Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

info VII

SSI SCHÄFER

Elements of an Appropriate Documentation Framework for Agent-Based Simulation Models

Cornelia Triebig & Franziska Klügl

2008-03-26 ABModSim – EMCSR – Wien

Outline

- Motivation
- Documentation Approach
- Documentation in the Scope of Reuse
- Conclusion and Future Work

Motivation

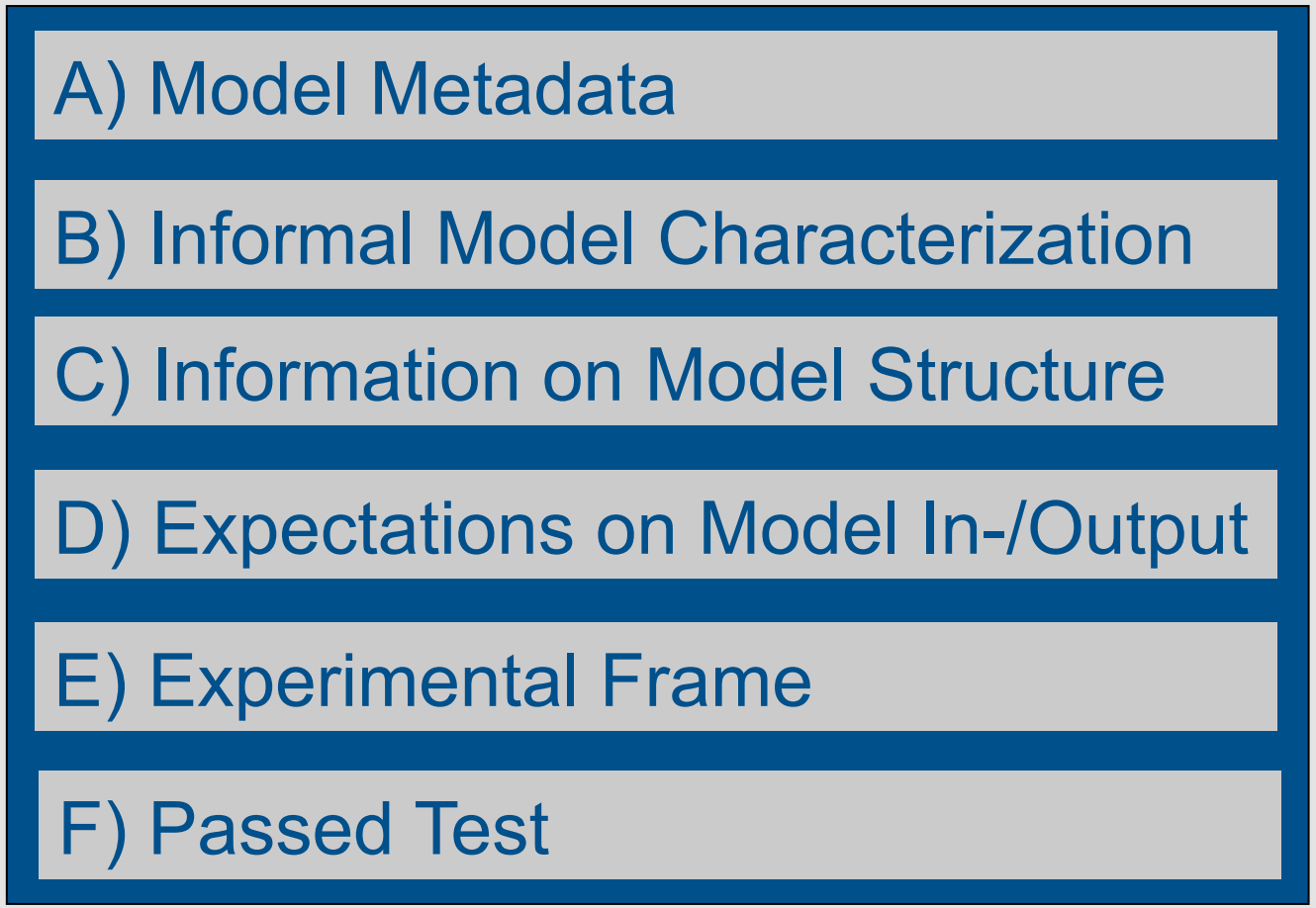
- Good code needs no documentation?
  → **When**? **What**? **How**?

- Documentation of ABSim models for:
  - Improvement of the methodological basis
  - Replication
  - Reliability
  - Maintenance
  - Different users in a project
  - Reuse
  - …

Problems

- Simulation models are not mere software
- Interdisciplinary teams
- Missing established formalism
- High level of detail and complex interrelations
- Characterization of emergent phenomena

6-Block Documentation for ABSim Models



Model Metadata

A) Model Metadata

B) Informal Model Characterization

C) Information on Model Structure

D) Expectations on Model In-/Output

E) Experimental Frame

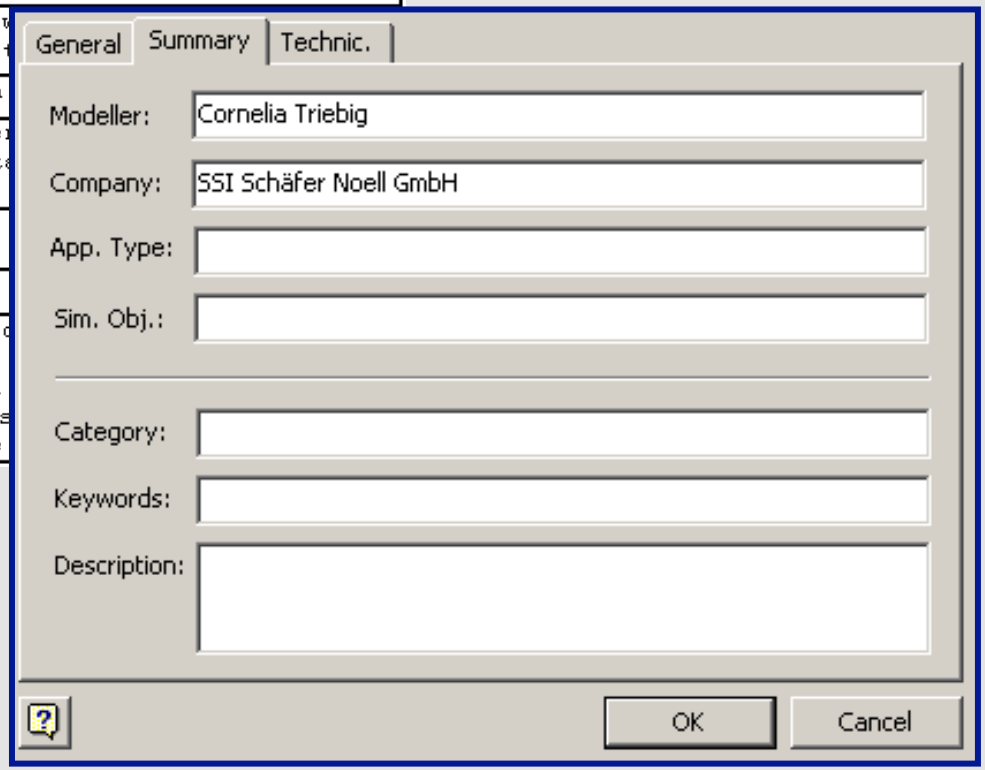
F) Passed Test

Model Metadata

- Model size, Date, Version, Storage Location
- Modeler
- Application Type
- Simulation Objective
- Performance
- Technical & System Features
- Development Environment

Model Metadata

META INFORMATION	
Modeler(s):	Cornelia Triebig
Modeling year:	2005-02-15
Version:	5.1
Status:	Stable
Storage location:	D:\CVS_IT\InternalProjects\SeSAm\Implementation
Simulation objective:	Representing a warehouse v and behavior for testing t
Application type:	Logistics, Automatic High
Performance (Exp. Sim. Cost):	*Initialization:* 20 sec per *Sim. Run* (Stress Test, Ste …
Technical & System Features:	Stepwise simulation, random update of agents
Development Env.:	SeSAm (www.simsesam.de)
INFORMAL MODEL CHARACTERIZATION	The goal of the simulatio free working control commissioning of the real spent time of a control s a building site should be



Information on Model Structure

A) Model Metadata

B) Informal Model Characterization

C) Information on Model Structure

D) Expectations on Model In-/Output

E) Experimental Frame

F) Passed Test

Information on Model Structure

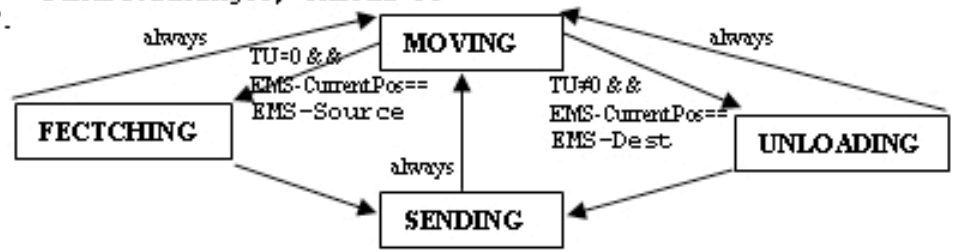
- Information on how the model works
- Assumptions → model structure
- Model constituents:
  - Agents
  - Environment
  - Interaction
- Information on functionality, services, interaction and perception

Information on Model Structure – Agent

EMS-Vehicle

CHARACTERIS. EMS-Vehicle is a mobile agent that moves on a predefined path (circle). An empty vehicle (without Transport Unit (TU)/load) polls on a black board for open Transport Orders (TO). The TO communicates where to fetch a TU and where to unload it afterwards. Possible fetch- and unload positions are Transfer Places or Buffer Places. …

PARAM SendTI, SendTC, AcknowledgementCode
STATE VAR EMS-Source, EMS-Dest, EMS-CurrentPos, ReceivedTO, CurrentCircle, CurrentGoalCircle, CurrentTU
STATES Moving, Unloading, Fetching, Sending
ACTIONS Circle, MoveToSource, FetchTU, SendTI, MoveToDestination, SendTC, SendDeviatingTC, Unload TU
STATE TRANSF.



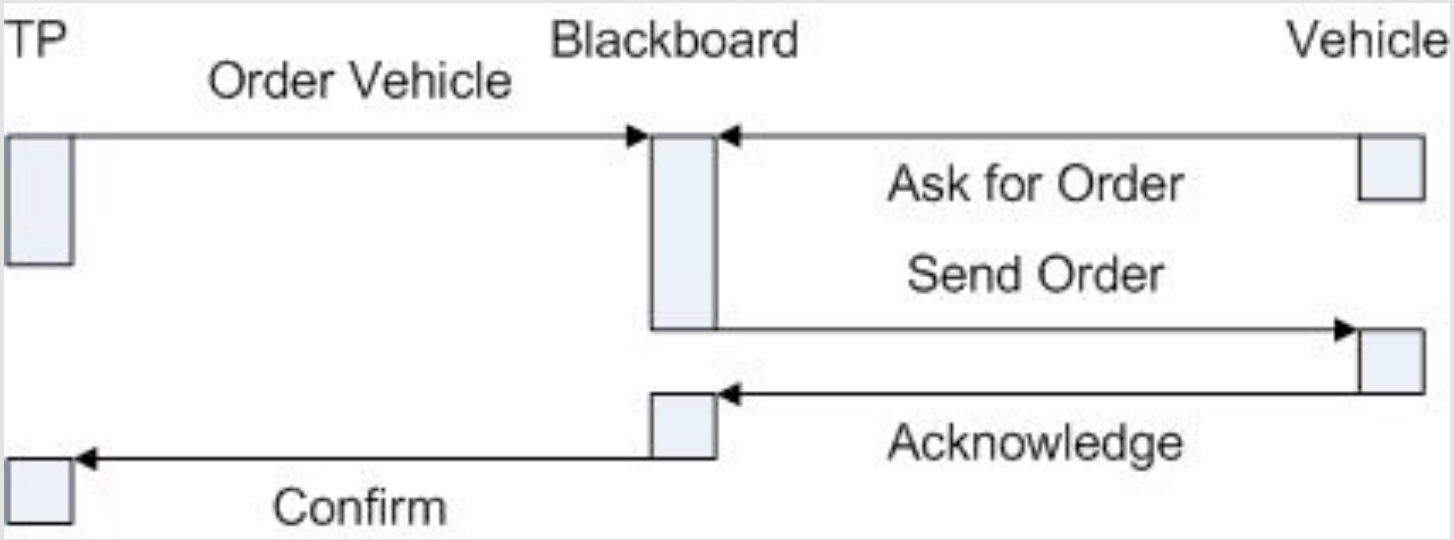
INTER. PARTNERS EMS-TransferPoint, EMS-BufferPlace
PERCEPTION …

BEHAVIOR

MOVING: Circling ! EMS-Source ≠ [0.0,0.0] → MOVING: MoveToSource
…

Information on Model Structure - Interaction

Interaction between Vehicle, TransferPoint and Blackboard



According to AgentUML [Bauer, 2001]

Expectations on Model In-/Output

A) Model Metadata

B) Informal Model Characterization

C) Information on Model Structure

D) Expectations on Model In-/Output

E) Experimental Frame

F) Passed Test

Experimental Frame

A) Model Metadata

B) Informal Model Characterization

C) Information on Model Structure

D) Expectations on Model In-/Output

E) Experimental Frame

F) Passed Test

Experimental Frame

$EF_{model} = \langle I, initConfig, O, tCond, dataColl \rangle$, with

- Input schedules I
- Initial mapping of input values to variables initConfig
- Observational output values O
- Termination condition tCond
- Data collection and aggregation function dataColl

$EF_{agent,}$ & $EF_{environment}$ analogous

(Nach [Klügl, 2007])

Passed Test

A) Model Metadata

B) Informal Model Characterization

C) Information on Model Structure

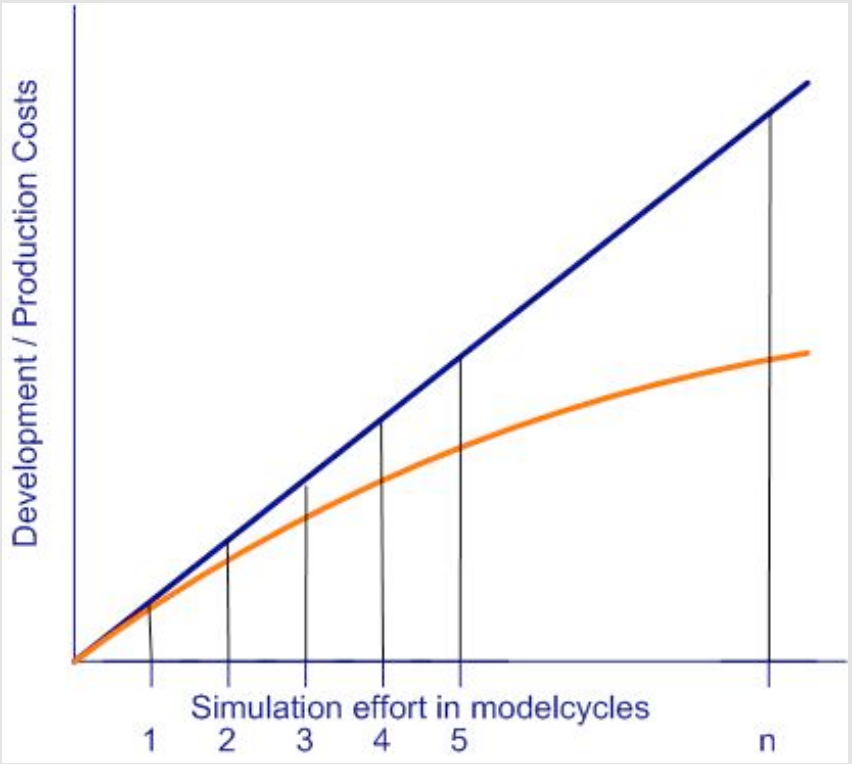
D) Expectations on Model In-/Output

E) Experimental Frame

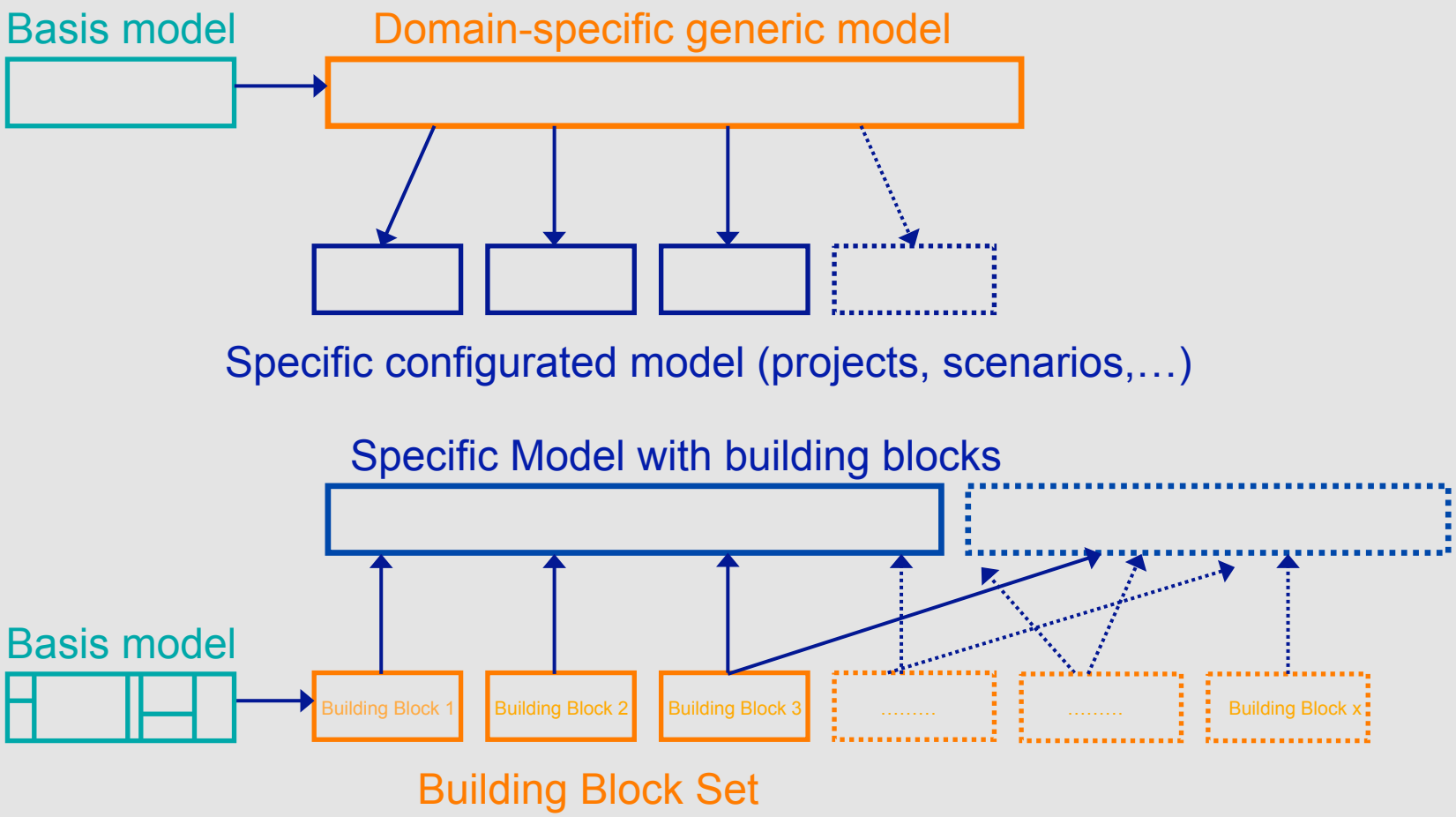
F) Passed Test

Reusability – Protection of Investment

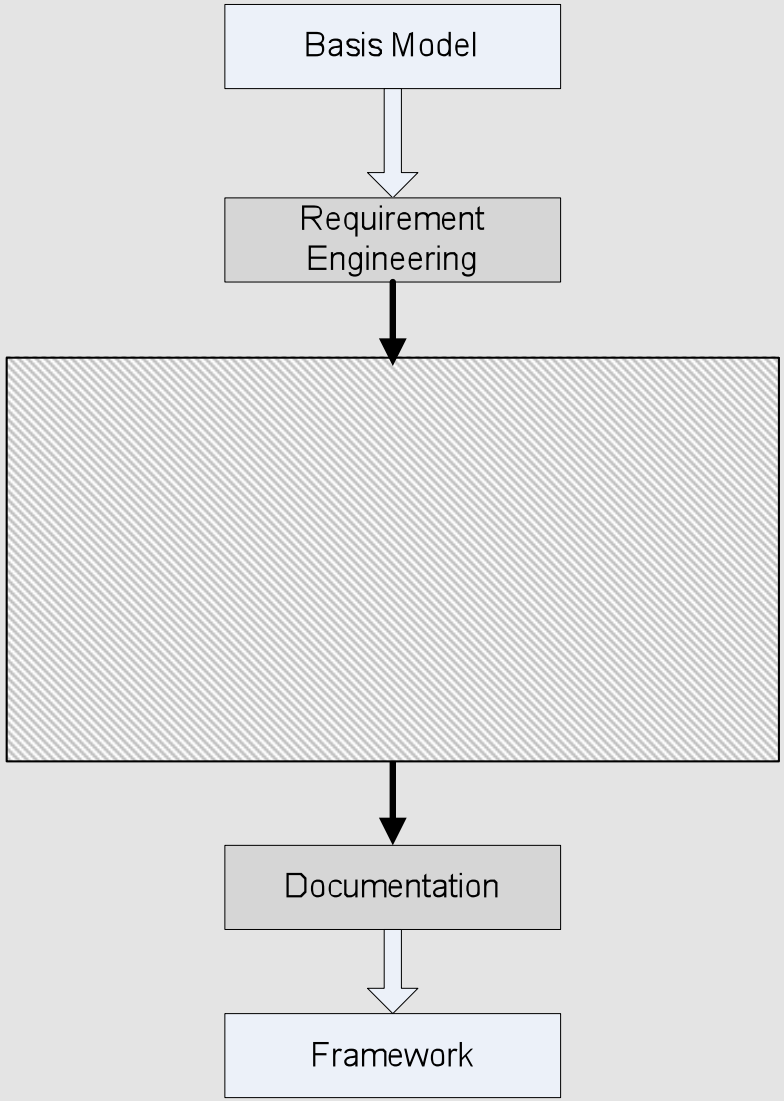
- Design & development of simulations is expensive
- Case studies
- Models grow → pretty elaborated models?
- Save & reduce investment



Generic Model vs. Building Blocks



Transfer-Process



Conclusion & Future Work

- First documentation approach, improvable and maybe not complete
- Documentation and its importance in the scope of reuse

- More application scenarios → better elaboration
- More statistical values and metrics?
- (Partially) automated documentation?
- Integration of the documentation during the development process
- Emphasize particularities of ABSim like emergent phenomena, assumptions, …

Thanks! - Discussion

Refactoring Methods

Extract-Parameter	Extract-Parameter alters the model by replacing the absolute values by variables. The parameter is associated with the agents or other model parts where the quantity is actually used. This procedure lays the basis for model configurability and supports readability.
Extract-Activity / Extract-Method	The overall structure of the activity program can be clarified and its structure improved, when parts of the complex activity are extracted to new activity and an activation of the new one is inserted. This has not only the effect of improved structure, but again lays the foundation for reusability or simplified reproducibility of the extracted part.
Extract-Partial-Graph	corresponds to re-organization of the reasoning engine by inserting additional levels of aggregation – if the used language allows for such hierarchical behavior structures. A partial skeleton plan or a partial activity graph is extracted and placed as a kind of subroutine.
Inline-Activity / Inline-Method	reverse refactoring to extract-activity and should be applied when the overall activity graph contains many, small activities with only simple connections between one-action activities. …
Rename	Rename changes the model element name at every occurrence. This is a standard refactoring method that is provided by nearly every development tool. It improves readability and transparency of the model enormously.
Inline-Assertion	Insert-assumption is a refactoring where assertions, guards, etc. are inserted into the behaviour code, stating what conditions have to be fulfilled at certain steps of the program. Assertions thus improve testability of a model and spare simulation time by cancelling simulations that run out of scope.
Make-Calculation-Explicit	Make-Calculation-Explicit identifies interesting partial values and introduces auxiliary variables for storing these values. A new variable is added used in the formula that hereby becomes accessible for human control. Auxiliary variables also support testing, etc.
Collect-Parameter	For comfortable configuration, parameters should be at hands of the modeller. When one assumes the existence of an explicit, overall model documentation and interface description), we could associate these parameter with this overall facility and thus define the interface of the overall simulation.
Reduce-Parameter-Set	In MASim, parameters tend to be highly dependent on others. Thus, there is a chance, that some of them can be computed based on the others' values. Reduce-Parameter-Set replaces a parameter by its calculation. Introducing a equation instead of a fixed value, relations between parameters are made explicit and noticeable in the model itself.
Replace-by-Value	Instead of using parameters calculations based on constants every time the value is accessed, one may use a refactoring similar constant folding to code optimization: computations are replaced by constant values. The aim of this refactoring is improving simulation speed. However, it is traded against the price of clarity, explicit relations are getting lost.
Delete-Partial-Graph / Activity	This refactoring method deals with dead and redundant code, much like dead-path-elimination in code optimization. This situation can clearly be the result of the iterative modeling process applied without permanent re-design efforts. With Delete-Subgraph redundant sub-graphs are cut off and removed.
Delete-Redundant-Parameters	During iterative model development, parameter may have been defined that turn out to be un-used or without influence in a sensitivity analysis. These redundant parameters are removed from model.
Delete-Redundant-Out/Input-Ports	Output ports define with which actions the agent moves through its environment; some of them may turn out to be unused throughout the simulation runs. Thus, if no other agent addresses these ports, they can be deleted. This may happen because of a changed agent behaviour or environmental program.
Bundle-Messages	Depending on the type of message and the ordering of available information combining small messages to a larger one may reduce at least traffic on the communication channel, but also simplifies communication protocols. **Split-message** forms the reverse refactoring method that may be applied when protocols require sending really large messages.
Fix-Interaction-Partner	In MASim models, interaction partners for certain agents can be determined in several ways. Firstly, the agent evaluates its perceptions and addresses the agent that it has recently perceived again and again as in purely reactive agent architectures. The second option lies in using a mediator. Thirdly, the address of the other interaction partner could be stored. Fix-Interaction-Partner introduces the address of the interaction partner; other refactorings may exchange addressing methods appropriately and can be defined in a similar way.

Informal Model Characterization

A) Model Metadata

B) Informal Model Characterization

C) Information on Model Structure

D) Expectations on Model In-/Output

E) Experimental Frame

F) Passed Test