Complexity Research Initiative for Systemic InstabilitieS

FP7-ICT-2011-7-288501-CRISIS

Deliverable D6.1
Economic Simulator specification

Author(s)	Tamas Mahr, Laszlo Gulyas (AITIA)
Abstract	This document is the first deliverable of WP6, whose title is "A user-friendly economic simulator". The simulator to be developed in this WP will be a tool to be used by policy makers (or their staff) to support intuition building, dialogue and dissemination. The simulator will be designed to ensure the model is transparent and not a "black box", making it easy to run experiments and scenarios, collect data, and analyze and visualize results. This will enable policymakers to run "what if" policy experiments. Like the online game in WP5, this will use the agent-based model developed in WPs 2 and 3 as its underlying engine, based on data from WP1.

Distribution level	Public	**Status**	Final	**Version**	01
Contractual delivery date	31/10/2012	**Actual delivery date**		17/12/2012	

The CRISIS project is funded by the European Commission under the Seventh Framework Programme (ICT)

Economic Simulator Specification

AITIA International, Inc.

Table of Contents

1 Introduction

This document is the first deliverable of WP6 of the CRISIS project, titled “A user-friendly economic simulator”. The simulator to be developed in this WP will be a tool to be used by policy makers (or their staff) to support intuition building, dialogue and dissemination. The simulator will be designed to ensure the model is transparent and not a "black box", making it easy to run experiments and scenarios, collect data, and analyze and visualize results. This will enable policymakers to run "what if" policy experiments. Like the online game in WP5, this will use the agent-based model developed in WPs 2 and 3 as its underlying engine, based on data from WP1.

As described by the DoW, the idea of the simulator is to make it very easy to pose policy questions and understand the answers. The policy maker (or her staff) will be able to experiment with policy prescriptions and see the results of his or her prescription under different scenarios. Provided with a specific scenario, the simulator would produce a range of possible future outcomes over a given time horizon. The outcomes will be a range because there will inevitably be stochastic elements to any scenario requiring multiple runs. In this case the simulator will produce either a set of representative values or a range.

The ability of the simulator to explore large numbers of scenarios and parameter space, and the speed with which it will produce results will naturally depend on computational efficiency and its computing platform. Computational efficiency will be taken into account throughout the design phase, and will be considered in detail later in this document. We also plan to build the tool so that it can both be run usefully from a PC on a policymaker's desk for developing intuition about the system and rapid hypothesis testing, but also can be used on a set of computers (possibly summing up to supercomputer power) for in depth analyses of large scale ensembles of runs. Moreover, due to novel computing technology, these two approaches may be combined by running the user interface of the simulator on an individual’s PC or laptop, but executing large experiments “in the cloud” on a set of computers over the Internet.

The WP is scheduled to perform two tasks and to develop four deliverables. The two tasks are the following (from the DoW):

- **T6.1 Design of the economic simulator**. For the specification and design of the simulator, a collaboration of WP2 and WP3 is required to produce a version of the combined agent-models suitable for a simulator. (This may involve some simplifications that are not present in the full version, and will happen in a series of iterations.) The specification will target the simulator for a single non-technical user, and a special emphasis will be placed on ease of use.
- **T6.2 User interface.** For the decision-support application, the graphical user interface is high priority. To enable non-technical users to set parameters, run simulations and analyze the results, the user interface needs to be easy to use and visually expressive. This task should specify the graphical user interface, and the graphics used in the analysis of results.

The four deliverables of the WP are the following:

- **D6.1:** A specification of the simulator
- **D6.2:** Prototype simulator
- **D6.3:** Production version
- **D6.4:** At least one paper submitted for publication in a top ranking journal

The deliverable is structured as follows. First we analyze the current developments in the CRISIS project and the role and main components of the Integrated Simulation in the next section. This will be followed by the requirement analysis of the two identified *use cases* of the software: *expert analysis* and *scenario analysis*. The discussion of these requirements will conclude in a summary of specifications for the Integrated Simulator.

2 The Integrated Simulator

This section provides an overview of the developments in the CRISIS project, with special emphasis on the major software components under development or to be developed. This is then followed by an identification of the main use cases identified for the Integrated Simulator. Analyzing the various software components to be integrated in the Simulator is important in order to understand the differences, as well as the connections between the various models developed in WP2 and WP3 and between the online game and the Integrated Simulator under development in WP5 and WP6, respectively.

2.1 Integrated simulation model, game, and economic simulator

This subsection discusses the main software components under development in the CRISIS project and their relations. The text in this subsection is identical to the respective subsection in *D5.2 Integration Plan* developed in WP5. The extensive quotation is warranted by the fact that both deliverables deal with designing the two inter-related software and their respective use of the CRISIS software components. This shared text helps making clear the role of the two distinct pieces of software.

The integrated simulation model, as depicted in Figure 1, is composed of a macroeconomic (MABM) and a financial model (FABM). The macroeconomic part describes households, and firms, while the financial part describes banks, central banks, and other financial institutions. These entities interact through different markets (deposit, loan, labor, etc.) that can be implemented by different mechanisms like limit-order-book or over-the-counter market. The macroeconomic model, the financial model and the market implementations are melted into a common CRISIS base layer, the integrated simulation library that can be used to implement various different economic models. This library is extensible, new model elements can be added in the future to support other complex economic simulations.

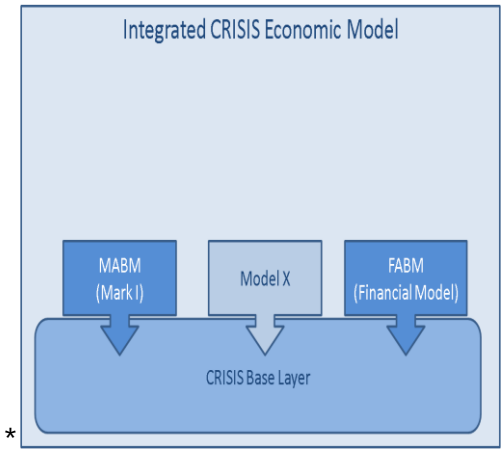


Figure 1. The integrated model

The integrated simulation model is built upon the integrated simulation library. On the integrated model, two other artifacts of the project depend: the game and the economic simulator. How the game is built around the integrated simulation is depicted in Figure 2. In the center of the architecture is a web portal that allows players to find and connect to games, and launches pre-configured simulations on request. The portal is also used by administrators to upload new versions of the game simulation engine, and to create new games by configuring the simulations. When a game is started, the corresponding game simulation is started either at the same server, or on a remote server. The game clients connect directly to the newly started simulation, and interact with it during the game. Consequently, for each game one simulation is started, and the different games can be based on different game simulation engines with different parameter settings.

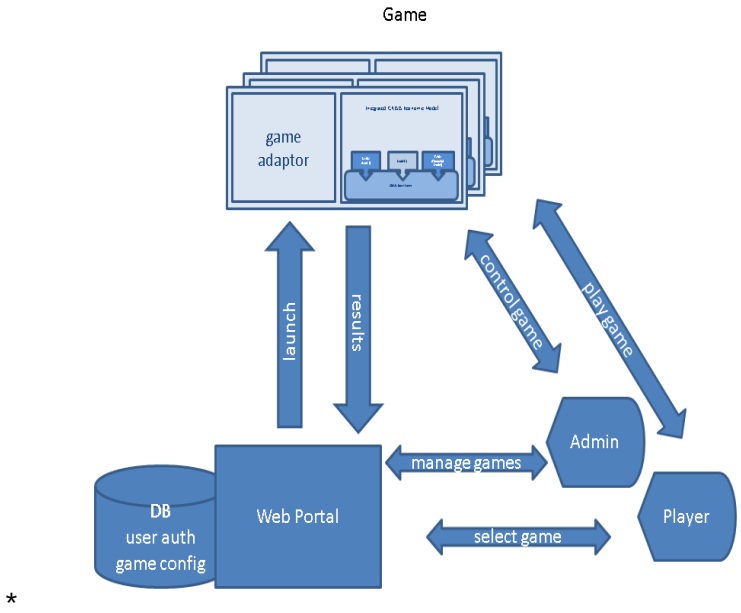


Figure 2. Game overview

In contrast, the economics simulator is used to explore the behavior of simulation models; therefore (see Figure 3) it starts multiple instances of the same simulation model with different parameter settings. From a simulation model built upon the integrated simulation library, the user can generate a WUI that allows him or others to set up a parameter-sweep experiment. The parameter-sweep experiment is executed either at the server hosting the WUI, or remotely on some cloud infrastructure. Depending on the available resources, the simulations of a user (the same model with different parameter settings) can run in parallel on several different machines. The WUI helps the user to follow the completion of the simulations, and to access the results.

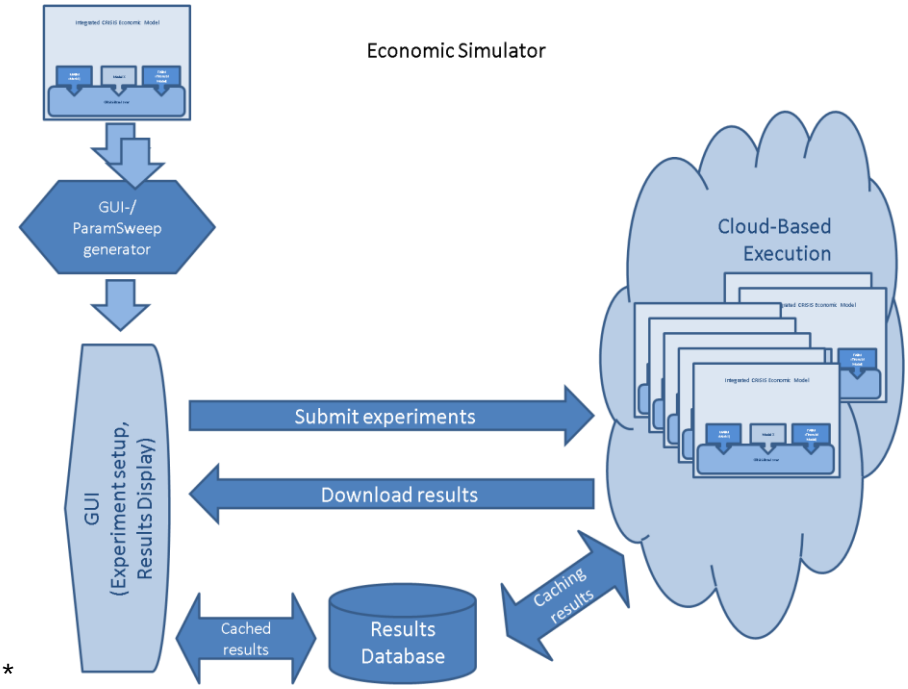


Figure 3. Economic simulator overview

To summarize, both the game and the economic simulator uses the integrated simulation model, or more precisely a simulation model built with the integrated simulation library. The main difference is that while in a sole use-case of the game, a single simulation is started with a given parameter setting and the users/players interact with the simulation while running, in an economic simulator use-case, multiple simulations may be started with different parameter settings (a.k.a parameter sweep) in a non-interactive mode.

2.2 The main Use Cases of the Integrated Simulator

In the following sections we discuss the main Use Cases of the Integrated Economic Simulator. We identify two main approaches to using the system. One is for expert analysis of the entirety of the underlying economic model (the simulation), while the other is a more focused, usually interactive analysis of specific scenarios produced by the system. Therefore, the first use case looks at vast collections of possible behaviors generated by the system (i.e., large subsets of the parameter space specifying the initial and running conditions of the simulated model) and yields large data sets to be analyzed by statistical means. On the other hand, the other use case aims at the understanding the system's behavior under specific, more closely specified conditions. Therefore, the latter provides more details about the unfolding of the events under the investigated settings.

Since the two use cases have different goals, their optimal design is different as well. The expert analysis requires means to configure the *experiment* (i.e., the parameter combinations) that are to be carried out, as well as tools to define the *statistics* to be collected about the

results. This assumes a more complex, expert level setup panel, but does not necessitate interactive visualizations of the resulting dataset. Indeed, the exploration of large (subsets of) parameter spaces usually means large number of simulation runs (often in the order of hundreds, thousands or even tens of thousands of individual runs), which, in itself, renders real-time and online visualization impractical. On the other hand, the scenario analysis use case implies a more user friendly user interface with more emphasis on visualization. This is also made possible, because the number and complexity of settings required for a run (to be visualized) is typically simpler (since only a narrower set of possible parameter configurations is considered to choose from).

The following sections discuss the two Use Cases in more details.

3 Expert analysis

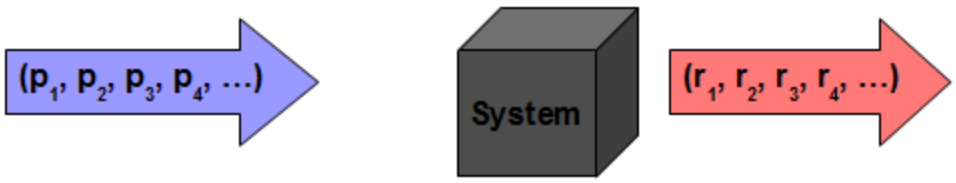
Models of complex social systems typically depend on a number of assumptions, quantified in the form of specific values to certain model parameters. Ideally, any such model should be tested with any meaningful combination of these parameters, in order to determine the validity of the model. Such experiments also allow for the evaluation of possible alternatives for the crucial model components, which, in the case of the CRISIS project, are outputs from WP1 and WP4.

In addition to the dimensionality and the size of the parameter space, the *sensitivity analysis* of complex system models has to face the additional challenge of establishing the results' statistical validity, independent of the probabilistic model elements. Because computer programs, and thus computational models, are inherently deterministic, random factors are modeled by using so-called pseudo random number generators (RNGs). RNGs generate a deterministic sequence of numbers, with the desired statistical properties, depending on an initial value termed *seed*. Different seeds result in different random number sequences. Thus, the task of establishing the results' statistical validity involves running the simulation with various RNG seeds and analyzing the collected results.

In the following, we start with an overview the problem of efficiently executing and analyzing large computational simulations. Then we discuss potential solutions, like the options of distributed execution and non-naïve approaches for sampling the parameter space. This is followed by a detailed discussion of the functions to be supported by the Integrated Simulator addressing these issues. The functionality described below is a limited subset of the capabilities of the Model Exploration Module (MEME, http://meme.aitia.ai/), AITIA International's tool for the computational exploration of agent-based simulations. The implementation of the Integrated Simulation will consist of the customization of the software and its integration in the Simulator. Part of this task will involve making MEME compatible with the MASON simulation framework used in the CRISIS project.

3.1 Computational Experiments At Large Scale

Simulations are computer programs and computational experiments are carried out to investigate their behavior. Generally speaking, computer programs transform their input data into output data (user-interaction can be and often is viewed as real-time input). In this sense, computer programs and thus computer simulations are multi-variate, multi-valued functions. However, the "transformation rules" are typically very complex even for very simple programs. Therefore, it is generally infeasible to derive closed form solutions for the above "function". This is why simulation execution is used to map the "behavioral function" of the simulation.



This is akin to non-computational experiments where experimenters try to determine how the system's *response* (output) depends on controllable *factors* (parameters), while controlling for natural variability (e.g., stochasticity in case of computer simulations) by doing *replicates*. The problem is, however, that the *parameter space* (i.e., the set of relevant inputs) of any minimally interesting simulation tends to be large. (Having 5 relevant parameters, each having 10 discrete values of interest, immediately yields a set of 5^{10} ~10 million potential input combinations.) Considering that the running time of simulations is typically in the order of minutes or hours, it becomes clear that running simulations at this scale and complexity is a very computationally intensive task.

There are two basic approaches to address this problem. One increases the computational capacity available, while the other attempts to lower the size of the sample taken from the parameter space without loosing too much information. (Given the size of the potentially relevant parameter space, in practice all simulations are only studied by *sampling* the parameter space.) In other words, the first approach tries to explore as large an area of the parameter space as possible, but attempts to make this possible by using more than a single computer. On the other hand, the second approach derives carefully designed plans to pick the parameter sample, in order to be able to learn as much about the model's behavior, as possible.

In the following, we discuss these two approaches in more detail, in the context of the CRISIS project. A large part of the work carried out in WP6 will focused on integrating the solutions to these issues into the Integrated Simulator.

3.1.1 Feasibility of Distribution

Even in case of carefully designed experiment plans, the task or executing the simulation with the selected parameter combinations (i.e., initialized with the selected points in the parameter space) may easily exceed the abilities of today's PCs or workstations. Therefore, it is often useful if the execution of computational experiments is distributed among several computers on a network. There are two basic approaches to do this. In the first, each simulation instance is run on a single computer, but the many instances required for parameter space exploration are distributed over the network. In the second, even components of the same simulation run (e.g., the agents) may be divided up among the participating computers. In the former case, neither the size nor the computational requirements of any of the simulation runs to be executed can exceed the capabilities of the participating (single) computers. However, this approach is relatively easy to implement, and can speed-up the execution of the set of experiments (typically involving thousands or tens of thousands single simulation runs).

On the other hand, the latter case allows for the execution of simulation runs that exceed the capacity of any participating computer. However, this comes at a price of considerable implementation efforts on the side of the engineers of the execution platform, and typically also

on the side of simulation developers. It also raises the issue of close synchronization among the participating computers, since discrete time models typically assume a single 'central clock', which is often inconvenient in a networked environment with computers with varying computational abilities. (It can often be the case that the entire 'computational pool' operates at the rate of the weakest participating unit.) Moreover, in case of models with strong interaction/communication among the agents, the second type of distribution is highly impractical due to the increased communications costs.

In case of the CRISIS project, the second approach will not be supported in the Integrated Simulator. That is because developing distributed, parallel code requires considerable efforts. In addition, these efforts are typically required *after* the task (algorithm) to be parallelized is already defined. In our case, this would mean that parallelization can only start after the model of the Simulator has already been specified. Since the modeling is a continuous, ongoing task in CRISIS, aiming for a distributed version would be impractical and would exceed available resources in WP6. It is to be noted that we are aware of ongoing efforts in d-MASON (distributed MASON) project at http://isis.dia.unisa.it/projects/dmason/. However, the works there are, as of now, incomplete. Thus the CRISIS Integrated Simulator cannot be based on results from that project.

On the other hand, the Integrated Simulator will support the first type of distributed execution. Distributed parameter sweeps are possible because individual simulation runs in parameter sweep experiments are *completely independent* (often also termed “embarrassingly parallel” or “perfectly parallel”). That is, the output of a particular run does not depend on the result of any other run. Since the runs to be executed (i.e., the parameter combinations to be explored) can be fixed in advance, this means that there is no need to wait for the completion of any run in order to be able to start another one. Note, however, that in its entirety, the above statement only holds for the naïve approach of setting the parameter combinations. With the introduction of the more “intelligent” parameter exploration methods discussed below, this independence condition might be violated. However, a distributed approach can still be useful there. On one hand, in case of several experimental designs the independence condition is preserved. On the other, even in the case of methods that introduce dependence between runs, it is typically possible to arrange runs in ‘batches’ containing independent runs that can be executed in parallel. This way, dependence is confined to among batches and the entire experiment can be executed as a sequence of batches that each contains numerous distributed simulation runs.

Distributed parameter sweeps are relatively easy to implement, but they are better handled at the simulation platform level than at the level of individual simulations. First, this is the natural level of generality. Second, this task might be too technical for modelers. Therefore, we have adapted this approach from MEME and will base our integrated solution on the general, easy-to-use toolset that it provides. Using this approach the parallelization of the computational experiments will be seamless and transparent to the user.

3.1.2 Specially Assembled Parameter Configurations

The second approach to handle the large computational demand of simulation experiments is the “smart” assembly of the set of parameter combinations, so that it reduces the size of the set without heavily compromising the results and their validity.

There are two simple, traditional approaches to 'batch experiments' as large scale explorations of the behavior of computational simulations are often called. One is the so-called *One-Factor-At-A-Time (OFAT)* method that changes the value of a single parameter, while keeping the others at their default value. This approach is very effective in reducing the number of simulation runs and works well if there is little interaction among the parameters and the parameters have a reasonably meaningful 'null value'. (For example, this is the case if the model is to investigate the effect of changes from the present situation of the system.) However, in case of agent-based simulation models, default (null) values are often very hard to find or argue for, and interactions are more often than not play a very important role. (In these cases, the effect of changing several parameters together affects the output more heavily than changing the parameters individually.) The other traditional approach addresses this problem by the application of 'brute force' – combining and testing all values of all parameters that would be tested in an OFAT experiment, thus creating a *full factorial* design. However, while this solution handles the problem of interactions perfectly, it does so at the maximum price, i.e., measuring the response at all possible points of interest. For the Integrated Simulator, we also need better solutions than these.

In many non-computational disciplines experiments are costly, hard to carry out and/or imply ethical issues (like experimenting with living beings, etc.). In these fields limiting the number of *experimental runs* is imperative. It is for this reason that *experiment design* has a long tradition and a large literature dating back as far as the 1920s. [1] Unfortunately, these methods are often overlooked in the practice of agent-based computational simulations, or more generally, in complex systems studies. Recently, this methodology has found its way to the general computational simulation community, but as of yet, it is not general practice there either. [2][14] MEME was the first tool dedicated to agent-based computational experiments that supported these methods.

Below, we provide a short overview of the *Design of Experiments (DoE)* methodology and concepts and introduce some of the simplest designs, following in the footsteps of [11] and [17].

3.1.3 Design of Experiments

In the context of DoE, the definition of an experiment is the study of a given system by "(…) deliberately changing one or more process variables (or factors) in order to observe the effect the changes have on one or more response variables. The (statistical) design of experiments is an efficient procedure for planning experiments so that the data obtained can be analyzed to yield valid and objective conclusions." [11]

A *design* is a detailed plan in advance of doing the experiment. The goal is to maximize the amount of "information" that can be obtained for a given amount of actual experiments carried out. There are several motivations to design an experiment. According to [11], one can be interested in "choosing between alternatives; selecting the key factors affecting a response; regression modeling; response surface modeling including hitting a target, maximizing (or minimizing) a response, reducing variation, making a process robust and seeking multiply goals." Among these, selecting the key factors and response surface modeling are the most relevant in the context of agent-based computational simulation and the CRISIS project, in general. Regression modeling and reducing variation might also be interesting, albeit most

agent-based models are non-linear and high variation regimes might themselves be an important result of such models.

For the most relevant motivations, the DoE literature offers the following broad categories of designs, with several objectives for each. [11]

3.1.3.1 Screening Designs

Screening designs, as the name suggests, are used for initial screening of the system's behavior. They help identifying which factors/effects are important. If there are only 2-4 potential factors of interest, one may be able to afford a *full factorial* design (essentially, the traditional 'brute force' approach of batch computational experiments). However, when having more than 3 potential factors of interest, one may want to begin with as small a design as possible. Screening designs are also useful when having qualitative factors or when suspecting non-monotonic effect on one on the quantitative factors.

3.1.3.2 Response Surface Modeling

Response surface modeling is fundamentally a way to create a (simplified) closed-form model of the response surface. This can be used to achieve various further objectives, like hitting a target value, maximizing or minimizing a response, reducing variation, etc.

3.1.3.3 Regression Modeling

Regression modeling, as its name suggests, is used to estimate a precise model of the response and quantifying the dependence of response variable(s) on process inputs. This is typically done the assumption of linear response, which is unlikely to be the case with the CRISIS models.

Naturally, the real art of DoE lies in the particular design addressing one of the above objectives. In the following, we will introduce a few of them. Some of these designs will be available for computational experiments in the Integrated Simulator to be developed in WP4.

3.1.4 Special Designs and Techniques in DoE

This section introduces a few simple experiment designs and techniques in order to provide a glimpse on the kind of techniques offered by the DoE literature. Here, again, we are relying on the very useful summary of [11].

3.1.4.1 Full Factorial Designs in Two Levels

Above we have discussed the traditional approaches of computational experiments that often apply 'brute force' to the problem of studying the behavior of the system. In fact, this technique creates a *full factorial* design, in which all possible combinations of all values of all factors are included in the design. A simpler and more common experimental design is the *two level* full factorial design, in which all input factors are allowed two values (are set at two levels) each. For convenience, these levels are called `high' and `low' or `+1' and `-1', respectively.

Naturally, if there are k factors, each at 2 levels, a full factorial design has 2^k runs. When the number of factors is large, a full factorial design requires a large number of runs, even with two

levels, and thus it is not very efficient. In these cases, a fractional factorial design or a Plackett-Burman ([12]) design is a better choice.

3.1.4.2 Fractional Factorial Designs

Fractional designs reduce the size of the design by omitting some of the combinations of a factorial element. Or more precisely, a fractional factorial design is "*A factorial experiment in which only an adequately chosen fraction of the treatment combinations required for the complete factorial experiment is selected to be run.*" [15] In general, about ½, ¼ of the combinations defined by the full factorial is picked. Of course, the art is in selecting the 'adequate' fraction, but luckily, there is a large literature containing good advices. Various strategies exist, which we cannot discuss here. However, it is important to emphasize that properly chosen fractional factorial designs for 2-level experiments have the desirable properties of being both balanced and orthogonal. This means that all combinations have the same number of observations, and that the effects of any factor balance out (sum to zero) across the effects of the other factors (which ensures that the design does not 'waste' experiments in regard to the information gained, even if one factor has no effect at all on the response).

In case of two-level fractional designs, the set of combinations are often extended by a 'center point' (between the low and high values for each factor studied) in order to estimate the curvature of the response function.

3.1.4.3 Box-Wilson Central Composite Designs

The Box-Wilson Central Composite Design, also called 'central composite design', consists of two parts. It contains a factorial or fractional factorial design with center points. In addition, it contains a group of 'star points' that allow the better estimation of curvature. Central points are specified depending on the variety of the central composite design. In the *circumscribed* design variety, if the distance from the center of the design space to a factorial point is normalized to ±1 unit for each factor, then the same distances for the star points are ±α, where $|\alpha| > 1$. The precise value depends on certain desired properties of the design and also on the number of factors. The star points represent new extreme values (low and high) for each factor in the design. On the other hand, if the limits specified for the factors are true limits, the *inscribed* variety of the design creates a factorial or fractional factorial design within those limits (i.e., it is a scaled down circumscribed design with each factor level divided by α). Finally, in the *face centered* variety the star points are at the center of each face of the factorial space, so $\alpha=\pm1$. Importantly, central composite designs always contain twice as many star points as there are factors in the design. [11]

3.1.4.4 Box-Behnken Designs

The designs discussed so far were fundamentally based on the assumption of linear response, with center points testing for curvature and thus checking the validity of this assumption. Since computer simulations often produce non-linear responses, and this is strongly expected to be the case in the CRISIS models, it is very useful to have designs that can handle such situations. One of the simplest of such designs is the Box-Behnken design, which is a quadratic design that does not contain an embedded factorial or fractional factorial design. Here the combinations are at the midpoints of edges of the parameter (process) space and at the center. Geometrically,

this design resembles a sphere within the parameter space such that the surface of the sphere protrudes through each face with the surface of the sphere tangential to the midpoint of each edge of the space. Given their spherical nature, these designs are rotatable (or near rotatable) and require 3 levels of each factor. The designs have limited capability for orthogonal blocking, in comparison to the central composite designs. [11]

3.1.4.5 Latin Hypercube Designs

Latin Hypercube Designs (LHDs) have the specialty that they consist of *n* runs in a setting where each factor has *n* distinct levels (not necessarily the same *values* for each factor). This is a distinctly richer environment than the two levels of factorial designs or their extended variants discussed above.

More formally, a *k*-dimensional Latin Hypercube Design is a set of *n* points $x_i = (x_{i1}, \ldots, x_{ik}) \in \{0, \ldots, n\text{-}1\}^k$ such that for each dimension *j* all x_{ij} are distinct. The best Latin hypercube designs are often based on orthogonal arrays. Usually the factor levels are equally spaced, which is often achieved by ensuring the 'maximin' property, i.e., that maximizes the separation distance (the minimum distance among points of the design) among all LHDs of a given size *n*, according to some distance measure. Maximin LHDs designs are hard to find, however, therefore for larger designs and higher dimensionalities, often approximations are used.

Latin Hypercube Designs are especially useful for computer experiments, because they ensure that few design points are redundant when the effects are sparse and some parameters do not incur changes in the output. It was also observed that "the designs proposed for computer experiments have almost exclusively been Latin Hypercube Designs". [10]

3.1.4.6 Randomized Block Designs

In certain cases, the experimenter has initial knowledge or assumption suggesting the primary factor of interest, yet, she needs to prove it excluding the possible effects of other, 'nuisance' factors. Such situations are more common in non-computational experiments, but they also exist in the context of computer simulations. For example, in real world experiments nuisance factors might be the specific operator who carried out the experiment or applied the prescribed treatment, the time of the day the experiment was run, the room temperature, etc. In short, nuisance factors are those that may affect the measured result, but are not of primary interest. In case of computer simulations, such effects can be, for example, the particular pseudo random number sequences the simulation was run with (controlled by their RNG seeds).

If it is possible to control nuisance factors, the technique known as *blocking* can be used to reduce or eliminate their contribution to experimental error. The basic concept is to create homogeneous blocks in which the nuisance factors are held constant and the factor of interest is allowed to vary. Within blocks, it is possible to assess the effect of different levels of the factor of interest without having to worry about variations due to changes of the block factors, which are accounted for in the analysis.

When nuisance factors abound, Randomized Block Designs may come to help. In this scenario, blocking is used to handle a few of the most important nuisance variables, while *randomization* is used to reduce the contaminating effects of the remaining ones. This technique can also be

viewed as a set of completely randomized experiments, each of them run within its own block (i.e., in one of the blocks in the entire experiment).

3.1.5 Iterative Approach to Designing Experiments

The designs introduced so far were all ‘static’ in that they designed the entire experiment in advance, fixing all the design points and excluding feedback from the already observed responses to the selection of the design points tested in the future. Indeed, this is the classic DoE approach, but even there the importance of an iterative approach is emphasized, calling it a mistake to believe that ‘one big experiment will give the answer.’ [11] The concept of the iterative approach is that *each stage provides insight for the next*.

The heavy computational demand of the planned simulation experiments with the Integrated Simulator creates a special emphasis on the design and sequencing of the individual runs, even when they are executed distributed on multiple computers. The concept is that the sequencing of the runs should allow for branching depending on earlier results and also for revisiting previously explored areas of the parameter space with greater 'resolution'. Therefore, in designing the simulator a special emphasis is put on studying and developing methods and heuristics with the above dynamic and iterative nature.

3.2 Specifications for the Expert Analysis Use Case of the Integrated Economic Simulator

In the previous subsection we briefly overviewed the basic methods and approaches from the Design of Experiments literature. These methods are of immediate value to us in designing the Integrated Simulator for the *expert analysis* use case. In the following, we will summarize the functions that will be included in the Integrated Simulator to support this use case.

3.2.1 General Description

The Integrated Simulator will have a “parameter sweep” mode that supports the exploration of large ranges and subsets of the parameter space of the underlying economic model. The user will be able to design the experiment either by manually setting up the parameter combinations to run the simulation with, or by using one of the DoE wizards. The wizards will support a set of the DoE methods discussed in the previous subsection. The methods planned to be supported are the following:

- Manual definition of the DoE
- Full factorial designs
- Fractional factorial designs
- Box-Wilson central composite designs
- Box-Behnken designs
- Latin hypercube designs

In addition, we may also support randomized block designs as well. However, this will not be our priority. The Integrated Simulator, on the other hand, will explicitly support working with several pseudo random number sequences and controlling for their seed values.

3.2.2 Recorded Values and Statistics

Simulation experiments are useless without the values of certain model variables (responses) are measured and collected. In most existing simulation platforms this measurement and collection must be implemented by the modeler within the code of the model. This is an unfortunate practice for several reasons. First, the modeler uses its time and efforts on coding routine tasks. Second, the model should be changed regularly, following the data requirements of the different experiments. Or conversely, the model's code becomes packed with uninteresting, technical code. Third, data collection can be tricky in case of distributed simulation experiments or in case of adaptive, dynamic designs. Therefore, in contrast to most existing simulation packages, the Integrated Simulator will offer an easy-to-use, very flexible, user friendly graphical wizard for the specification of data collection requirements. Any global variable of the model will be available for selection and the user will be able to specify the frequency or condition when the data is to be recorded.

The Simulator will offer another, menu-based wizard for the point-and-click assembly of descriptive statistics, as well as, tools for construction and transformation of data collections from variables and return values available in the model. The data construction operations and the statistics defined using the wizard will result in Java code that is 'automatically attached' to the original model on-the-fly, using advanced Java techniques.

Expert users will be able to go even one step further and providing their own statistical routines (scripts) by a few lines of Java code, but without the burden of specifying the entire context or a fully fledged Java method. These lines are also being attached to the model on-the-fly.

The statistics and scripts defined in this wizard also appear among the selectable items in the Data Collection wizard discussed above, thus making them as flexibly configurable for recording as any original, 'first class' variable.

3.2.3 Execution of the computational experiment

After the specification of the parameter space to study, the measurements to be performed and the data to be recorded, the Integrated Simulator will execute the specified simulation experiment. Depending on the particular settings, the same experiment can be executed i) on a local computer, or i) on a remote set of computers (in the cloud).

When the computational experiment is finished the results are offered to the user for analysis. Not wanting to compete with professional statistical packages, the most common envisioned way of analysis will be to export the resulting data set from the Integrated Simulator and analyzing it in SPSS, STATA, R or other statistical tools. (This use case is different from scenario analysis, when interactive visualizations will be provided.)

The collection of the experimental results will be handled automatically even in case of distributed experiments (in the cloud).

3.2.4 Iterative and Adaptive Designs

The Integrated Simulator will also support more advanced, iterative and adaptive experiment designs. In particular, we plan to have heuristic optimization methods (e.g., a Genetic Algorithm-based method) for the adaptive searching of response values. [3-9] In addition, we also plan to

develop iterative methods to map the response function in detail along a single dimension of the parameter space.

4 Scenario analysis

This section discusses the requirements for the second main use case that aims at understanding the system's behavior under specific, more closely specified conditions. This use case provides more details about the unfolding of the events under the investigated settings than the one discussed under the expert analysis use case above. This use case is interactive, where the decision maker analyses the particular scenario in detail, with the aim to uncover as much as possible about the particulars of the given setting. The interactive setting puts more emphasis on ease of use, as well as on the easiness of understanding the output and thus on advanced visualization techniques. Also, this approach is less statistical than expert analysis, since the user looks at individual runs, rather than on a set of experiments. On the other hand, interactive scenario analysis is inherently real-time, which puts execution efficiency into focus as well.

This use case is in line with Task 6.2 of WP6 that aims at creating an advanced graphical user interface (GUI) for the decision-support application. As explicitly stated in the DOW, the graphical user interface is of high priority. To enable non-technical users to set parameters, run simulations and analyze the results, the user interface needs to be easy to use and visually expressive.

Unfortunately, easy to use GUI's cannot be designed in the abstract. The usability is guaranteed by engineering the interface to the particular task it needs to perform. It is a kind of ergonomic optimization that, in our case, would require the exact agent-based simulation model that is at the core of the Integrated Simulator. However, this model is, at the time of this writing, under heavy development in WP2, WP3 and WP8. Moreover, the model is expected to undergo a series of serious changes in the coming months as the modeling efforts in those WPs continue and as the agent-based models of finance and macro-economy mature.

The situation is similar with the advanced data visualizations that are required for this use case. The difference is that data visualization methods and tools can be developed in the abstract (as it is exemplified by a number of general purpose data visualization/analysis packages that are available in the market). However, this tools and methods must be applied to the particular data set and application context, in order to make them accessible by decision makers (or their staff), i.e., our users. This customization task requires software development that can only begin when the near-final details about the Integrated Model become available.

Therefore, we cannot provide a full specification of the functions the Integrated Simulator will contain in order to support this use case. Instead, in the following we will summarize the main functionality of this part of the software, as well as we will provide a brief overview of visualization options we intend to support. It is important to emphasize that, in the end, WP6 will deliver an Integrated Economic Simulator that will have two major components or usage modes. One to support expert analysis along the lines discussed in the previous section. On the other hand, we will also implement components to support the scenario analysis mode. This latter will be tailored to the final version of the Integrated Model at the very end of WP6, and it will be customized to make the exploration of the most important and interesting behaviors of the model as easy and accessible as possible.

The scenario analysis mode of the Integrated Simulator will provide:

- **Ergonomic UI components to set and control the most important parameters of the underlying model**

  The model itself is expected to have quite a number of parameters, but we intend to hide this complexity from the end user. Many of the parameters will be estimated from relevant data sets, or set during the customization process of the Simulator. Naturally, the exact subset of parameters to be exposed to the user will be decided after careful consideration. These decisions will be partially based on the analysis of the model's behavior and the strengths of the effects of the various factors/parameters.

- **Named "scenarios" embodying alternative combinations of sets of parameters**

  Some of the parameters that we will not expose to the end user may have more than a single "interesting" setting. We will save these alternatives as different "scenarios" (i.e., a recession scenario, a boom scenario, etc.). These scenarios will be offered to the user in the form of drop-down menus, enabling the decision maker to switch between vastly different parameter settings by a few clicks.

- **Controls to run, pause and continue the unfolding of events in the simulator at different speeds**

- **Dynamic charts to analyze the model's behavior**

  The exact content, style or format of these charts cannot be specified at the time of writing of this document, since they obviously depend on the particular data to be shown to the user. However, it is expected that these charts will be of the classic set of data visualization charts (i.e., scatter plots, time series plots, histograms, bar plots, pie charts, etc.). In addition, it is expected that networks will also be visualized using some of the standard layout algorithms from complex network analysis.

  The charts will also be interactive: i.e., zooming and rescaling will be done on the fly. Coloring and other details of the outlook will also be configurable and customizable, although we expect limited use of such technical usage.

  We were also considering making the selection of the particular charting format interactive. This may, however, require too much technical skill and a higher level of commitment from the end user, so its implementation is uncertain. Therefore, we delegated the decision about the set of charts and about their types to the customization phase and to the participants of customization (during the latest development phase when the Simulator is tailored to the final version of the Integrated Model).

- **UI Components to inspect individual agents**

  Normally, agent-based simulations are observed and studied at the aggregate, "global" level. The behavior of individual agents is much less important. Therefore, the bulk of the visualization / analysis effort of the scenario analysis module will be focused on aggregated data, using charting tools as discussed above.

  However, we will also provide means to mark certain types of individual agents and follow their behavior. This option could be useful for understanding the way the global events unfold at the micro level. In addition, this may also be important for understanding the rationale for the behavior of the individual.

5 Summary

In this document, we have analyzed the requirements for the Integrated Economic Simulator. We started by discussing the differences and connections among the various software components being developed in the CRISIS project. Especially, we have discussed the Integrated Model, the Online Game and their respective differences from the Integrated Simulator.

After this introduction, we have identified two main use cases for the Simulator. One of the use cases is for the expert analysis of the entirety of the underlying economic model (the simulation), while the other is a more focused, usually interactive analysis of specific scenarios produced by the system. Therefore, the first use case looks at vast collections of possible behaviors generated by the system (i.e., large subsets of the parameter space specifying the initial and running conditions of the simulated model) and yields large data sets to be analyzed by statistical means. On the other hand, the other use case aims at the understanding the system's behavior under specific, more closely specified conditions. Therefore, the latter provides more details about the unfolding of the events under the investigated settings.

Since the two use cases have different goals, their optimal design is different as well. We dedicated this document to analyzing these different requirements. In the case of the expert analysis, we could cover most of the functionality that we intend to build into the software. We overviewed the general problem of executing large scale computational experiments, and discussed the available solutions: distributed execution and the use of advanced statistical methods to methodologically sample the parameter space. We have also reviewed Design of Experiments (DoE) methods that are immediately available for implementation and that could improve the analyzing capability of the expert user. At the end of this section we provided a selection of these methods to be built into the Integrated Simulator.

In the case of the interactive scenario analysis use case, however, we could not go into such details. The reason for this was that user friendly GUI's and easy to use, advanced data visualizations need to be optimized to the particular task at hand. This was not possible at the time of writing of this deliverable, since the agent-based model that will be at the core of the Simulator is still under development and is subject to a series of major changes in the future. Therefore, after briefly reviewing the options available for implementation to support this use case, we have postponed the final decisions about the details of the interactive mode of the Integrated Simulation. However, we have provided a list of important functions and components that will be incorporated in the scenario analysis mode of the Simulator.

5.1 References

1. Box,G. E, Hunter,W.G., Hunter, J.S., Hunter,W.G., "Statistics for Experimenters: Design, Innovation, and Discovery", 2nd Edition, Wiley, 2005
2. J.R. Koehler and A.B. Owen (1996), Computer Experiments, in: S. Ghosh and C.R. Rao (eds), Handbook of Statistics, Vol. 13, Elsevier Science, pp. 261-308.

3. Lambrecht, M. R., Ivens, P. L., Vandaele, N. J., and Miller, J. H. 1998. Active Nonlinear Tests (Ants) of Complex Simulation Models. *Manage. Sci.* 44, 6 (Jun. 1998), 820-830.
4. László Gulyás: „Agent-Based Modeling and Simulation with MASS", *Summer School on Computational Social Sciences*, *National Chengchi University*, Taipei, Taiwan, August 2008.
5. László Gulyás: „Agent-Based Modeling and Simulation with the Multi-Agent Simulation Suite", *Tutorial at the 5th Conference of the European Social Simulation Association*, Brescia, Italy, September 1, 2008.
6. László Gulyás: „Agent-Based Modeling and Simulation with the Multi-Agent Simulation Suite", *Tutorial scheduled for the 22nd European Simulation and Modeling Conference*, Le Havre, France, October 29, 2008.
7. László Gulyás: „Social Simulation with Agent-Based Modeling (with MASS)", *Complex Systems and Social Simulation, Summer University*, Central-European University, Budapest, July 2008.
8. Márton Iványi, László Gulyás, Rajmund Bocsi, Gábor Szemes, Róbert Mészáros: "The Model Exploration Module", *Agent 2007: Complex Interaction and Social Emergence Conference*, Evanston, IL, November 15-18, 2007
9. Márton Iványi, Rajmund Bocsi, László Gulyás, Vilmos Kozma, Richárd Legéndi: "The Multi-Agent Simulation Suite", *AAAI Fall Symposium Series*, Washington DC, USA, November 8-11, 2007
10. Neil A. Butler: „Optimal and Orthogonal Latin Hypercube Designs for Computer Experiments", *Biometrika*, Vol. 88, No. 3 (Sep., 2001), pp. 847-857
11. *NIST/SEMATECH e-Handbook of Statistical Methods*, http://www.itl.nist.gov/div898/handbook/pri.htm, 2008
12. R. L. Plackett and J. P. Burman: „The Design of Optimum Multifactorial Experiments", *Biometrika*, Vol. 33, No. 4 (Jun., 1946), pp. 305-325
13. Space-filling Designs, Tilburg University, http://www.spacefillingdesigns.nl/,
14. Th.J. Santner, B.J. Williams, and W.I. Notz (2003), The Design and Analysis of Computer Experiments, Springer Series in Statistics, Springer-Verlag, New York.
15. The ASQC Glossary & Tables for Statistical Quality Control, 1983
16. Ziegler, B. P.: *Theory of Modelling and Simulation*, John Wiley and Sons, New York, London, Sydney, Toronto, p. 435. 1976.
17. Deliverable D4.1 of the EMIL (Emergence in the Loop) Project: Simulations Execution, FP6 IST #033841, 2008.