SPXI - 11th conference on Stochastic Programming
Universitätscampus Wien, Wien, Austria
Thursday 30th August 2007

SPInE: Stochastic Programming Integrated Environment

a combined paradigm of SP and simulation

CONTRIBUTORS

Coordinator: Gautam Mitra

Modelling: Enza Messina, Patrick Valente, Robert Fourer, Nico Di Domenica, Cormac Lucas, Diana Roman

Solver Algorithms: Frank Ellison, Chandra Poojari, Suvrajeet Sen, Csaba Fabian

PRESENTATION BY

Christian Valente

CARISMA

OptiRisk SYSTEMS

Brunel UNIVERSITY WEST LONDON

INTRODUCTION
SP MODELLING
Modelling Languages
SAMPL
SG in SPInE
Scenario Generation
Computational Architecture
INVESTIGATION
Simulation and testing
Case studies
SOLVING
Solvers considerations
CONCLUSIONS

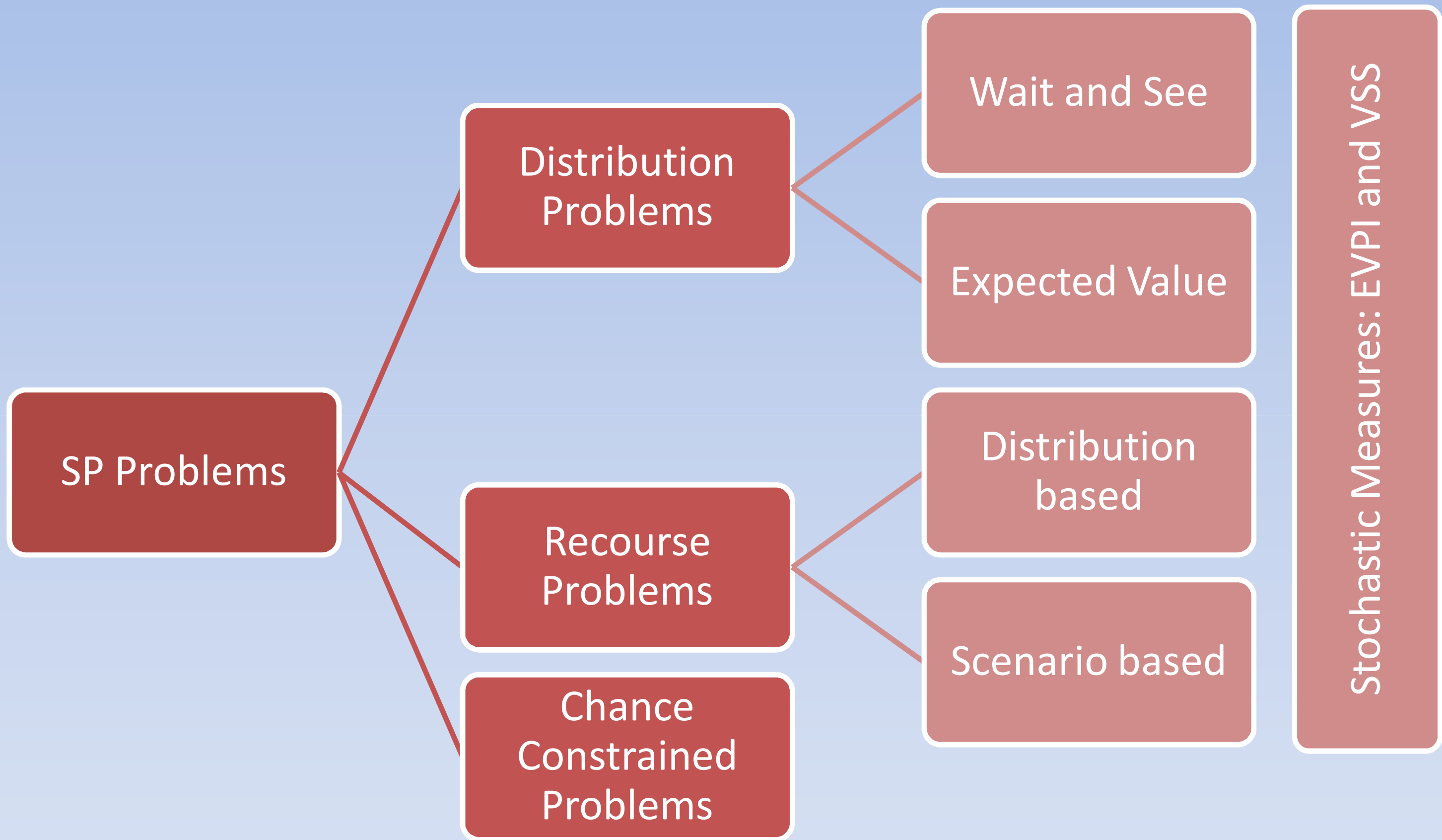
Agenda

- **Introduction**
  - SP taxonomy and notation
- **SPInE as ex ante decision tool**
  - Modelling languages
  - Extensions to AMPL for SP: SAMPL
- **Scenario Generation**
  - Computational architecture
  - Link of scenario generators to SP decision models
- **SPInE as an investigation framework**
  - Simulation, back and stress-testing framework design
- **Solving subsystem**
- **Conclusions**

From a modelling tool to SP framework

- Modelling languages facilitate the expression of an SP model and its solution but..
- SP models are not just the decision models, and the steps to obtain a reliable result are more than just the modelling of the decision process
  - Decision modelling
  - Randomness modelling (Scenario Generation)
  - Problem investigation (back testing/stress testing)
- We propose a framework that assists in all those steps of SP modelling

Taxonomy of SP models



Taxonomy of SP models

- We first consider the linear programming problem:

$$\begin{aligned} Z &= \min cx \\ \textit{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

$$\textit{where} \quad A \in R^{m \times n}; c, x \in R^n; b \in R^m$$

- Let (Ω, F, P) denote a (discrete) probability space where ω∈ Ω denote the events. Let us denote the realizations of *A, b, c* for a given ω as:

$$\left(A^\omega, b^\omega, c^\omega\right) = \xi^\omega \quad or \quad \xi(\omega)$$

OptiRisk SYSTEMS

CARISMA

Distribution problems (EV)

- The distribution of the objective function value for different realisations of the random parameters and also for the expected value of such parameters are broadly known as the distribution problem
- **The Expected Value Problem**
  - The Expected Value (EV) model is constructed by replacing the random parameters by their expected values
  - Such EV model is thus a linear program, as the uncertainty is dealt before it is introduced

Brunel UNIVERSITY WEST LONDON

Distribution problems (EV)

- We can reconsider the previous problem as EV:

$$(\bar{A}, \bar{b}, \bar{c}) = \bar{\xi} = E[\xi^{\omega}] = \sum_{\omega \in \Omega} p^{\omega} \xi^{\omega}$$

$$Z_{EV} = \min \bar{c}x$$

$$\textit{subject to}$$

$$\bar{A}x = \bar{b}$$

where $p^{\omega} = P(\xi(\omega))$ denotes the probability associated with the realisation $\xi(\omega)$.

- Let x^*_{EV} denote the optimal solution to the above problem

Distribution problems (EV)

- This solution can be evaluated for all possible realisations $\omega \in \Omega$
- We can thus determine the corresponding objective function values and compute what is called the **expectation of the expected value** solution

$$Z_{EEV} = E\left[c^{\omega} x^{*}_{EV}\right]$$

- If an ω exists such that x^{*}_{EV} is not feasible for some realisations of the random parameters, we set:

$$Z_{EEV} \rightarrow +\infty$$

Distribution problems (WS)

- **Wait and See Problems**
- Wait and See (WS) problems assume that the decision-maker is somehow able to wait until the uncertainty is resolved before implementing the optimal decisions
- The corresponding problem (and solutions) is stated as: $Z^{\omega} = \min c^{\omega} x$

$$A^{\omega} x = b^{\omega}$$

$$Z_{ws} = E\left[Z^{\omega}\right] = \sum_{\omega \in \Omega} Z^{\omega} p^{\omega}$$

Recourse Problems (HN)

- **Recourse Problems**
- The term Here and Now (HN) is often used to refer to stochastic programming problems, reflecting the fact that decisions are taken before perfect information on the states of nature is revealed
- The classical stochastic linear program with recourse separates the model' s decision variables into
  - **first stage** strategic decisions which are taken facing future uncertainties
  - **second stage** recourse (corrective) actions, taken once the uncertainty is revealed

Recourse Problems (HN)

- The formulation of the classical two-stage SP model with recourse is as follows:

$$\begin{aligned} Z_{HN} = \min \quad & cx + E_{\omega}[Q(x,\omega)] \\ \textit{subject to} \quad & Ax = b \\ & x \geq 0, \end{aligned}$$

where:

$$\begin{aligned} Q(x,\omega) = \quad & \min \ f(\omega)y(\omega) \\ \textit{subject to} \quad & D(\omega)y(\omega) = d(\omega) + B(\omega)x \\ & y(\omega) \geq 0. \\ & \omega \in \Omega \end{aligned}$$

Scenario Based RP

- Giving that the random parameter vector ξ^ω has discrete distribution
- The event parameter ω takes the range of values $\omega = 1,\ldots,\#(\Omega)$
- There are associated probabilities p^ω and random parameter vector realisations ξ^ω such that:

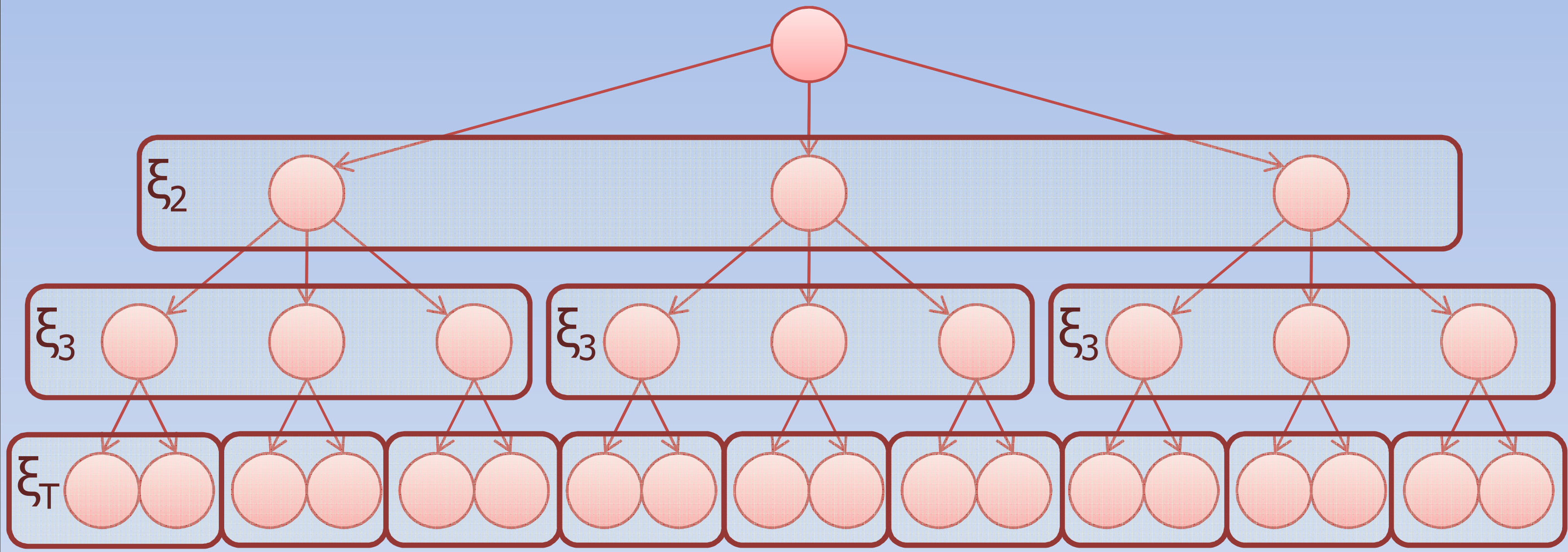
$$\sum_{\omega\in\Omega} p^\omega = 1 \qquad \text{and} \qquad \Xi = \bigcup_{\omega\in\Omega} \xi^\omega$$

Scenario Based RP

- Ξ is the set of all random parameter vectors realisations and is often called the *set of scenarios*
- The uncertainty defines a structure in the form of an event tree, which represents the possible sequence of realisations (scenarios) over the time horizon, with their probability
- When the event tree is explicitly given, we refer to the model as a *scenario based recourse problem*

Scenario Based RP

- The event tree of a T-stages recourse problem:



- All the values of the realizations of random parameters (scenarios) ARE to be defined

OptiRisk SYSTEMS

SP Modelling

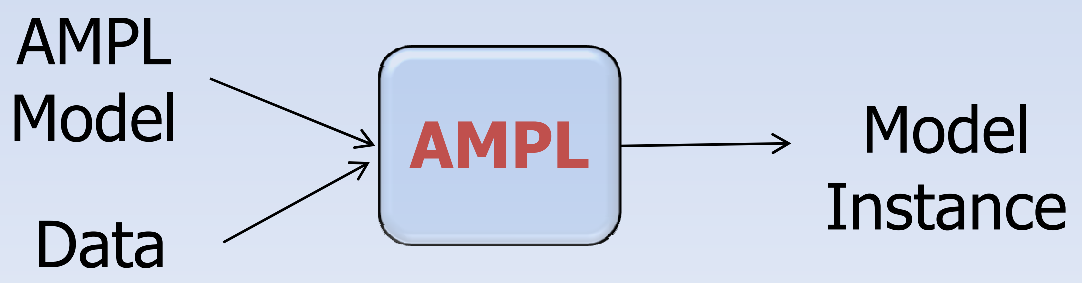
- AMLs allow to conveniently express MPs in a format both easy to understand and that can be processed by a solver
- SPs have different requirements, both in language constructs and in coupling with data
- The design of SAMPL extends an AML (in our case AMPL) to provide these additional constructs
- SPInE is the framework that deals with the second requirement (interpreting SAMPL and coupling to SGs)

CARISMA

Brunel UNIVERSITY WEST LONDON

AML concepts

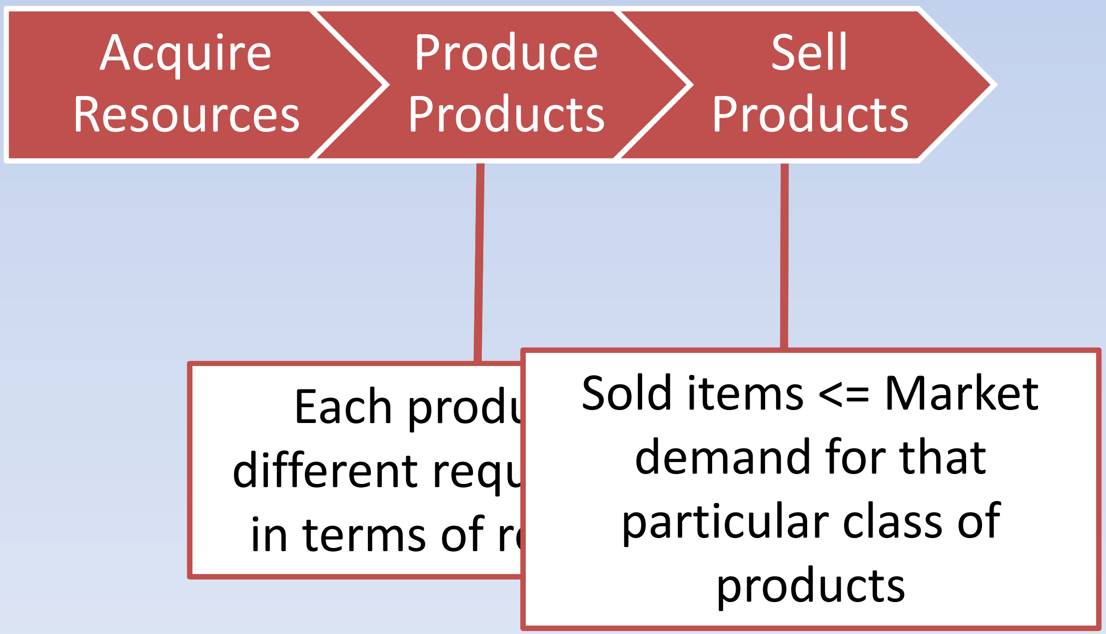
- Mathematical Programming:
  - Modeler’s understanding of the problem leads to “modeler’s form”
  - Solvers accept a different format: “algorithmic form”
  - -> Algebraic Modelling Languages, AMPL being one of those
- It allows a separation between model and data, recommended for most applications



INTRODUCTION
SP MODELLING
Modelling Languages
SAMPL
SG in SPInE
Scenario Generation
Computational Architecture
INVESTIGATION
Simulation and testing
Case studies
SOLVING
Solvers considerations
CONCLUSIONS

Introductory Example

- Deterministic, all parameters are known
- Introductory model: Dakota [J. Higle, S.W. Wallace, *Interfaces, 2003*]



OptiRisk SYSTEMS

Introductory Example

		Description
SETS	P	Products
	R	Resources
VARIABLES	X_r	Resource Acquisition (r)
	Y_p	Production Levels (p)
	Z_p	Sales Quantities (p)
PARAMETERS	C_r	Cost (r)
	R_{rp}	Requirement (r, p)
	S_p	Selling Price (p)
	D_p	Market Demand (p)

OBJECTIVE FUNCTION

$$\max \sum_{p \in P} S_p Z_p - \sum_{r \in R} C_r X_r$$

subject to :

$$X_r \geq \sum_{p \in P} R_{rp} Y_p \quad \forall r \in R$$

$$Z_p \leq Y_p \quad \forall p \in P$$

$$Z_p \leq D_p \quad \forall p \in P$$

CONSTRAINTS

- Five types of entities: SETS, PARAMETERS, VARIABLES OBJECTIVE(s), CONSTRAINTS,
  -> AMLs are to be able to handle them

CARISMA

Brunel UNIVERSITY WEST LONDON

AMPL Considerations

- Distinction between declarations and definitions
- Declaration of the model:

		Description
SETS	P	Products
	R	Resources
VARIABLES	X_r	Resource Acquisition (r)
	Y_p	Production Levels (p)
	Z_p	Sales Quantities (p)
PARAMETERS	C_r	Cost (r)
	R_{rp}	Requirement (r, p)
	S_p	Selling Price (p)
	D_p	Market Demand (p)

```
set products;
set resources;
var amountbuy{r in resources} >=0;
var amountprod{p in products} >=0;
var amountsell{p in products} >=0;
param cost{resources};
param prodreq{resources,products};
param price{products};
param demand{products};
```

AMPL Considerations

OBJECTIVE FUNCTION

$$\max \sum_{p \in P} S_p Z_p - \sum_{r \in R} C_r X_r$$

```
maximize wealth : sum{p in products} price[p]*amountsell[p]
- sum{r in resources}cost[r]*amountbuy[r];
```

CONSTRAINTS

subject to :

$$X_r \geq \sum_{p \in P} R_{rp} Y_p \quad \forall r \in R$$

$$Z_p \leq Y_p \quad \forall p \in P$$

$$Z_p \leq D_p \quad \forall p \in P$$

```
subject to
balance{r in resources}: amountbuy[r] >= sum{p in products}
prodreq[r,p] * amountprod[p];

production{p in products} : amountsell[p] <= amountprod[p];

sales{p in products}: amountsell[p] <= demand[p];
```

Extending AML for SP

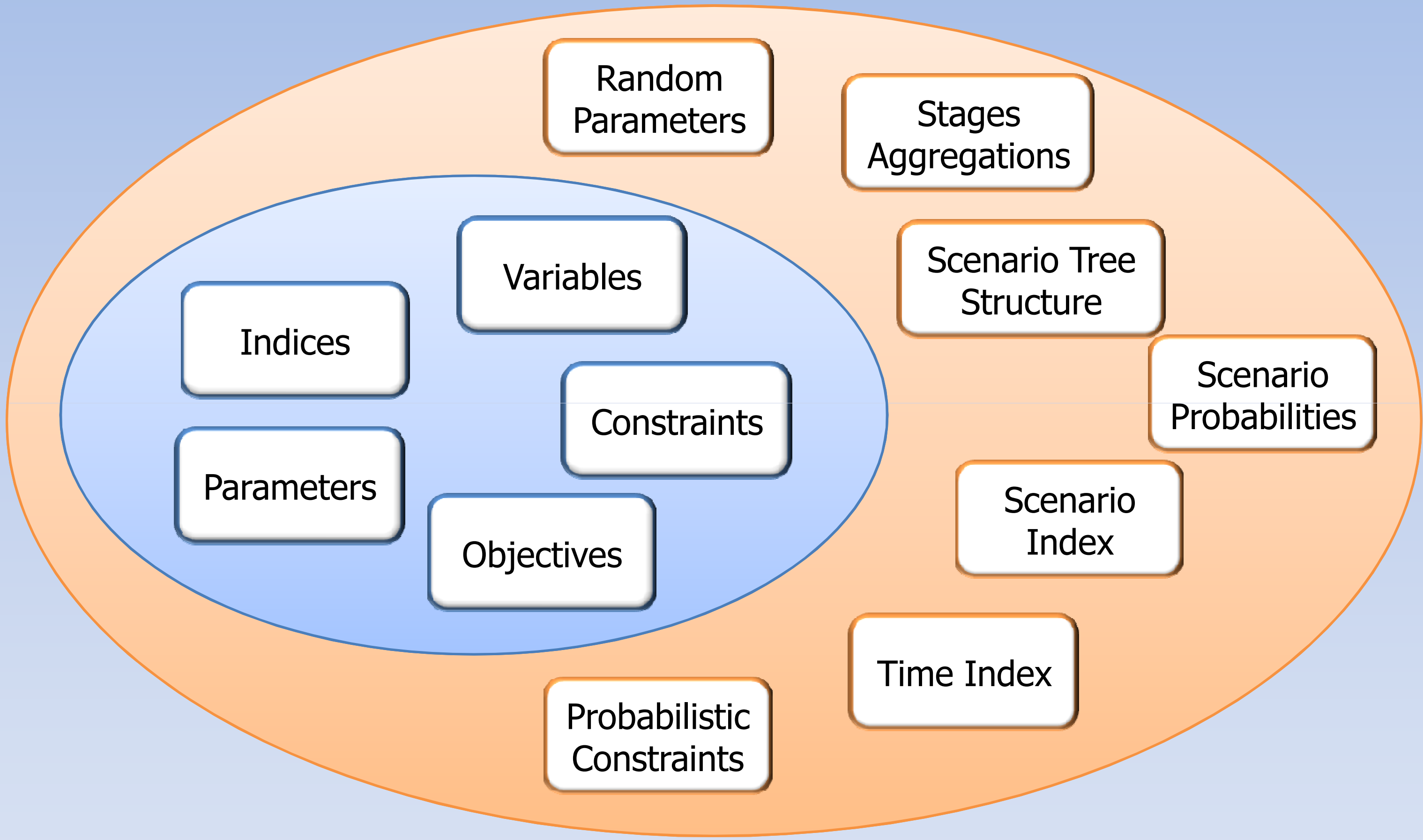
- Stochastic Programming model considered as a MP model extended and refined by the introduction of uncertainty
- The probability distributions of model's random parameters are provided by models of randomness called Scenario Generators (SG)
- The SG to be used is specific to the particular optimisation problem under investigation
- SG provide the stochastic information in form of a Scenario Tree

Language requirements

- Scenario based recourse problems
  - Declaration of **random parameters**
  - Realizations given in the form of a **scenario tree**
  - **Stages** identify the sequence of the decisions
- Chance Constraint problems
  - Declaration of **individual chance constraint** with related reliability level
  - Declaration of **joint chance constraint** with relating reliability level
  - Declaration of **random parameters** in terms of scenarios or distributions

OptiRisk SYSTEMS

SP Modelling Constructs



CARISMA

Brunel UNIVERSITY WEST LONDON

SP Introductory example

- Same model (Dakota) with uncertain demand
- Introduction of scenarios for future values of the demand

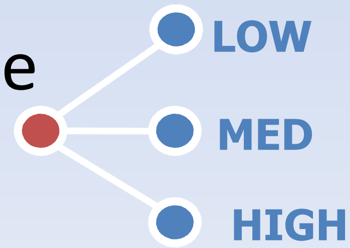
SCENARIO INDEX

RANDOM PARAMETER

Scenario	LOW	MEDIUM	HIGH
Demand(Desk)	50	150	250
Demand(Table)	20	110	250
Demand(Chair)	200	225	500
Probability	0.3	0.4	0.3

PROBABILITY PARAMETER

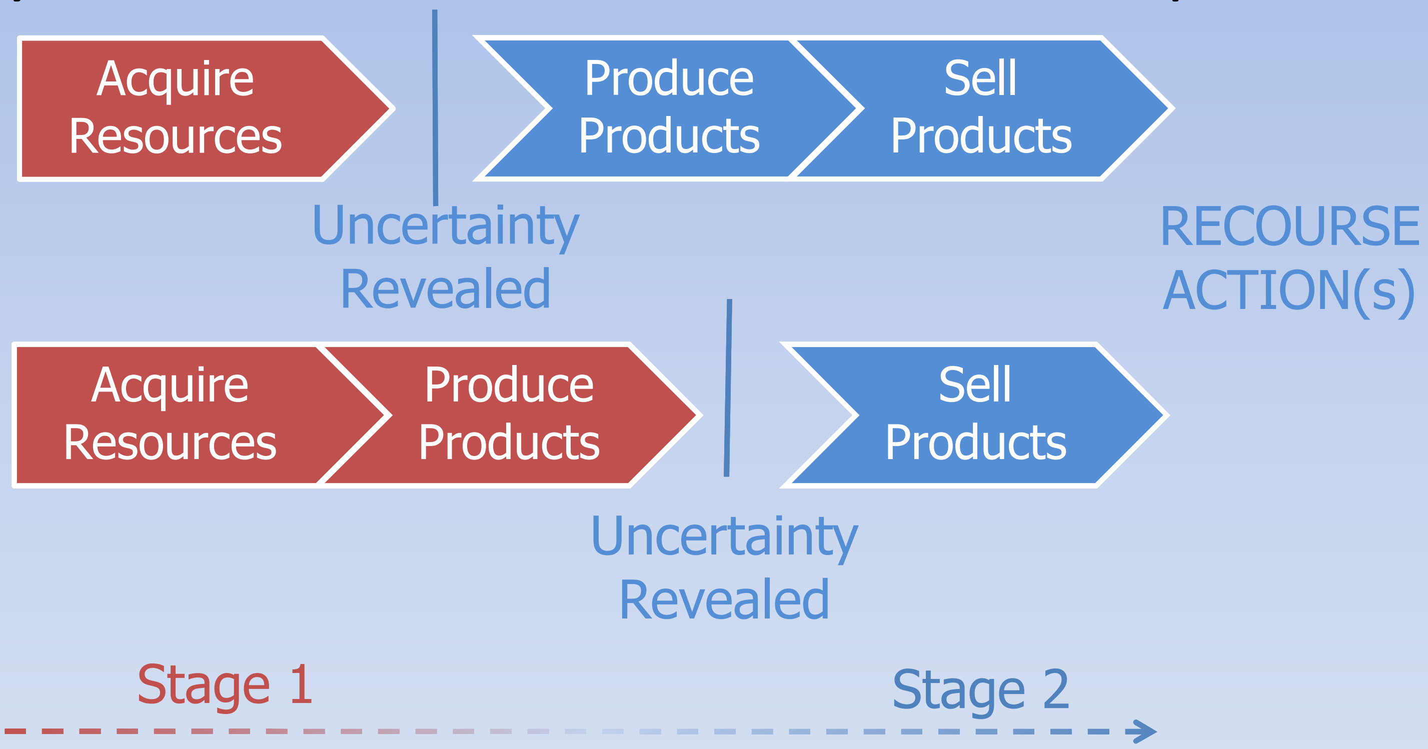
- The event tree is two stage



INTRODUCTION
SP MODELLING
Modelling Languages
SAMPL
SG in SPInE
Scenario Generation
Computational Architecture
INVESTIGATION
Simulation and testing
Case studies
SOLVING
Solvers considerations
CONCLUSIONS

SP Introductory example

- Dynamic structure needs to be further specified



-> Specification of tree structure AND grouping of decision variables into stages

SAMPL Dakota Model

- The same as deterministic Dakota, adding:

```
scenarioset scen:= low med high;

tree theTree:= twostage{3};

random param demand{products, scen};

probability param Prob{scen};

var amountbuy{r in resources,s in scen} >=0, suffix stage 1;
var amountprod{p in products,s in scen} >=0, suffix stage 2;
var amountsell{p in products, s in scen} >=0, suffix stage 2;
```

Variables are now scenario-dependent

Stages definition

SAMPL Dakota: HN Results

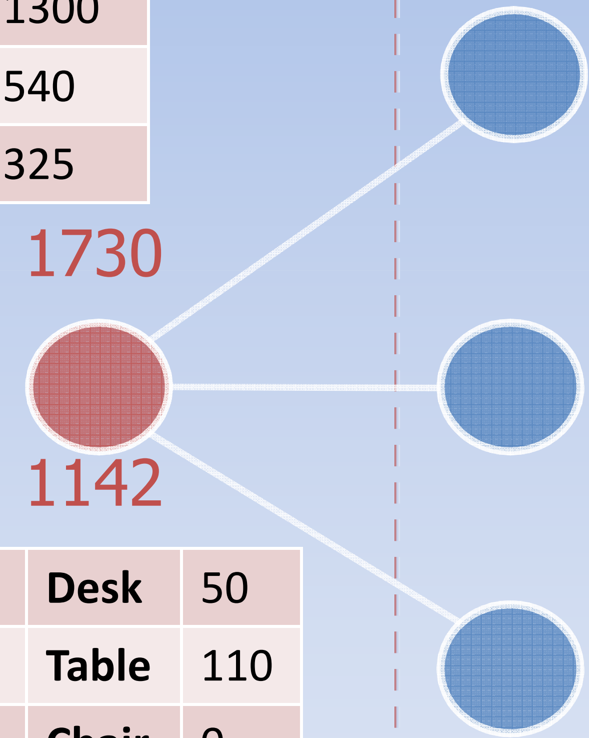
Acquire Resources

Lumber	1300
Finishing	540
Carpentry	325

Objective = 1730

Produce Products → **Sell Products**

		Produce	Sell
LOW	Desk	50	50
	Table	20	20
	Chair	200	200
MED	Desk	80	80
	Table	110	110
	Chair	0	0
HIGH	Desk	80	80
	Table	110	110
	Chair	0	0



Objective = 1142

Lumber	1060	Desk	50
Finishing	420	Table	110
Carpentry	265	Chair	0

Acquire Resources → **Produce Products**

		Sell
LOW	Desk	50
	Table	20
	Chair	0
MED	Desk	50
	Table	110
	Chair	0
HIGH	Desk	50
	Table	110
	Chair	0

Sell Products

CARISMA

Brunel UNIVERSITY WEST LONDON

Scenario Generation

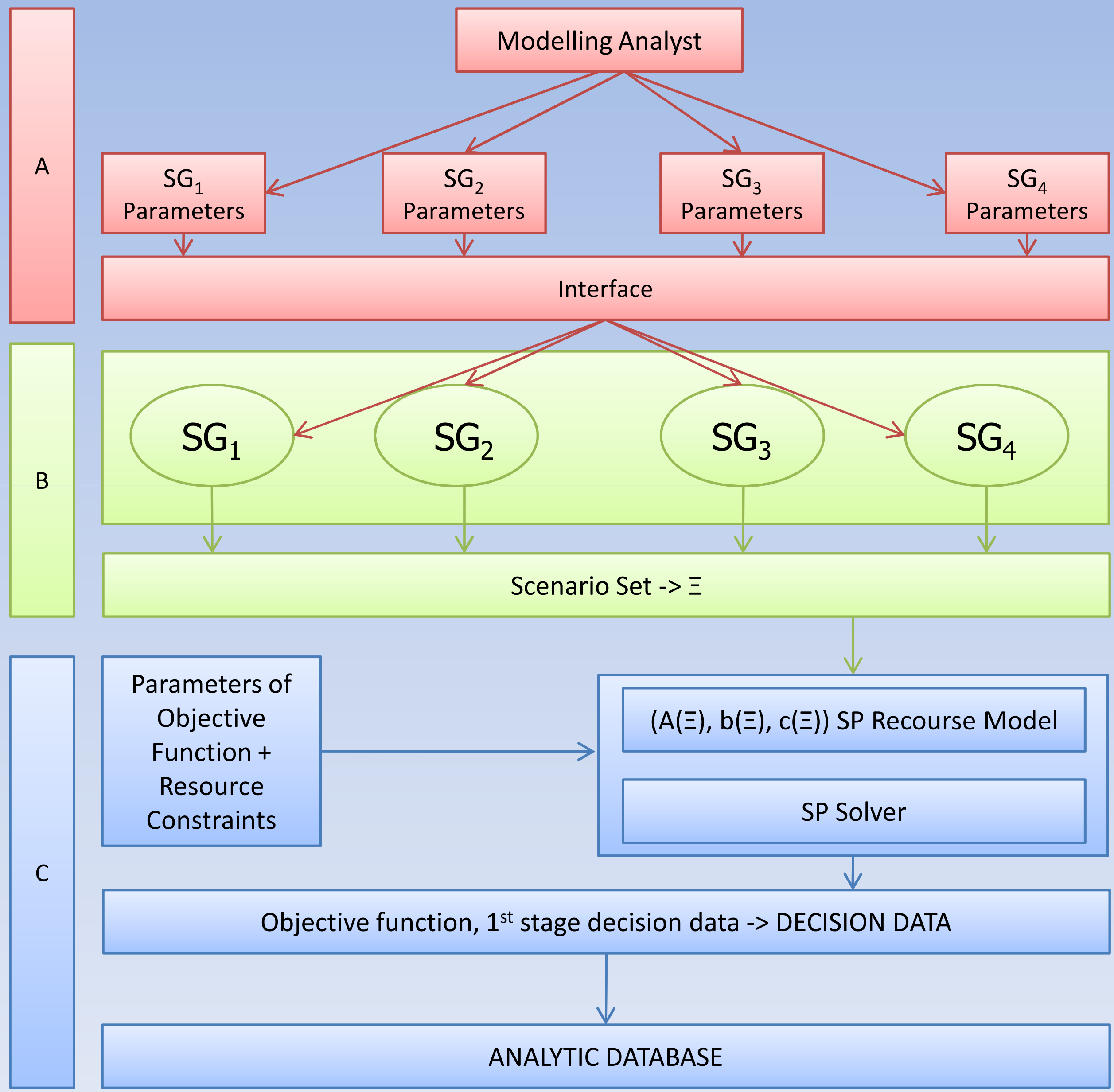
- SPInE is the interpreter for SAMPL language
- Random data definition:
  - In the example, explicit values for the realizations of the random vectors were provided
  - Those are usually obtained through the use of Scenario Generators
  - Direct connection to SGs is currently under implementation
    - Plug-ins architecture
    - SAMPL model declaration remains the same

Scenario Generation

- There are many well established methods - models for describing random parameters
- The models can be summarised as:
  - AR, MA , ARMA, ARCH, GARCH, ...
  - Regression: quantile, robust
  - SDEs, GBM
  - Forecasting: Parametric, nonparametric
  - Simulators: MCMC, HiddenMC, VECM, Bootstrap
- Real applications use Domain specialist's model / knowledge

OptiRisk SYSTEMS

SG in Decision Making - Rationale



CARISMA

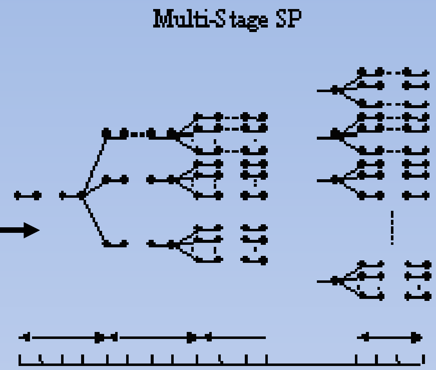
Brunel UNIVERSITY WEST LONDON

OptiRisk SYSTEMS

Stochastic processes and SG

ARCH/GARCH
Robust Regression Time Series
Brownian Motion SDEs
...

→ **Mathematical structure** →

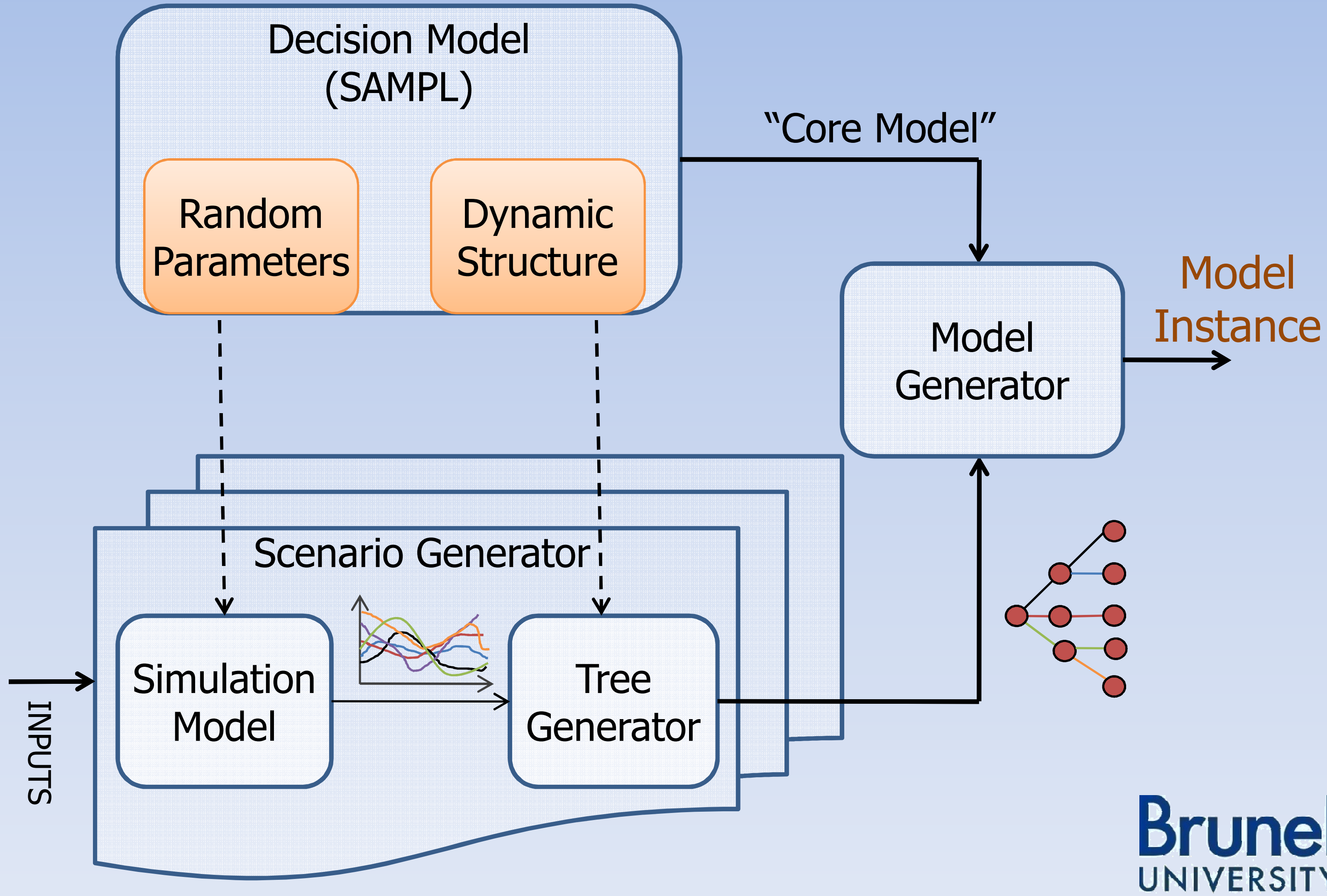


- Stochastic processes generate possible futures in form of fans
- For apply our scenario based approach, we need to map those fans to trees
- The research problems are investigated by: Georg Pflug, Jitka Dupačová, Michael Dempster, Werner Römisch and others

CARISMA

Brunel UNIVERSITY WEST LONDON

SG Library in SPInE

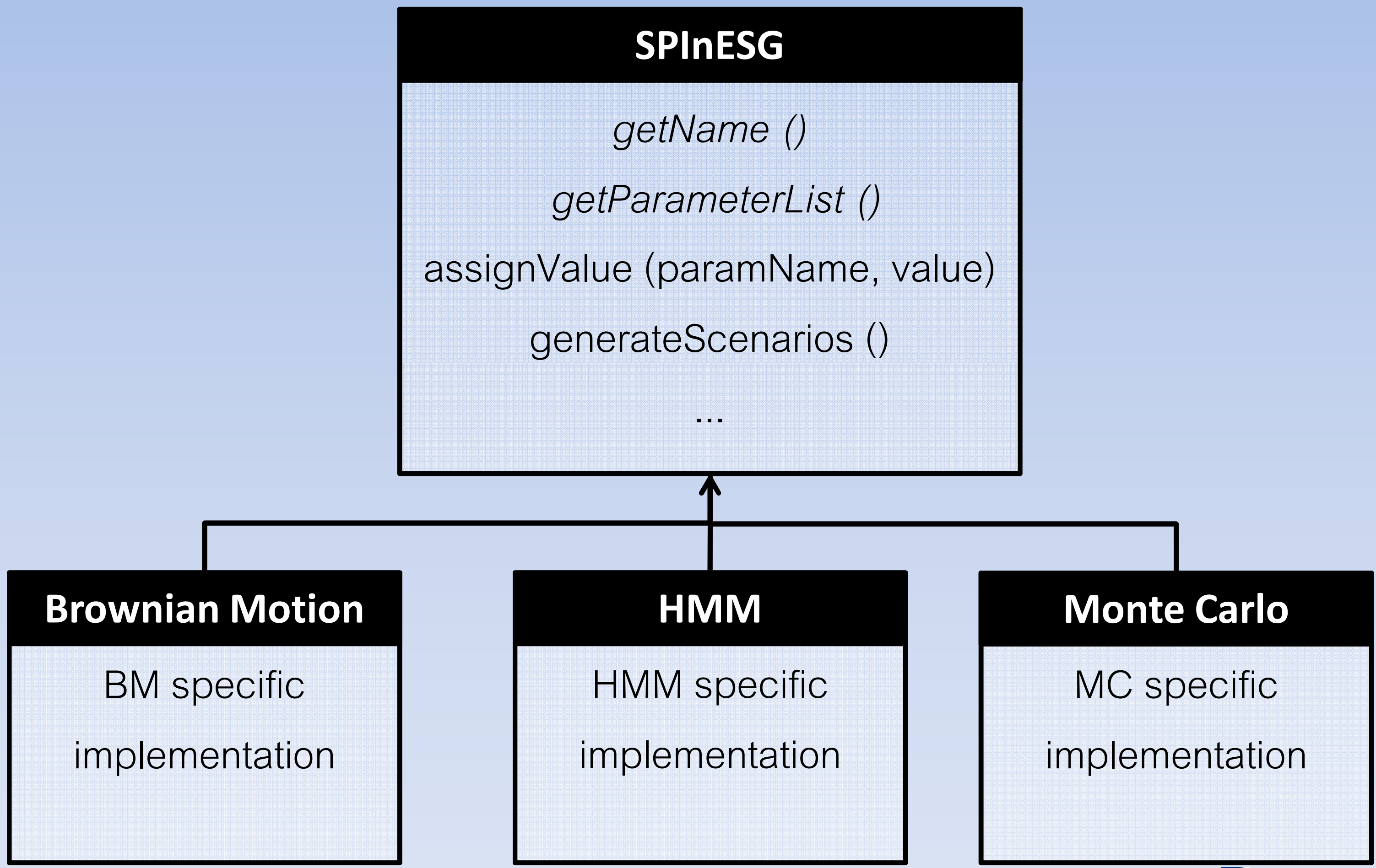


SG Library – Basic concept

- Every scenario generator to be included has to implement the same (simple) interface
- Since different SG methods require different parameters/data, every SG Module has to be self-descriptive
- The interface specifies two types of methods:
  - Descriptive - "*meta-functions*": return the description of the parameters of the SG
  - Functional - "*execution units*" i.e. To start the scenario generation

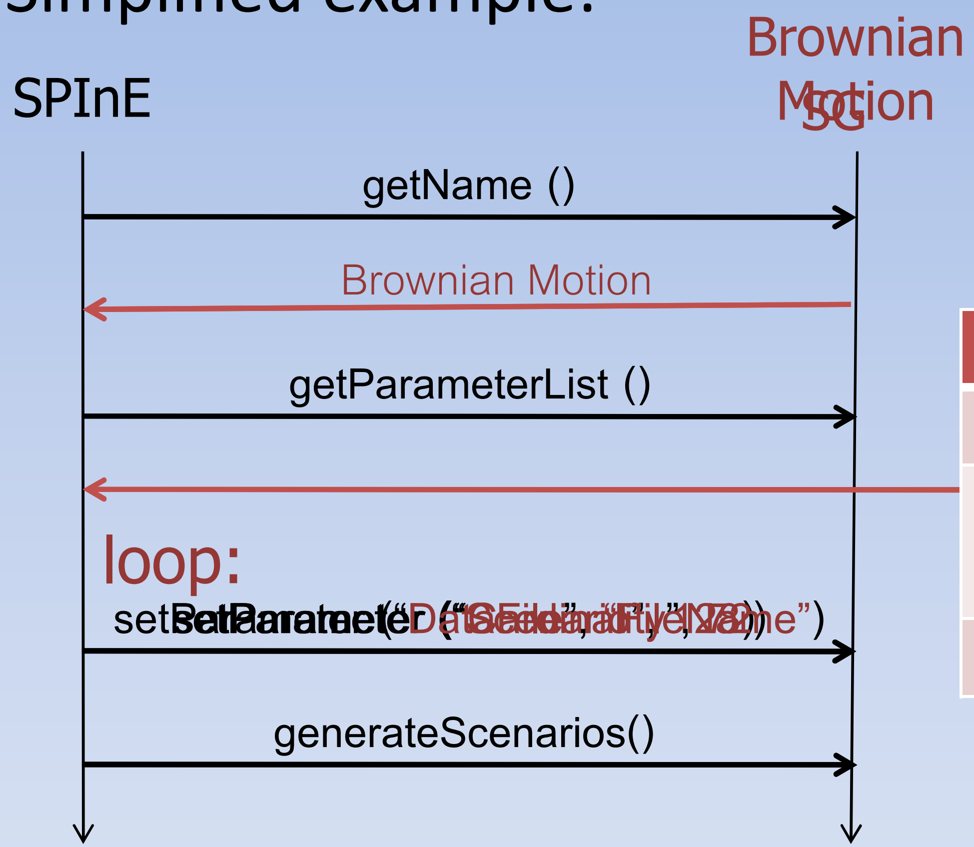
SG Library – Basic concept

- Simplified interface example:



SG Library – Basic concept

- Simplified example:

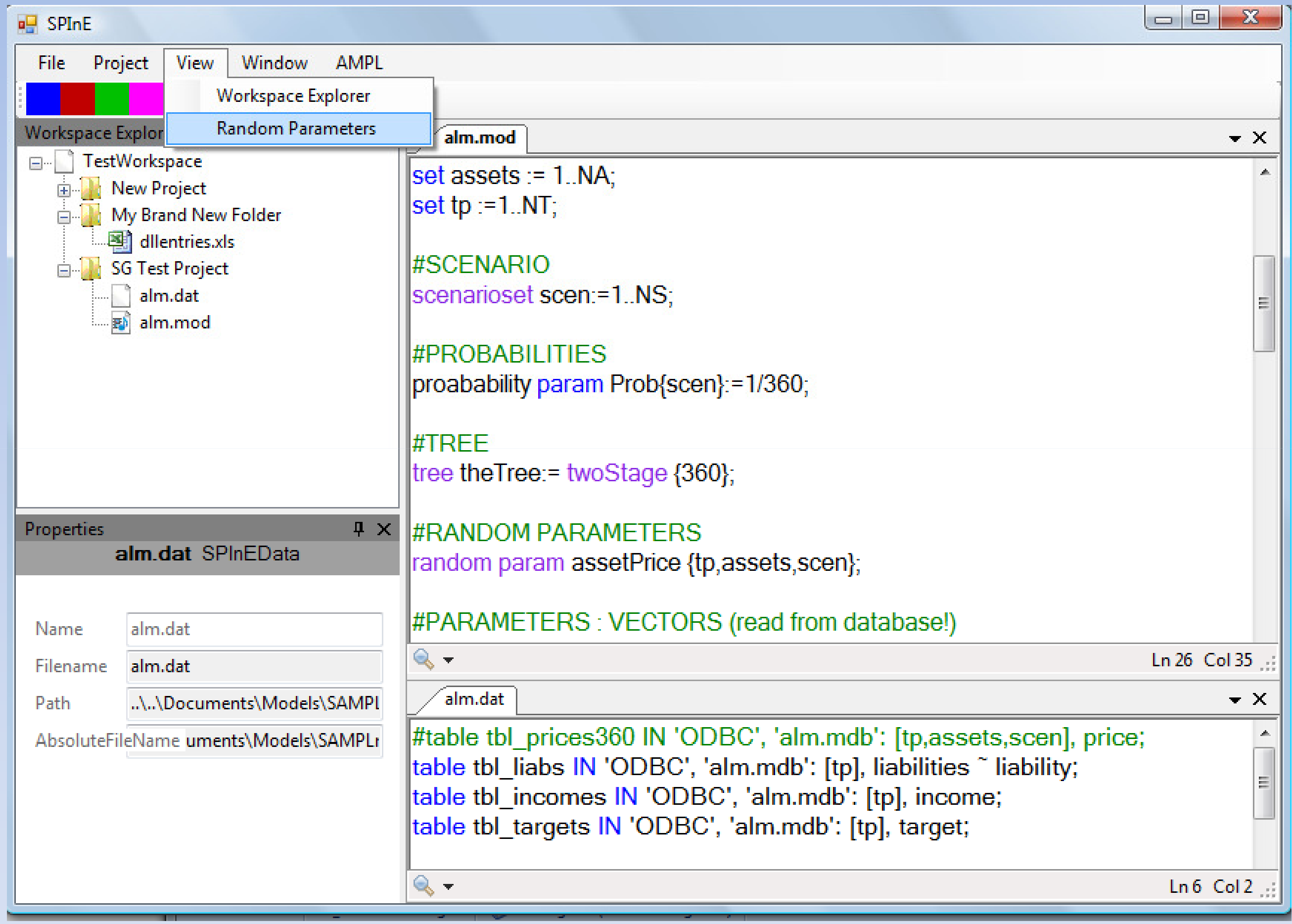


Name	Type	...
Scenario #	Long	
Random Vector Dimension	Long	
DataFile	string	

- The generated scenarios are communicated back and linked to the existing SAMPL model.

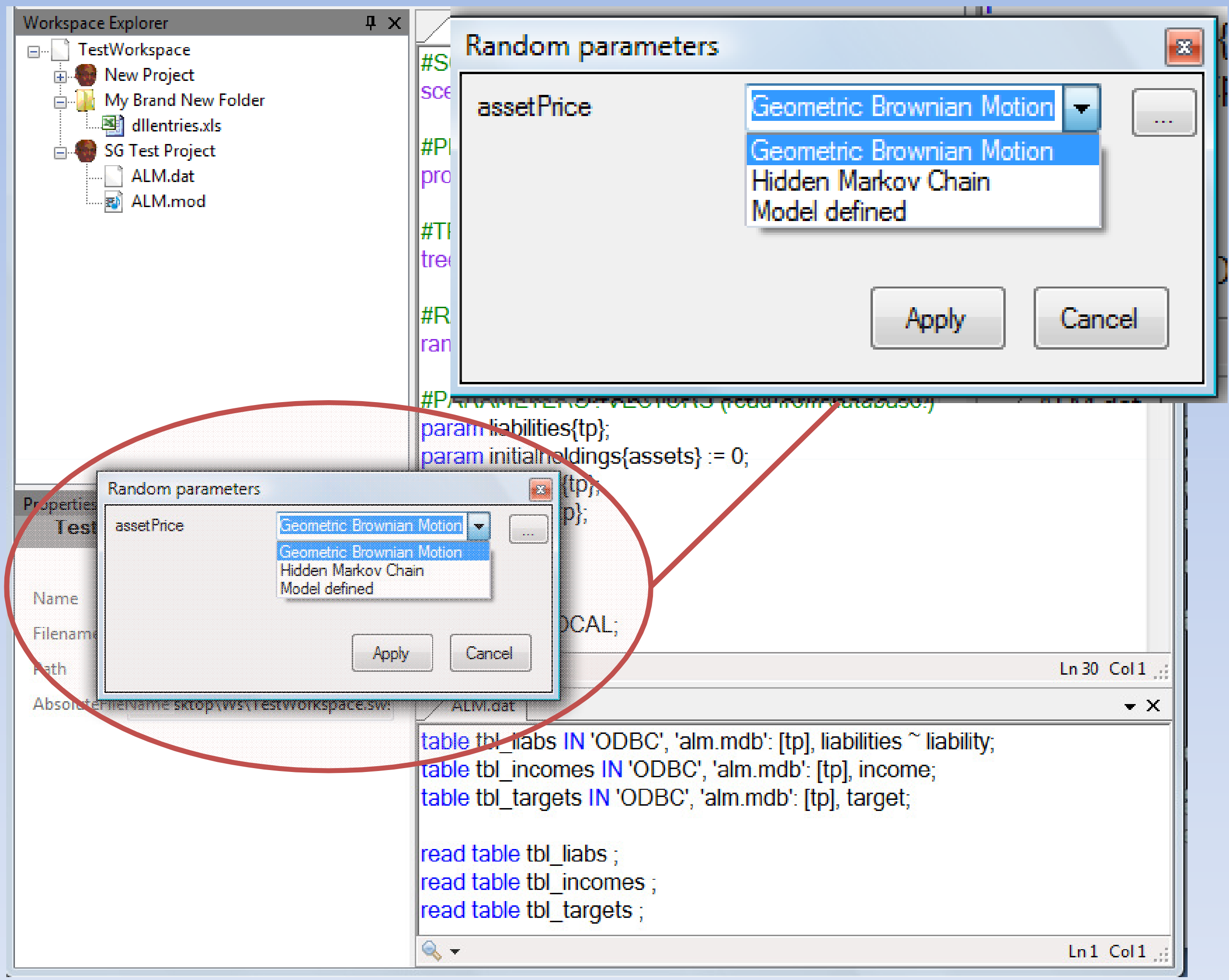
SPInE SG Example

INTRODUCTION
SP MODELLING
Modelling Languages
SAMPL
SG in SPInE
Scenario Generation
Computational Architecture
INVESTIGATION
Simulation and testing
Case studies
SOLVING
Solvers considerations
CONCLUSIONS



OptiRisk SYSTEMS
CARISMA
Brunel UNIVERSITY WEST LONDON

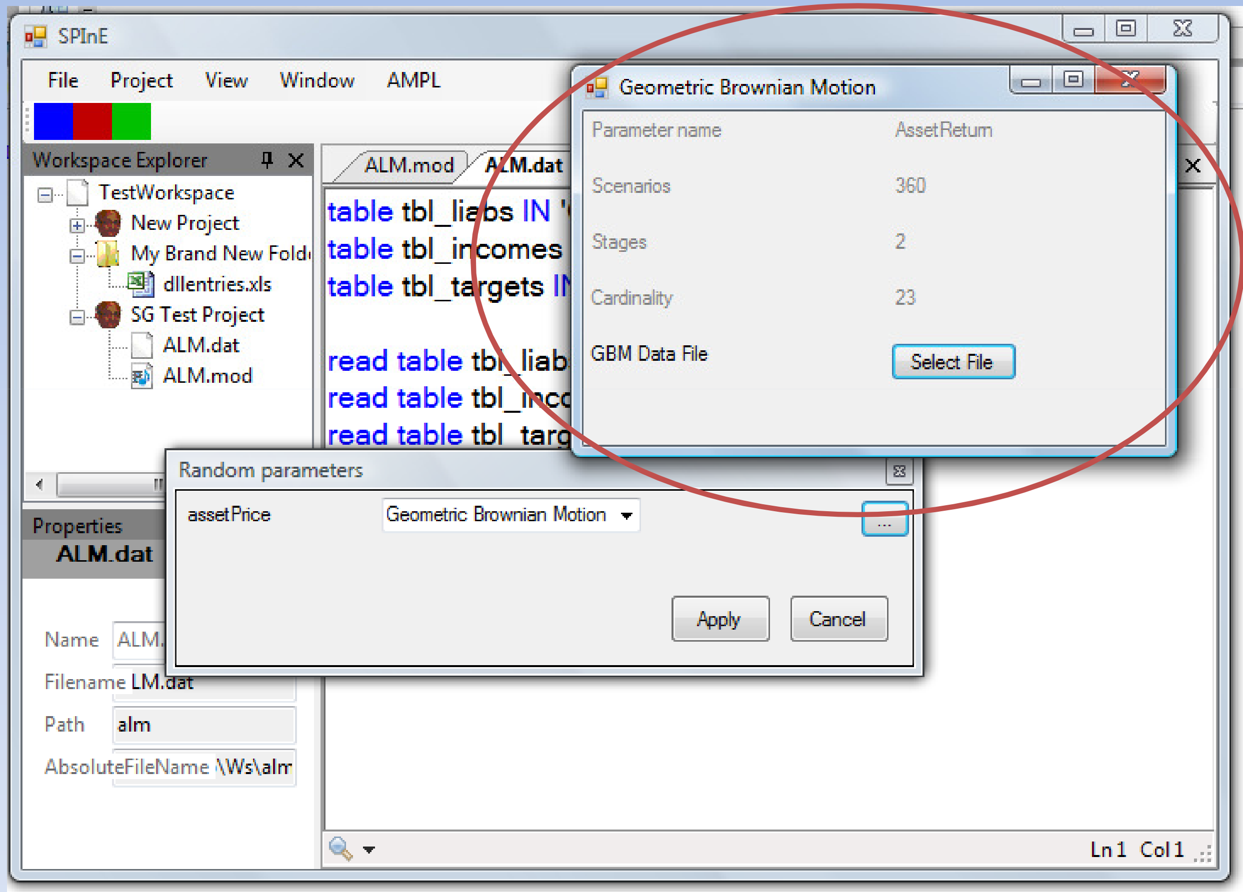
Example: SG selection



OptiRisk SYSTEMS

Example: SG specific settings

INTRODUCTION
SP MODELLING
Modelling Languages
SAMPL
SG in SPInE
Scenario Generation
Computational Architecture
INVESTIGATION
Simulation and testing
Case studies
SOLVING
Solvers considerations
CONCLUSIONS



CARISMA

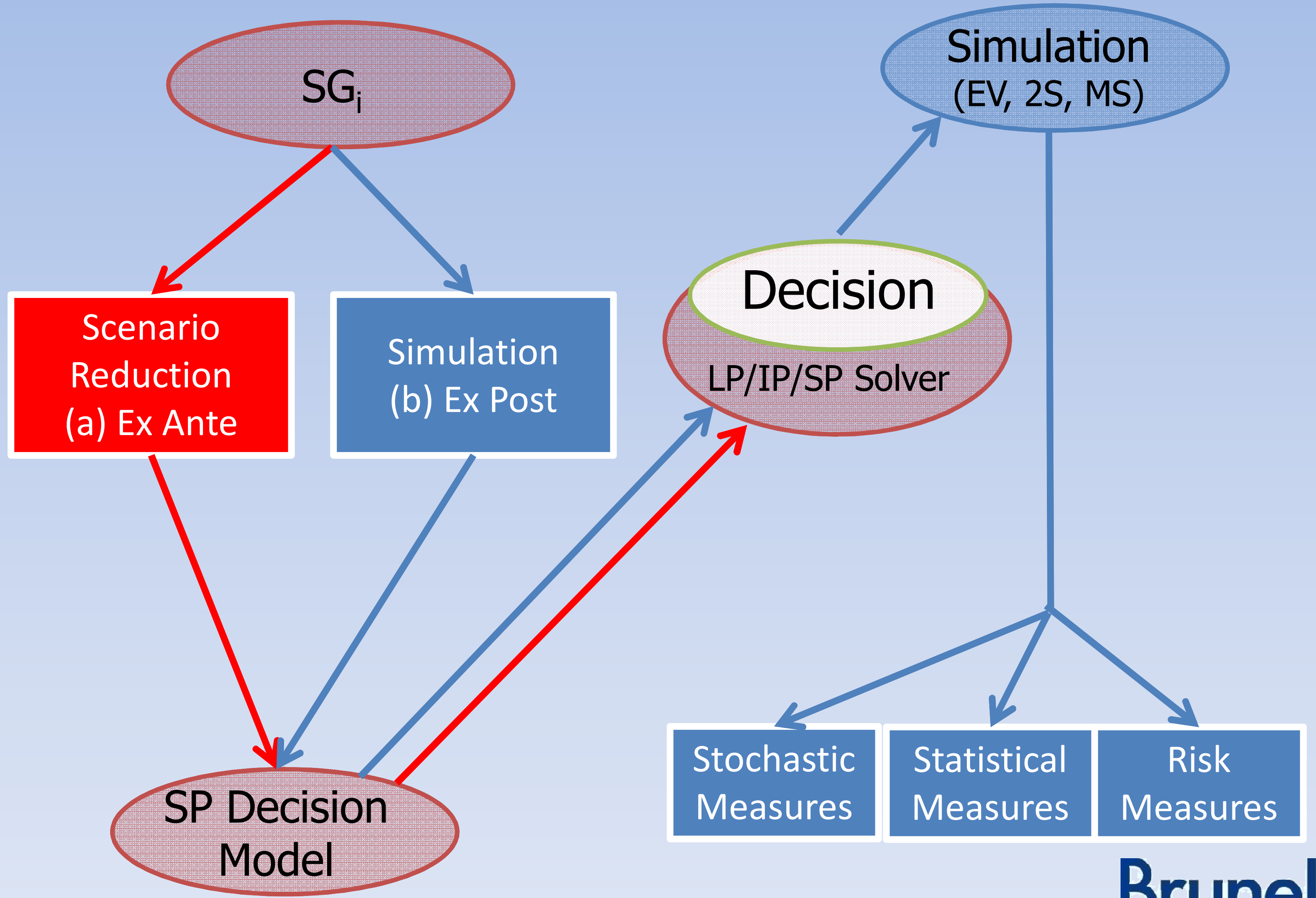
Brunel UNIVERSITY WEST LONDON

Investigation - Rationale

- Emergence of risk analysis has led to novel reuse of established modelling paradigms
- Ex-ante decisions coupled with ex-post evaluation (combined paradigm: optimisation and simulation) is a method of choice in many applications
- Work in progress: the design of a user friendly but "general enough to be useful" framework

OptiRisk SYSTEMS

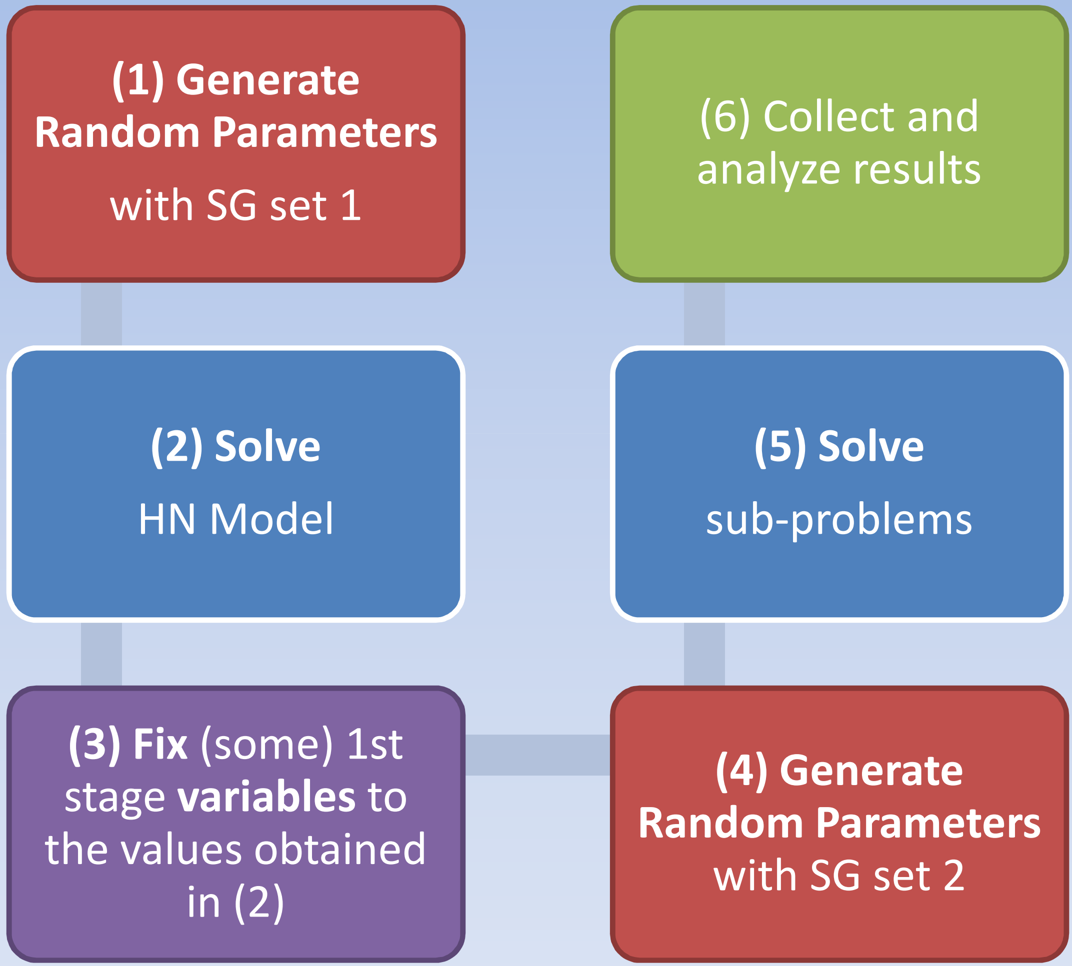
Simulation and testing



CARISMA

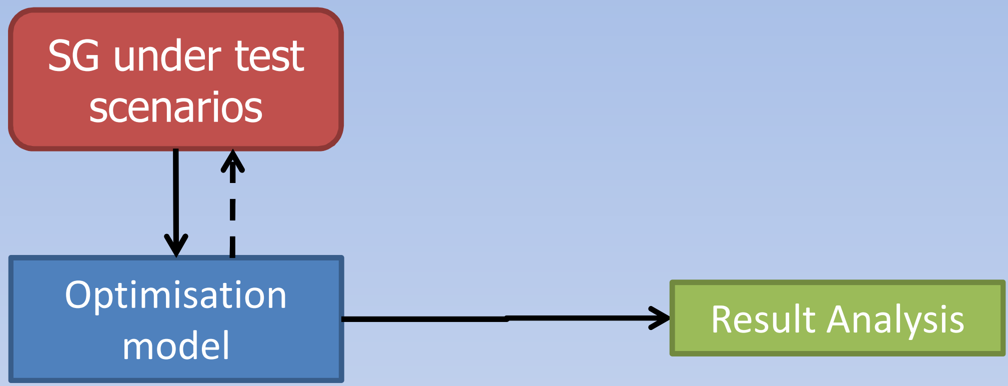
Brunel UNIVERSITY WEST LONDON

Simulation and testing



OptiRisk SYSTEMS

SG stability test – in sample

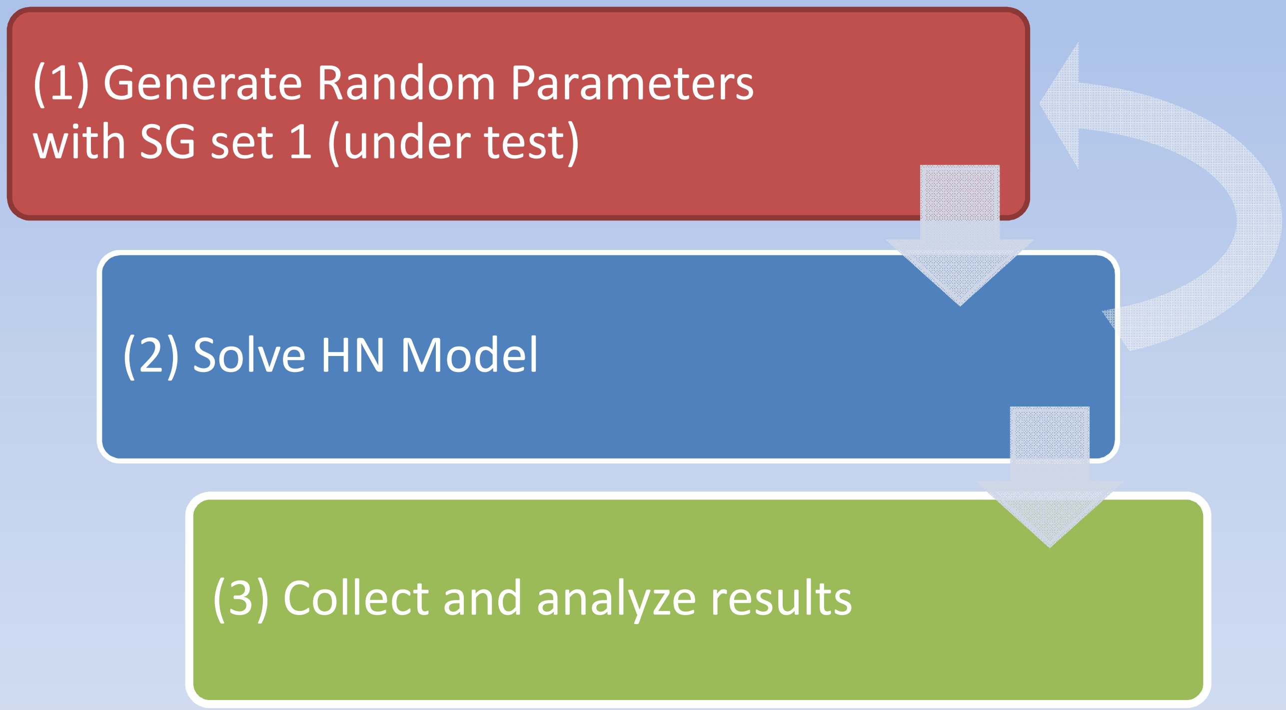


- For the in-sample stability of a Scenario Generator, we repeatedly solve the model with different instances of the SG under test
- Then we analyze the distribution of the objective values

CARISMA

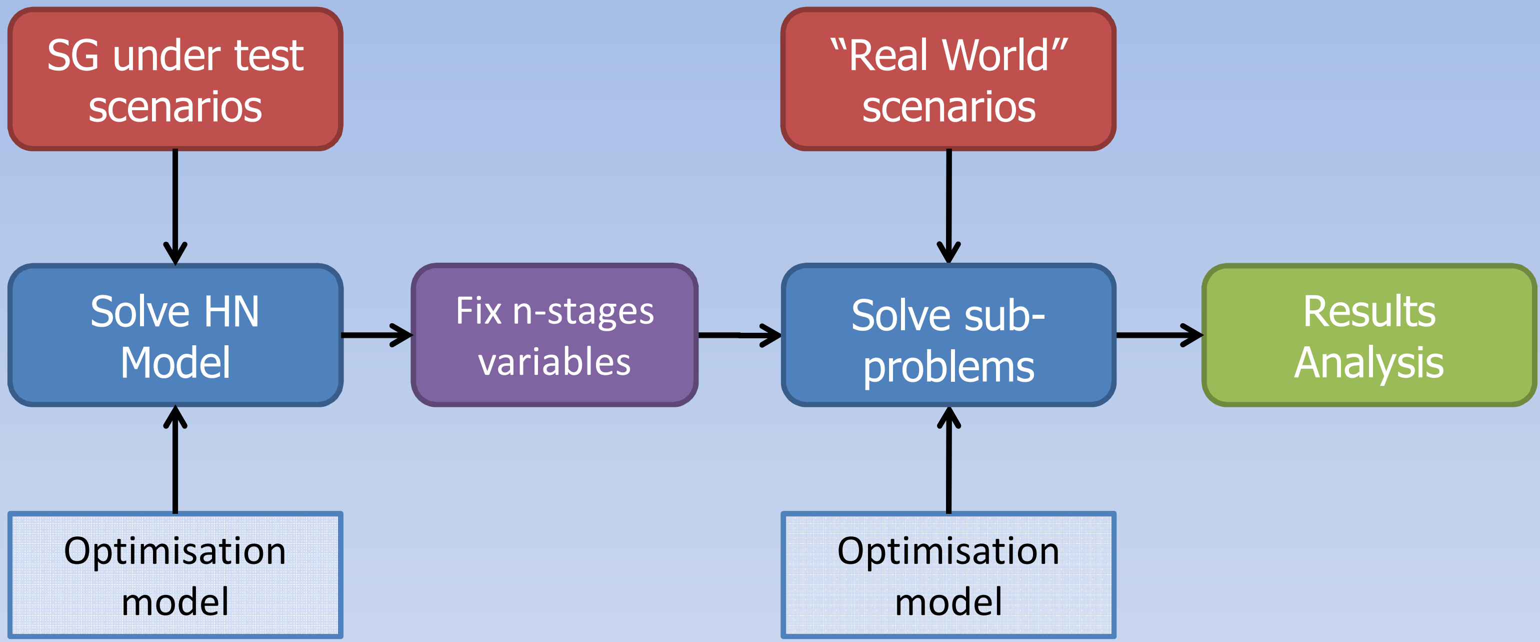
Brunel UNIVERSITY WEST LONDON

Simulation and testing



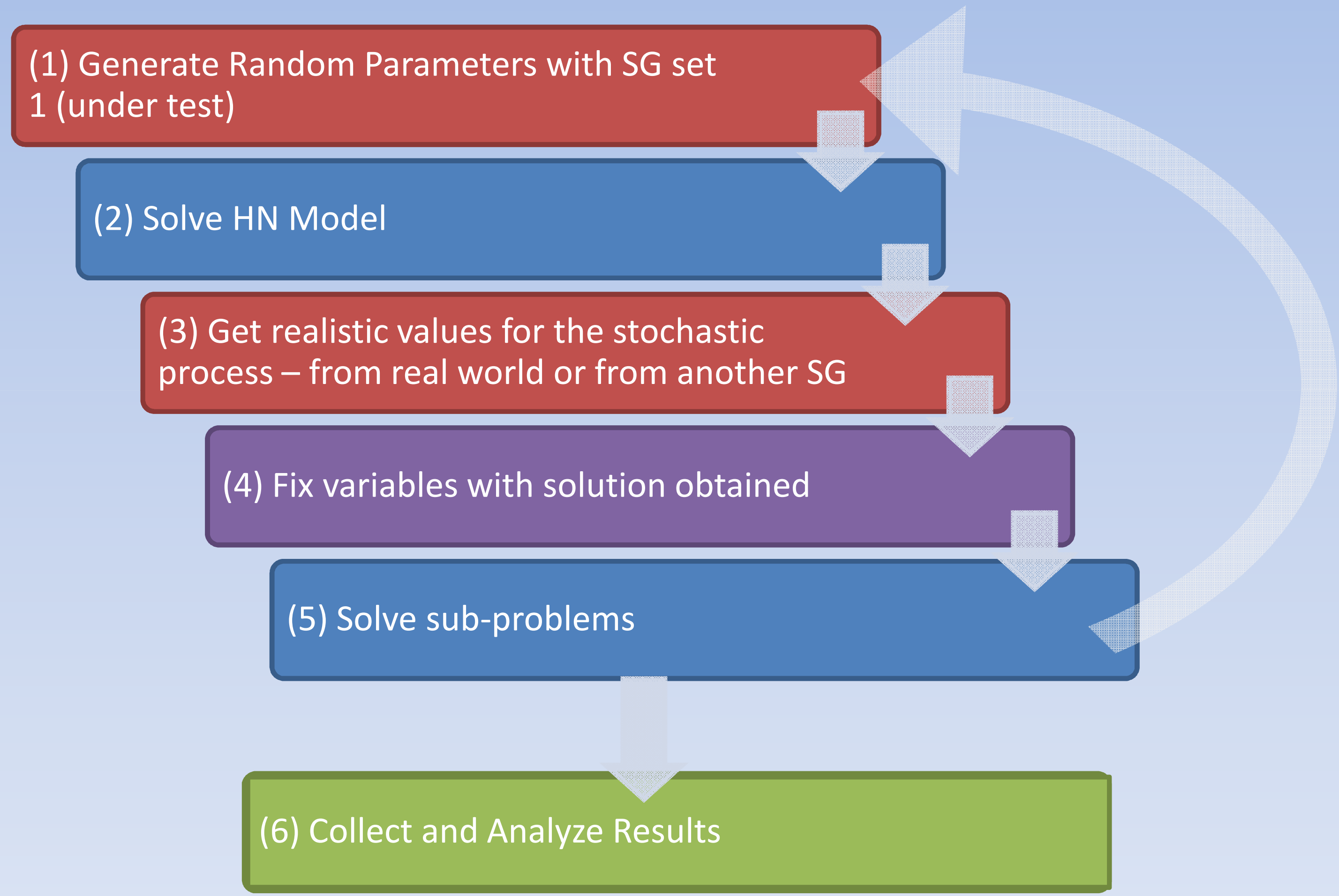
- A different view is using the same blocks as before, organized in a slightly different way

Out of sample SG testing



- "Real world" scenarios is a large scenario tree, which is assumed to be the best available approximation of the stochastic process

Out of sample SG testing



Simulation and testing

- The design of the framework to allow various kind of testing is still in progress
- Natural approach seems to be the workflow one, in which a few specialised modules can be organized in a user defined way to perform potentially complex tasks

Generate Scenarios
Define SG set / variable bindings

Collect Data
Data Analysis is the final aim of simulation

Fix Variables
Define which variable is bound to which other

Solve Model
With specified solver / solver's settings

Solving SP Problems

- The solution of SP problems is difficult
- Decomposition based methods allow to greatly speed up the solution of the Here and Now (recourse) model
- SPInE includes three solvers:
  - FortSP: can use DEQ as well as Benders decomposition
  - FortSD: can use Stochastic Decomposition [Sen and Higle, 2000]
  - FortLD(*): uses Level Decomposition [Fabian, 2006]

Solving SP Problems

- As seen before, for simulation/testing there's often the need to fix the first stage variables and solve all the sub-problems
- In the important class of the two stage models, this leads to solve a lot of models that differ from each other just for a few parameters
- Those parameters are the stochastic ones in the second stage of the original model
- This can be viewed as a *family* of models

Solving SP Problems

- The use of *warm restart* usually helps in the solution of the sub-problems:
  - store the optimum basis factors information in a back up store,
  - retrieve the basis factor information from the back up store and continuing with processing.
- We find that in general the average number of iterations required to solve the problem from 'scratch' is much bigger than if starting from the neighbouring optimum solution
- SPInE is designed to invoke base restart functionalities of solvers – where present

Future Developments

- Work in progress:
  - Extend the SG library
  - Update the model instance format (currently SMPS)
  - Allow dynamic structure to be specified in the scenario generators
  - Implementation of the simulation/decision evaluation environment
  - Use of OpenMP to exploit multi-core/multi-processor computers, especially for simulation