be around anything else, it could put that into the structure of the data within the parameter sheets. Obviously we hold all rates, dates, indexes as parameters, also all the regression equations are held in parameter sheets, all the selection filters, one of the things that I particularly hate about programme code is sometimes you may see some what appears to be a clever piece of code that says is, if and then you get this like whole string of variables, this and that equals this and that and then goes over the next line and then do and then you have a whole load of other complex. So we've avoided that completely, every little component of an 'if' statement is in its own separate cell. Now that means that instead of, that sort of code can look very clever and the person that develops it can write 200 lines in 2 line, that's fantastic for that person but for the other 250 people perhaps that pass through looking at our PENSIM2 model, that doesn't work. So each little component of an 'if' statement or a 'then do' this, this and this is in, each component is broken down into one single cell so that you've got a very simple way of understanding and reading the code. And I think Howard would agree that that has worked particularly well for us.

The SAS static code does allow for flexibility particularly if we want to add new cases, so new people want to come, we call them in flows into the benefits system perhaps, things like that we use a static code for. I have mentioned before it's very easy to add or remove modules, providing obviously a variable that's created within a module is then not used in another module because obviously that then wouldn't work so well.

So I think I've already covered the fact that we use simple sheets, I haven't got examples of them here but we use standard templates and the whole idea is that it's easy to understand for everybody to, that comes after you.

We also have used data access routines so that accessing an item of data isn't written in a complex way and this includes the, the default will be that you'll use the current person's current year's variable value, but you have the facility to use the value belonging to something related to that current person, I'm saying person but it doesn't necessarily have to be a person. So you might say oh we want actually to use their partner's information so you could use the partner or a spouse or a father, mother, youngest child, oldest child, something like that if you want, just very simply with the three characters and then it goes away and generates the code for that. Or you can use, or, and you can use a data manipulation routine,

there are various ones that are there within the code generator and the example I've given you here is the maximum value during the past ten years, so if you put in MA10 it just returns maximum over the past two years. So an example given is MA10 PAR PA salary and what that would return would be the maximum value during the last ten years of the salary for the individual's partner taken from the PA table. So if you were actually coding that yourself, that could be a little bit more complicated, well would be more complicated than just writing that variable there, but the code generator goes away and generates the necessary code to do that and that's certainly been very useful for us.

And one of the points I wanted to sort of bring out again is the, we always make sure that we try and keep the equations, the filters, the process flows defined in a simple, standard format that's unambiguous so that it's easy for models to be maintained. We have lots of, the words used are we try to use words that are easy to understand and make sure we define whether something's weekly, annual, that's the sort of thing that I think often goes amiss and so we're quite careful with that.

Just moving on to the maintenance protocol that we've followed, and this, I think this is where, Howard was on the analytical team during the development weren't you Howard? And so Howard does prefer a more free, there's Howard, Howard do you want to wave? This is my colleague! Now you did always prefer a more free format development style, but I think I'm beginning to win him round, so he's now adopted some of the maintenance protocols that we've put in place. So for example change management, every single change to one of our models goes through a very formal change control procedure, so each change that's requested by a user is impacted, recorded on a register and determined whether or not it's going to be put in. And just, when I've got a chance, I'm going to run through this quite quickly. This is really important so that we don't end up putting changes in willy-nilly and that we keep a clear track of what changes we do put in and how we put them in, particularly some of the changes are repeated, so changes to assumptions repeated each year or each, every period, so it's really important that we know how we did it so we can go back and do it again with a new set of people, they've got a record of how to do it. We've had a lot of questions that have arisen much later and said well why did we do this, what's all this about? And then we've got all the documentation there for changes that have gone in.

Similarly we've got the project management process that's followed, so every problem that is identified is recorded and again impact assessments carried out and we maintain a register, we maintain a log of all the problems and details of whether or not they're fixed and how they're fixed.

User model, we have as I said several dynamic microsimulation models, they fall into two categories really, we have the models that are maintained and owned by the model development unit and those that are handed over to the users who actually then, having developed them ourselves, the users then own and maintain them. The ones that are owned and maintained by the unit, we have a user group and/or a steering group that consists of representatives from each of our user teams and the purpose of these groups is to make sure that we have regular meetings, so we agree the content, priority and timing of any changes there are made to the models. And that's, we've found that to be a really useful forum, PENSIM 2 particularly we meet once every three weeks, nothing goes into the model without the group's authority. So as I've always mentioned, yeah so the user group reviews changes and problems and there's always a key sponsor assigned to each change and that person ensures that it's accurately specified and signs off the testing. Our users also may have a separate user forum and PENSIM2 does and they present analysis that's carried out on the output and explain how the results are used.

I'm probably running a bit short of time aren't I?! I'm just going to very briefly, another thing that we do have is a very strict release management process so that when new releases are put forward, are implemented, that's every six to nine months we implement a new version, each release will contain one or two very large components and a bundle of small change and problem fixes, all bundled into a specific release which is agreed with the user group. And when new release is built, we add each change or each fix separately one at a time so that we can see the impact and this is really important for us because what we started off doing was we'd say oh we're going to put in this release and put in these twenty changes and oh look how different it is, and someone will say well why has that gone shooting off there? And because we've put twenty changes we couldn't figure out why it had gone shooting off there. So by putting each change in one at a time and have a different version of the model with each change, we can say actually that went shooting off there when we put this one particular change in, is that really what we want to do? And it enables us to really get the buy in from the user community to say actually we do, or perhaps we don't want that change to go in. It's quite a painful lesson to learn, this technique.

We also maintain a clear audit trail of all changes and all previous versions of the model are available to use, so if someone wanted to go back to the pensions reform version they could do.

Do I have time for what worked well and lessons learned or should I stop there?! I'll just quickly go on!

Now just the, going back to my original list of objectives, the ones I've highlighted in green are the ones I feel we did satisfy, so that's the flexibility, robustness, reliable output, availability via a desk top, done with our own expertise and independence of base data, those I feel we satisfied very well. The ones that were partially satisfied were transparency, user friendliness, done with their own resources, because we did have two or three contractors in, ease of handover and ease of maintenance, I think with any model of this type especially PENSIM2 it's not going to ever be easy to hand over and maintain. The ones that I feel proved more tricky for us were speed, by having a code generator we really traded that off against speed; efficiency and use of memory, again the code generator is not necessarily efficient; timeliness, the original project plan was much shorter than the actual development timescale.

So overall we feel that the framework has been successful and easy to use. GENESIS architecture has been remarkably robust, I mean if anyone was interested in potentially using it for a dynamic microsimulation model of your own then we would be interested in talking to you about that, we'd be happy to explain to you how it works and with potential perhaps for collaborative working with you. We find it's quick to develop microsimulation models as shown by the fact we've actually developed quite a few since our original one. And we have an automated documentation facility which checks errors and provides automated documentation showing how the modules inter react.

Lessons learned, we tried to make the model too complicated, particularly the analytical side, we should have started with a simple set of modules and then added complexity for a phase two development and we really were over ambitious initially. Some of the modules, particularly a labour market process was way too complicated. There wasn't sufficient documentation produced during the development and the actual GENESIS engine code is quite difficult to understand and modify. Fortunately it doesn't require modification because it is robust but that is an area of concern. Some of our original functionality was over ambitious.

So, right any questions, this is Simon's slide, so (LAUGHS) and I think I've run over my time, so I don't know if you have any specific questions for me now or whether you want to, if you wanted to ask questions afterwards, Paul, it's your show, it's up to you.

QUESTIONS

Male question – You were saying how you, when you're adding bits to the programme you kind of did it one at a time. Obviously if a problem's occurred because of the individual components that works wonderfully. I was wondering what ?? with the, is it an interaction effect between the other parts, so you kind of implement a new process, that works fine, implement another one and then it messes up.

Sally Edwards – Right.

Male question – But if it was the other way round, the first would have been fine, sorry the other one would have been fine …

Sally Edwards – Like on it's own?

Male question – … because there's interaction ??, how would you kind of like work out you know …

Sally Edwards – Do you want to take that question Howard, yeah Howard would like to take the question.

Howard Redway – what we do in practice is each of those changes is we test them initially on the current released version which then when we assemble the actual net release, when we build stuff, but the reason we did it was precisely for the reason that there might be interactions which we wouldn't otherwise have noticed. So yes we put them in intervention but we recognise there is an interaction it's the second that's posing the problem you don't take that one out, but you might have to go back and revisit the first one to check back because it could have been the first one or the second one.

What we try and do is assess when there are going to be interactions, so within our own actions, so if we change the proportion of people who are getting defined benefits schemes and then we change the actual performance of the schemes those two interact so we anticipate that and we do in as consecutive ?? in places and that is one of the criteria, that where there are interactions, we try and build process as a series of sequential steps.

Male question – OK.

Female question – When you are talking about you can ? from any two other models, for example popular module from another body who ?? another model.

Sally Edwards – Yes.

Female question – Have you used the others, other ? machines ?

Sally Edwards – Have we taken modules from other – not that I'm aware of, no (talks together with Howard) not from other non GENESIS models. We've re-used all the demographic, we have a whole set of demographic modules, partnership, fertility, separation, that we copied from one model to another model and then as I say we hadn't, didn't have to re-use that. But we haven't take, copied anything, because they'd have to be, it would have to be put into the format of the parameter sheets but …

Howard Redway – The point we've come to I think is a module which Cathal O'Donoghue is developing at the moment which is an international migration module and he's already built that in one of his models for Ireland, he actually used that as a template for putting into GENSIS We've had a lot of contact with other model developers, so where things have, we sort of relate them and we know from them and we haven't got the data somebody else's logic and a modest amount of tweaking integrate well into GENESIS

END OF RECORDING