

# TRACEABILITY IN EVIDENCE-BASED POLICY SIMULATION

Ulf Lotzmann  
Maria A. Wimmer  
Institute of Information Systems Research  
University of Koblenz-Landau  
Universitätsstraße 1, Koblenz 56070, Germany  
E-mail: {ulf, wimmer}@uni-koblenz.de

## KEYWORDS

Model analysis, Model acceptance, Policy-modelling, Agent-based simulation.

## ABSTRACT

In the field of policy modelling, a trend to growing complexity of simulation models can be observed. One of the reasons for this development is the fact that for many policy cases of practical interest there are no theories available from which "simple" simulation models could be derived. Instead a sometimes vast amount of information - scenarios describing stakeholders views, documents providing background information - has to be taken into account by the simulation model. Such models are usually referred to as "evidence-based models". In order to foster the use of the simulation method for decision-making, a constructive and structured approach for converting the evidence base into a simulation model is advised.

This paper outlines such a novel approach, which has been developed within the OCOPOMO project. This approach basically consists of a policy development process specification, and a software toolbox supporting this process. Main focus of this paper are simulation related aspects of both the process and the toolbox, with the aim to demonstrate role and benefits of traceability along the process of modelling, simulation and result analysis.

## INTRODUCTION

Since the early days of Computational Social Science, actual programming of simulation models for social systems has often been done based on intuition. The way model programmers formalise agent behaviour e.g. by writing rules on the basis of evidence gathered by reading documents or interviewing relevant stakeholders has sometimes the appearance of a "black art" (Edmonds and Wallis 2002).

While for examining emergent (or other) effects in rather small (and sometimes theory-driven) models this approach has regularly proven to be successful (refer e.g. to Epstein and Axtell 1996), in the case of rather complex policy models it can be regarded as questionable (although there is no implication that this approach systematically fails here; examples for successful models can be found e.g. in Barthelemy et al. 2001; Alam et al. 2007). This is primarily due to the following facts:

- typically, no (social) theory exists from which such models could be derived;

- instead, such models are constituted by diverse and possibly very large evidence bases;
- as results from such models are intended to have impact on some application domain, the process of validation may have to be extended to a non-scientific community, e.g. to policy makers and other stakeholders.

In order to find a remedy, research activities have been started with the aim to provide structured approaches for more coherent and, in particular, more provable inclusion of both the evidence base as well as the knowledge and experience of different stakeholders. One of these approaches is investigated in OCOPOMO (Open Collaboration in Policy Modelling), a project co-funded in the 7<sup>th</sup> Framework Programme of the European Commission (see <http://www.ocopomo.eu/>). Core of the approach proposed by this project is a process that guides the development and validation of evidence-based models with stakeholder participation, and a toolbox supporting this process (Wimmer et al. 2012).

This paper is directed to the part of the process that deals with the actual simulation models. It basically shows how (and by which technical means) understanding and documentation of models can be enhanced and, finally, how integration into the overall process can be achieved. Two perspectives on simulation models are important in this context:

- the perspective of the model developer, who is interested in traceability in order to understand or to keep track of the structure of the simulation model code,
- the perspective of the stakeholder not directly involved in model development, for whom provenance is essential in order to gain confidence in model results (and the simulation method as such) by unveiling the "black box" simulation model.

The paper is structured in a way that firstly the overall OCOPOMO process is outlined, followed by a description of the trace concept. Subsequently the paper presents some aspects for the realisation of the concept, and finally focuses on a demonstration with the aid of one of the pilot case simulation models.

## THE OCOPOMO PROCESS

In OCOPOMO, a novel approach for engaging stakeholders in policy development is conceptualized and implemented (Wimmer et al. 2012). Stakeholders are collaboratively involved in the development of scenario texts relevant in the context of a policy under discussion. In this regard, the term policy is referred to strategic areas of complex

decision-making with various stakeholders having potentially diverging interests. In OCOPOMO, public policies are investigated and modelled such as renewable energy policy of the Kosice region in Slovakia, housing policy of the city of London or the distribution of structural funds in the Campania region in Italy. The overall OCOPOMO policy development process consists of six phases:

1. An initial scenario describing a policy prospect is developed by policy makers or domain experts.
2. Stakeholders are involved to generate scenarios of potential policy aspects on the basis of the initial scenario, which are complemented with background documents to evidence statements in the scenarios.
3. The policy case is then conceptualized and
4. modelled by experts.
5. Simulations are run to generate outcomes.
6. The results of the simulation are exposed to the stakeholders, who compare their scenarios (of phase 2) and the simulation outcomes in order to either update their scenarios (and start another cycle with phase 3) or accept the insights from the simulation and agree that these are consistent with the inputs the stakeholders provided in phase 2.

The project develops an integrated ICT toolbox to support the policy development process and, in particular, a smooth transformation of policy inputs by stakeholders to inform formal policy models. The ICT toolbox consists of

- a participation platform that enables stakeholders to collaboratively develop their scenarios, to upload and share background documents, and to discuss among themselves about views and issues of a policy (supporting process phases 1 and 2);
- a consistent conceptual description (CCD) tool, which enables policy modellers to develop a conceptual model of a policy domain. The CCD tool supports annotation of scenarios and background documents and therewith keeps track of provenance (supporting process phase 3);
- a simulation tool (DRAMS), which supports policy modellers in programming and running policy simulation models (supporting process phases 4 and 5);
- a CCD2DRAMS transformation tool, which supports the semi-automatic transformation of conceptual policy model constructs into code of formal policy models (link between process phases 3 and 4).

A more detailed description of the OCOPOMO policy development process and the respective tools available in (Wimmer et al. 2012).

A key element in the novel approach of OCOPOMO is to keep track of how policy inputs by stakeholders feed into policy simulation and therewith convey provenance. This way, traceability and transparency in policy development are supported.

## TRACES IN SIMULATION MODELS

As pointed out in the previous section, OCOPOMO has developed a policy development process and ICT support toolkit with features to enable keeping track of inputs in form of evidence-based scenarios and background information provided by stakeholders. This way, provenance of arguments by stakeholders is ensured. In the Oxford English Dictionary, provenance is defined as “the

origin, or the source of something, or the history of the ownership or location of an object, especially when documented or authenticated” (for a more detailed elaboration of definitions of „provenance“, see Munroe et al. 2006). In OCOPOMO, the main purpose of provenance is gather evidence as to the views and background information for the creation of a public policy.

Provenance is thereby ensured through the establishment of traces and links between sources of information (the scenarios and background documents) provided by the stakeholders of a policy domain, and the simulation models developed by policy experts. The links show the evolution of formal elements of a simulation model from the description of the real-world section (the scenarios and background documents, i.e. informal artefacts) which is subject of the model. Usually, not all formal model elements have counterparts in the narrative descriptions/documents. The OCOPOMO process foresees enrichment of models by policy experts to complement and complete the formal simulation model.

To ensure provenance, the links from elements of a simulation model to provenance are stored in the CCD tool. Therewith, traces are enabled, which facilitate navigating from simulation outcomes back to the simulation model back to the conceptual model, and finally back to the provenance documents (i.e. scenarios and background documents). The traces established help stakeholders to better understand simulation models of particular policy perspectives.

Traceability is a key element in ensuring openness and transparency in the OCOPOMO policy development process. Therewith, good governance principles are implemented in policy modelling. Traces and provenance are also important auxiliary means for policy modellers, by helping the experts to better understand complex interrelations of policy aspects and how informal data elements feed into a formal model (provenance). Hence, traces are a basic instrument for model exploration and easier understanding of the structure of a simulation model, for simplification and visualisation.

The next section outlines the concept of a newly developed declarative rule engine as technical basis for maintaining the traces.

## A DECLARATIVE RULE ENGINE FOR AGENT-BASED SIMULATION MODELS

As elaborated in (Lotzmann and Meyer 2011), a rule engine is a software system that basically consists of a fact base, a rule base and an inference engine. As part of the OCOPOMO toolbox, DRAMS (a Declarative Rule-based Agent Modelling System) has been developed as a distributed, forward-chaining rule engine. It equips an arbitrary number of agent types with type-specific rule bases and initial fact base configurations.

Heart of the inference engine is the data-driven rule schedule, an algorithm deciding which rules to evaluate and fire at each point of time. In order to decide which rules to evaluate for which agent instances, the schedule relies on a data-rule dependency graph. This is constructed once at the beginning of a simulation from all specified rules and initially available data; the graph does not change unless rule bases are modified. As to detecting fact base

modifications, the schedule keeps track of all fact base operations.

For typical simulation models, DRAMS only takes care of agent deliberation abilities, while the environment is “outsourced” to an external simulation tool. Since DRAMS is implemented in Java, in principle any Java-based simulation tool can be used for this purpose. At the moment, DRAMS provides interface to facilitate the integration with Repast (North et al. 2006).

The data-driven rule scheduling mechanism employed by DRAMS inherently supports the generation of traces. The following chapter will describe the process of creating the traces, together with the necessary prerequisites.

### Annotating Model Code

In the path of passing link information from provenance data to simulation results, the segment between model code and raw simulation outcomes can become quite complicated. This is primarily due to the fact that the means for processing initial data configurations within a simulation run can be very complex, and are influenced by manifold factors. When speaking in terms of agent-based declarative simulation models, each “individual” in an agent population of arbitrary size carries up to thousands of facts, which are processed by hundreds of rules. Under such conditions it seems hardly possible to reach the goal of finding adequate ways for extracting useful information in terms of understanding structure and behaviour of the model.

An approach for solving this issue has already been sketched in the first sections of the paper: a conceptual model, the CCD, is developed prior to programming the simulation model. This CCD incorporates and specifies all the crucial elements for the simulation model, but abstracts from technical necessities for making a simulation model “run”. With other words, only those simulation model elements are incorporated in the traces which have counterparts in the CCD.

As a starting point, these crucial elements have to be marked in the simulation model code by link annotations. Each element in the CCD is equipped with an unique uniform identifier (UUID), and this link is attached to the related simulation model code element (see Figure 1). In order to comfortably maintain these link annotations, tool support is advisable. For the OCOPOMO toolbox, a model-to-text code generation tool (CCD2DRAMS) is provided for this purpose (Scherer et al. 2012). This code generator adds link annotations to all generated elements, in particular for agent classes and instances, fact templates, facts and rule stubs.

The model programming then consists mainly in creating the code around the generated parts, on the one hand by filling in the complete logic into the rule stubs, on the other hand by providing “glue” code in-between the generated code.

Such development approaches are usually not sequential but rather cyclic; this means that during model programming missing crucial elements are discovered, which then have to be added at the CCD level. The CCD2DRAMS code generator takes care not to overwrite already elaborated rules - situated in a user code section of the source file - when re-transforming a modified CCD.

During parsing of model code by DRAMS, for each element with a link annotation a so called trace tag is generated. This is basically a small data container object, storing the link UUID, and bringing the possibility to define different kinds of neighbour trace tags. These are used as nodes in the generated trace graph, as shown in the next section.

```
/*Object: HouseholdCharacterstics
*@link _j2N48MdEeGILbUlSo0ubw
*/
(deftemplate Household::HouseholdCharacterstics
  (uuid:UUID)
  (instName:String)
)

/* Action: calculate the household heat demand
*@link _KSe8AK8GEeGItqqK0ME1Yw
*/
(defrule Household::"calculate the household heat demand"
(
  // insert LHS clauses here
=>
  // insert RHS clauses here
)
```

**Figure 1** Generated DRAMS code with a fact template definition (`deftemplate`) and rule stub (`defrule`) with link UUID annotations (`@link`)

### Creating the Traces

In DRAMS, the creation of the trace information is a ancillary procedure of the forward-chaining rule engine process. A simplified description of this algorithm comprises five steps:

1. At the initial state of the rule engine - no rule has fired - a number of fact templates, partly concrete facts for the templates and rules are present. For subsets of each of these elements (for which CCD elements exist), trace tags are attached.
2. When the rule engine is initiated, it firstly checks which rules might fire with the given set of facts. The LHS's (left-hand-sides, specifying the conditions) of these rules are evaluated, and for each successful evaluation, the RHS (right-hand-side, describing the actions) is triggered.
3. The RHS processing starts with checking, whether at least one of the facts evaluated by the LHS (and, hence, determining the data basis for the RHS execution) is attached with a trace tag. If this is the case, a new trace tag for this particular rule firing at the current simulation time is generated, using the information (link UUID) stored in the rule trace tag, if available. All trace tags for the LHS facts are then incorporated as predecessors of the rule firing trace tag.
4. The rule firing trace tag, or the rule trace tag, according to disposability, is then passed to all RHS clauses.
  - a. A clause for asserting a new fact to a fact base generates a new trace tag for this fact with the trace tag delivered by the rule as predecessor.
  - b. A clause for writing output data (e.g. a log record) passes the trace tag to the output processing facility.
5. When all rules have fired, the newly created facts constitute the new state of the rule engine, and the processing continues with step 2.

If a rule producing a log record or (numerical) outcome data is either equipped with a trace tag, or when any of the elements that lead to firing this rule have had a trace tag attached, then a "connector" trace tag is available as result of the algorithm. This connector trace tag can be seen as root node of a directed acyclic graph, which covers all the important steps for creating this log or outcome.

After finishing a simulation run, a potentially very large graph data structure is available, holding information about traces for all relevant generated facts and simulation outcomes. With this additional data, questions about the cause of a simulation result can be answered with very low additional effort. In order to answer such questions and to perform more sophisticated analyses, the information from the graph must be further processed and/or stored in adequate ways.

### Processing Simulation Outcomes

DRAMS brings a plugin interface with which any kind of output processing facilities can be embedded. Such plugins can either write files of a particular format, or can serve as an adapter to an analysis or visualisation tool.

A selection of implemented plugins for DRAMS is described in the following compilation, each representing the trace information in different ways and formats:

- Plain Text / CSV - these two plugins write log records or numerical outcomes in plain text files or CSV tables. In both cases, the trace information can be attached as lists of UUIDs, optionally attached with additional information (e.g. name of the element belonging to the UUID). For these formats, the usage of the UUID is usually restricted to manual handling.
- XML - this plugin creates XML files containing numerical or textual simulation outcomes. These XML files are processed by another component of the OCOPOMO toolbox for creating traceable logs or different types of diagrams. The trace information can be added to the values in different levels of details, e.g. as
  - a simple collection of UUIDs (as for text output above),
  - a diary, showing the involved UUIDs for the different simulation time steps, or
  - a complete XML representation of the evaluation graph.
- Model Explorer Tool - this plugin provides an UI for displaying the simulation log, and by selecting a log entry the related trace information is visualised and can be analysed in various ways. This tool is subject matter of the following section.

### ANALYSING SIMULATION OUTCOMES AND USING THE TRACES

One of the simulation models realised as a pilot case in the OCOPOMO project deals with developing a sustainable long-term strategy for use of renewable energy resources in the Kosice Self-governing Region (KSR), Slovakia. "The regional government is interested in a better understanding and identification of potential impacts of policy alternatives in support and exploitation of renewable energy resources, including their impact on employment, environment,

financial implications of investments, and a wide range of other related issues" (Scherer et al. 2012).

The version of the model available during the development of the Model Explorer Tool is aimed to explore the effect of rising energy prices and potentially stagnating (and on average rather low) household income. In order to save costs for heating energy, the households can either buy a new (additional) heating technology, insulate the house/flat, or decrease the room temperature. The model comprises 48 Household agents organised in a Household Association, three Heat Producer/Distributor company agents, a Regulatory Office agent and a Government agent. The households are distributed among six buildings of two kinds, namely three one-family houses and three blocks of flats. There are two different heating technologies available, and the houses or flats can be equipped with dozens of combinations of 19 different types of building and insulation materials. Altogether there are around 180 object, actor and relation instances in the CCD. The simulation is expected to show the change of average room temperatures and the investments in heating technologies and insulation over time.

Particularly important for this model is an evidence-based and, thus, realistic initial set-up of the simulation world, consisting of buildings constructed of meaningful combinations of different building materials, and the distribution of households among the buildings. This initialisation procedure will be used in the following paragraphs for demonstrating traceability to the evidence base.

The part of the conceptual model dealing with these issues is shown in Figure 2. A household agent is situated in a building that is built of several components and that is located in a city. A list of building components describes parameters of these components in regard to heat resistance, a value that can be interpreted as a measure for insulation capability for each component and building. The household association agent is in charge for maintaining this list and for calculating the actual values, in order to enable households to determine their own expenditure for heating energy.

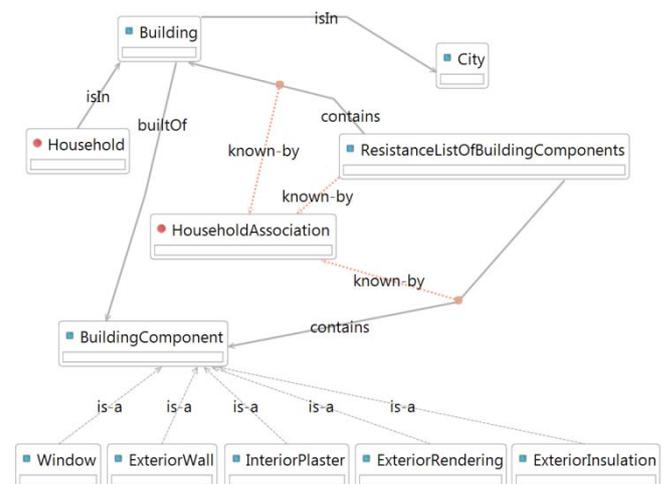


Figure 2 Extract from the Actor Network Diagram of the Kosice model CCD

The initialisation steps are conceptualised in the action diagram in Figure 3. Firstly, the resistance values for both construction material and windows are calculated. This

information is then used to calculate the energy information for buildings. With this information the households are in a position to derive the heating energy demand. For agents, buildings and all related initial data, link annotations are attached to all CCD elements that are regarded as crucial by the modeller, i.e. to be traced back from (intermediate) simulation results applying the Model Explorer Tool.

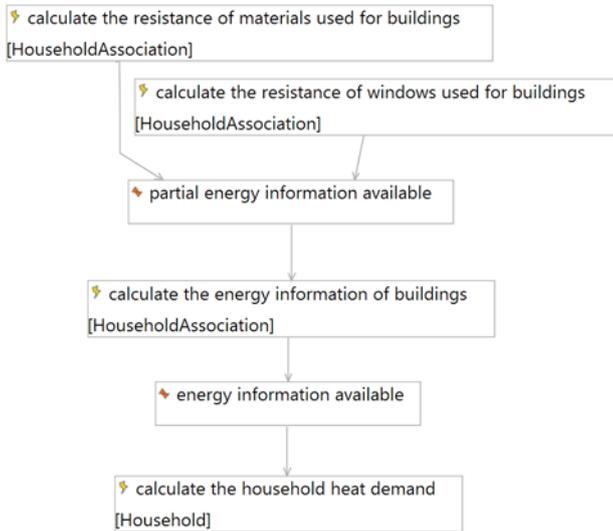


Figure 3 Extract from the Action Diagram of The Kosice model CCD

### Model Exploration

The process of exploring a model subsumes different tasks with the aim to gain insights in structure or behaviour of the model. Both aspects usually are interrelated. While it is

mainly of interest to understand the behaviour of the model, it becomes necessary to shed light on the structure of the process leading to this behaviour. Furthermore, different users require different levels of abstraction of the presented information. While the model programmer wants any detail of each rule that fired with any data processed by the rule, a decision maker (are other stakeholder) for instance might only want to see an overview of different fact types and how these are related to the actors dealing with facts associated with those types. Hence, a comprehensive Model Explorer Tool has to take all these aspects and perspectives into account. At the current stage, the model programmer perspective has been implemented, while possible designs for further developer and stakeholder perspectives are currently investigated and discussed (and not part of this paper).

Figure 4 shows an overview screenshot of the Model Explore Tool UI. On the right side a list of log records produced by the simulation is shown. The user can select one or an arbitrary set of entries, which then are further processed by the tool.

The first stage of processing is to create an internal data representation of the trace information attached to the selected log entries. As a "natural" approach to present this information, a decent visualisation of the trace graph is displayed left to the log view. The graph shown in the screenshot is the visualisation for a log entry generated in time step 1.0 after calculating the heat demand of a household living in a block of flats, as part of the model configuration described above. The green ovals on the left are all initial facts for building materials, several relations (e.g. in which building this particular household is located) and several other facts (e.g. the current month and year), processed during deriving this particular simulation

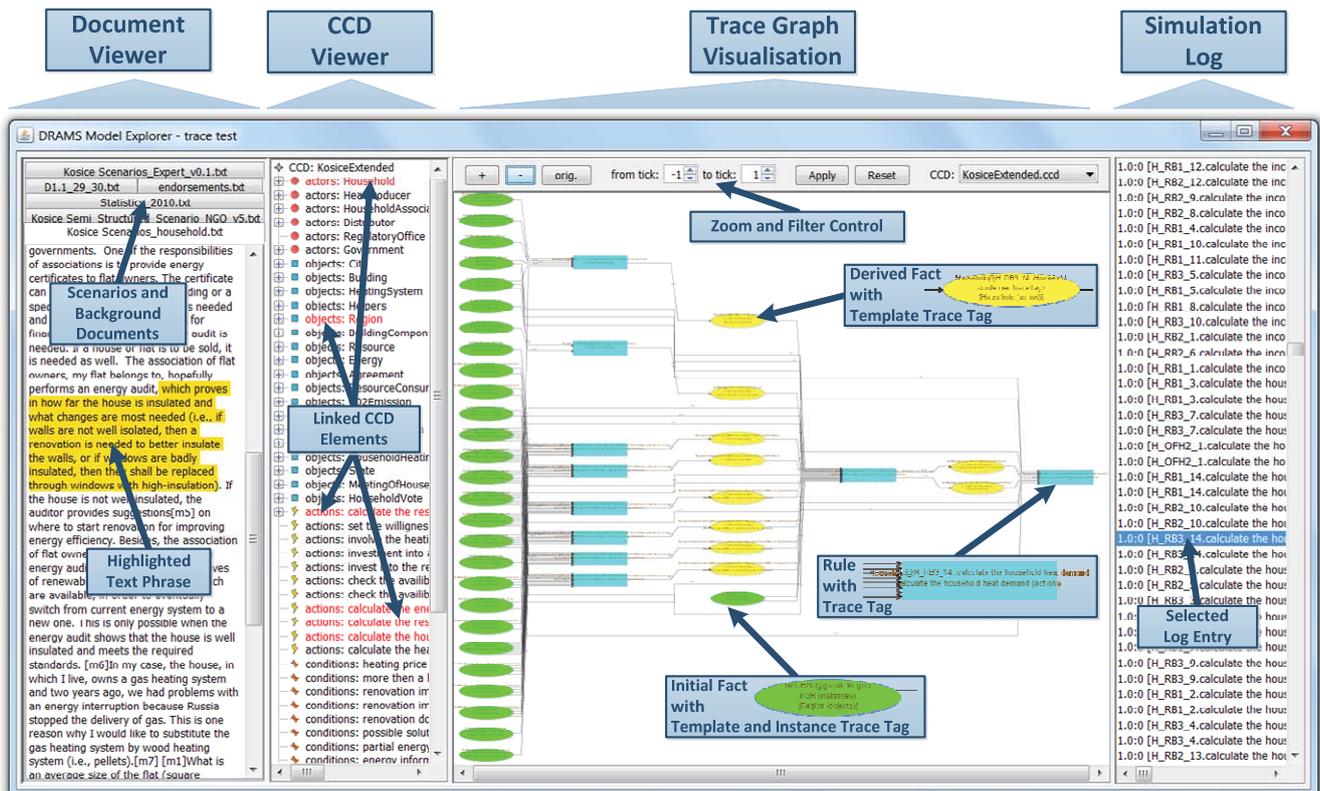


Figure 4: Model Explorer Tool GUI

outcome. The set of blue boxes on the right side to these facts are rules related to calculation of heat resistance, resulting in new facts with "partial energy information" (ovals with colour yellow, since these facts are asserted only during the simulation run). The single box further right represents the rule for calculating the final energy information (pair of yellow ovals), in order to calculate the household heat demand (single box on the right side of the graph visualisation).

All UUIDs from the trace tags contained in the graph are then used to identify the related elements of the conceptual model. These are highlighted in the CCD tree representation left to the graph. Finally, the text phrases attached to the CCD elements are accentuated as coloured text annotations in the editor views on the left-most side of the UI. For each document of the evidence base, an editor tab is present. So, for each simulation result the provenance information is disclosed. Performing these steps in the sequence as described above means following the phases of the OCOPOMO process in reversed order.

An important feature for model programmers is illustrated in Figure 5 (which focuses on the rule producing the final heat demand with two of the pre-condition facts, as a detail of the above-shown graph). Clicking on any element in the graph visualisation opens an info box window, giving details about the selected element. In the figure, an edge from the global fact `BuildingEnergyInformation` to the Household agent rule "calculate the household heat demand" is selected. The info box displays (among others) the source code of the rule, with the particular clause highlighted that retrieved the fact at the other end of the edge. Info boxes for facts inform about the concrete

content of the fact processed by a succeeding rule, while boxes for fact templates reveal the facts available for the template.

The graph visualisation can also be filtered. Besides restricting the visual elements to any time interval, it is e.g. possible, to pick out a single node and only show

- the direct neighbours of the node,
- the sub-graph of all succeeding nodes, and
- the sub-graph of all preceding nodes.

The UI of the tool can also be employed for visualising the traces on the stakeholder perspective, as information on how the model used the evidence base for generating particular results (i.e. the relevant relations between log entries on the one side, and concepts and phrases on the other side) are already shown. Only the graph view ought to be replaced by a visualisation on a more abstract level

The Model Explorer Tool can be applied for supporting the analysis of simulation results and to feed back to the conceptual model.

### Linking Results to the Evidence Base

Simulation outcomes can be either numerical data, showing the change of measurable values over the course of time, or qualitative logs, telling a "story" of the sequence of events. By simulation results, not primarily the raw outcomes of simulation runs are meant. A more important part probably play model-based scenarios. These are narratives written by simulation analysts (i.e. persons who interpret the simulation outcomes, who are in fact often identical with the model developers) on the basis of simulation logs together with appropriately processed representations of numerical outcomes, with the primary goal to condense the

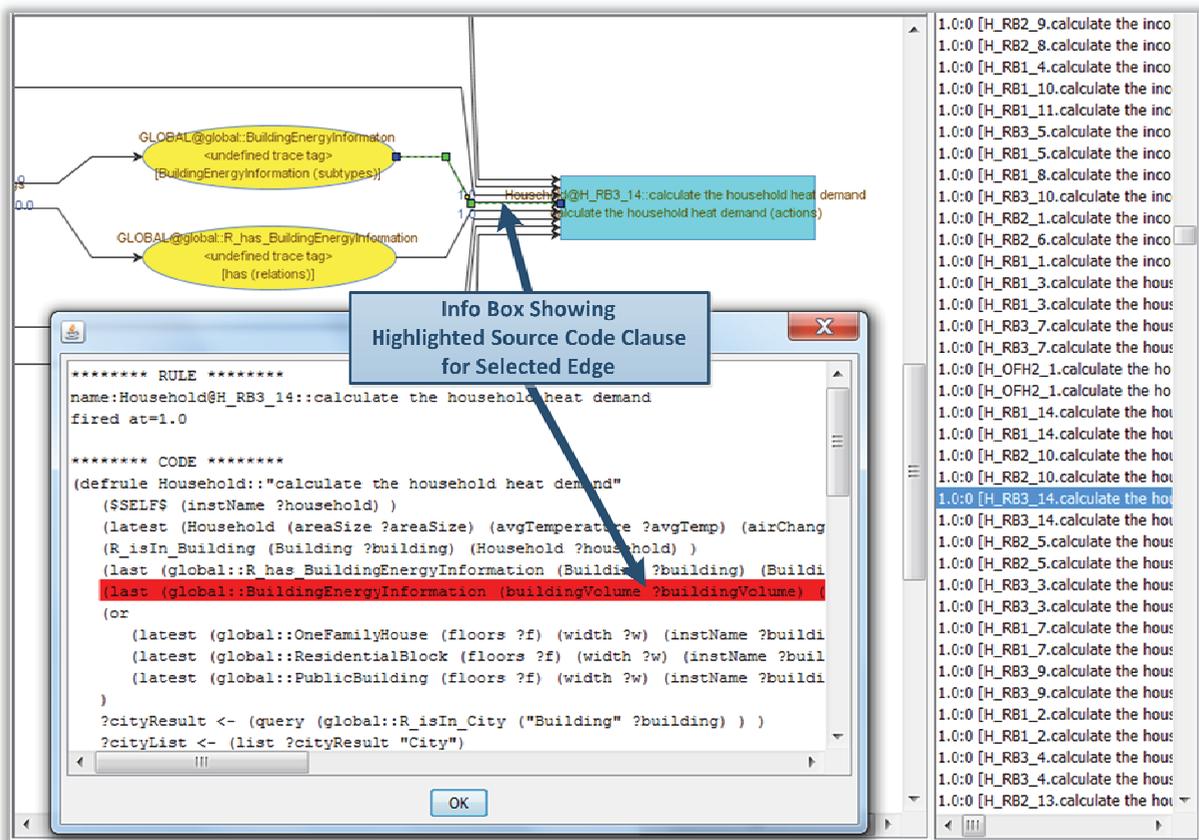


Figure 5 Detail view of the above screenshot with Info Box

sometimes vast amount of generated data into a human readable format. With dedicated tool support (as e.g. another OCOPOMO Toolbox component "Simulation Analysis Tool") it is ensured that the trace information is kept also within model-based scenarios.

As described at the beginning of this paper, one of the basic elements and major added values of the OCOPOMO process is the possibility to support the comparison between simulation results on the one hand, and the evidence provenance base for the simulation model on the other hand. Hence, for both the raw simulation outcomes as well as for model-based scenarios, the trace information can be used to attach simulation results to an enriched CCD.

This enriched CCD can then serve as a basis for a structured approach of comparing initial and stakeholder scenarios with model-based scenarios and other results, and, thus, constitutes a profound information platform for model presentation, discussion and validation. The consolidated CCD that emerges from this examination can in turn be used for developing a more detailed, more precise and with the arguments of stakeholders aligned simulation model. Hence, the modelling cycle proposed by the OCOPOMO process can be closed herewith.

## CONCLUSIONS AND FUTURE WORK

The approach presented in this paper shows a way for maintaining traces and provenance during and beyond simulation runs, as parts of a comprehensive development process for evidence-based policy models. Although the example described in the paper shows a part of a model that deals with engineering aspects rather than with social interaction (which one might expect to be more important in a policy model), the traceability concept and implementation should have become clear. It shows furthermore the capability to deal with a considerable level of details in simulation models, of course also with rules for social behaviour.

Mostly positive and promising experiences have been made during implementing and integrating the toolbox components, and applying both toolbox and process to the development of the three pilot case models in OCOPOMO. Firstly, the traces turned out to be extremely helpful for model developers in order to understand model structure and code details for models written by other modellers. Secondly, it is also likely to gain added value in cases where the model developing process is strictly bottom-up, since it allows the modeller to visually perceive the model structure, and, hence, helps to keep track even on large models. The stakeholder perspective is not covered by the Model Explorer yet, but there is another OCOPOMO toolbox component (Simulation Analysis Tool) dedicated to this topic which relies on the techniques described in this paper.

Concluding it can be said that this contribution entails considerable potential for enhancing manageability and maintainability of complex evidence-based simulation models.

In a short-term perspective, the future work will be guided by questions regarding the method for scenario comparison and more enhanced visualisations for policy makers and stakeholders. A long-term perspective will include an exploration of the prospective opportunities and boundaries

of this approach, especially related to model verification and validation on a more general level.

## ACKNOWLEDGEMENTS

OCOPOMO is co-funded by the European Commission under the 7<sup>th</sup> Framework Programme. DRAMS, CCD and CCD2DRAMS have been developed in OCOPOMO. This publication reflects the view only of the authors and the project consortium, and the Commission cannot be held responsible for any use, which may be made of the information contained therein.

Parts of this paper have previously been published in (Lotzmann and Wimmer 2012).

## REFERENCES

- Alam, S.J.; R. Meyer; G. Ziervogel and S. Moss. 2007. "The Impact of HIV/AIDS in the Context of Socioeconomic Stressors: an Evidence-Driven Approach". *Journal of Artificial Societies and Social Simulation* 10, No. 4, 7.
- Barthelemy, O.; S. Moss; T. Downing and J. Rouchier. 2001. "Policy Modelling with ABSS: The Case of Water Demand Management". CPM Report No. 02-92. Centre for Policy Modelling, Manchester Metropolitan University, Manchester.
- Edmonds, B.; and S. Wallis. 2002. Towards an Ideal Social Simulation Language. Technical report. Manchester Metropolitan University.
- Epstein, J. M.; and R. Axtell. 1996. "Growing Artificial Societies – Social Science from the Bottom Up". MIT Press, Cambridge, MA.
- Munroe, S.; P. Groth; S. Jiang; S. Miles; V. Tan; J. Ibbotson; and L. Moreau. 2006. "Overview of the Provenance Specification Effort". University of Southampton Institutional Research Repository ePrints Soton. URL: <http://eprints.soton.ac.uk/263055/1/OverviewVision.pdf>. Last accessed August 15, 2012.
- North, M.J.; N.T. Collier and J.R. Vos. 2006. "Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit". *ACM Transactions on Modeling and Computer Simulation* 16, No. 1 (Jan.), 1-25.
- Lotzmann, U.; and R. Meyer. 2011. "A Declarative Rule-Based Environment for Agent Modelling Systems". The Seventh Conference of the European Social Simulation Association, ESSA 2011. Montpellier, France.
- Lotzmann, U.; and M. A. Wimmer. 2012. "Provenance and Traceability in Agent Based Policy Simulation". The 26th European Simulation and Modelling Conference, ESM 2012. Essen, Germany.
- Scherer, S.; M.A. Wimmer; and S. Markisic. 2012. "Bridging Narrative Scenario Texts and Formal Policy Modeling through Conceptual Policy Modeling: The Consistent Conceptual Description Tool". Submitted to *AI and Law Journal*.
- Wimmer, M.A.; K. Furdik; M. Bicking; M. Mach; T. Sabol; and P. Butka. 2012. Open Collaboration in Policy Development: Concept and Architecture to integrate scenario development and formal policy modelling. In Y. Charalabidis and S. Koussouris, editors, Empowering Open and Collaborative Governance. Springer Berlin / Heidelberg, pp. 199 - 219