Compositional Model Conversion

Mohd Fadzil Hassan

MSc in Artificial Intelligence
Division of Informatics
University of Edinburgh
2001

Abstract

This dissertation presents an initial work towards the development of a technique to convert compositional models from one modelling paradigm to another, by means of a pair of equivalent compositional modelling domain theories. The mapping between model fragments of the two domain theories is not necessarily in a one-to-one manner. It might be the case that a model fragment in one domain theory covers parts of several model fragments in the other domain theory. This is one of the major conversion problems that this technique will focus on.

The compositional modelling of ecological systems is used as a testing domain for the implemented conversion technique. For this work, system dynamics and object-oriented representations are the two modelling paradigms adopted. The major intention of this conversion application, implemented in the C++ programming language, is to convert a system dynamics model, composed through a compositional modelling technique, to an object-oriented model. The resulting object-oriented model is expected to reflect the same scenario, but with a different representation, compared to the model produced within the system dynamics modelling paradigm.

Acknowledgements

I would like to thank the following: my supervisors, Dr. Qiang Shen and Mr. Jeroen Keppens, for their patience, careful guidance, suggestions and constructive criticisms during the course of the research reported here; my parents, Fatimah Khamis and Hassan Abdullah for their constant encouragements; my MSc colleagues, for their useful discussions and camaraderie during my stay and study here in Edinburgh; and my sponsor, the PETRONAS University of Technology, Malaysia who provided me the opportunity to study at this excellent institute.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General overview

The essence of model building is to decide which aspects of the world should be explicitly described in the model, and which should be omitted. Paraphrasing Einstein, a model should be as simple as possible, but no simpler; as complicated as necessary, but no more (Kuipers, 1994).

Compositional modelling addresses the problem of building a model, and it is one of the predominant approaches to automate the deduction of mathematical models of a system. A compositional modeller utilises a potentially vast background knowledge base, also termed as *domain theory*, that describes a class of related phenomena or systems. A domain theory consists of a set of *model fragments*, each describing some fundamental pieces of the domains physics, such as processes (e.g. liquid flows), devices (e.g. transistors), and objects (e.g. containers). The system or situation being modelled is termed as the *scenario*, and its model, the *scenario model*. Given a scenario describing the system under consideration, the compositional modeller instantiates applicable model fragments from the domain theory, selects the most appropriate ones, and composes them into a scenario model.

As the compositional modelling techniques have been devised to select a set of model fragments from a range of alternatives, different models can be deduced from the same scenario under different requirements (and sometimes, even with the same modelling requirements). This is possible even with a small knowledge base because the model fragments contain reusable model parts and the instantiation of these parts may not uniquely determine what parts to select and combine. Furthermore, due to the domain independent nature of this technique, a relatively wider variety of modelling paradigms may be implemented when compared to other automated modellers.

As multiple modelling paradigms may be used in the knowledge representation of a domain theory, given the same scenario and modelling environments, it is possible to compose a number of scenario models, each corresponds to a different modelling paradigm. Therefore, this project requires the development of a technique to convert models from one modelling paradigm to another, by means of a pair of equivalent compositional modelling domain theories, each implemented in a different modelling paradigm.

1.2 Objective

This project involves the development, implementation and testing of a technique for transforming compositional models between different representations. The primary aim of this technique is to enable a scenario model in one particular representation, composed by a compositional modeller, to be converted reliably to different representations given a particular set of domain theories.

1.3 Motivation

Formulating a model via compositional modelling technique requires searching the modelling space to find an adequate set of model fragments to instantiate. Based on this technique, it is possible to model a scenario from many different perspective, based on the preference of human users. However, many existing conventional engineering application of compositional modelling are mostly involved with a single modelling paradigm or representation. Therefore, given this domain, the proposed conversion technique might not be really crucial. However, for certain important problem domains, for instance compositional modelling of ecological systems, this technique will be really beneficial as ecological models may employ a variety of modelling paradigms. To some extend, many of these modelling paradigms may just express the same scenario model in different ways. However, each paradigm of model construction has its own useful features and there is little agreement amongst expert ecologists with respect to the modelling language they use, and what constitutes an ideal modelling paradigm. Therefore, this approach may help bridge the gap between modelling paradigms and languages, and possibly, enable the exploration of the similarities and differences between modelling paradigms.

1.4 Scope

Due to the domain-independent nature of compositional modelling technique, a relatively wider variety of modelling paradigms or representations may be implemented when compared to other automated modellers (Keppens and Shen, 2001). However, this feature has not been fully explored because most typical applications only involve a single modelling paradigm. Exploring the many different modelling paradigms which are applicable, is beyond the scope of this project, nonetheless it will concentrate on the following issues:

1. Developing a conceptual technique which can be utilised to reliably convert models from one modelling paradigm to another.

2. As a prove of concept, the developed technique is implemented using a programming language.

3. Testing the technique that has been implemented. In the testing, two equivalent compositional modelling domain theories that utilise two different modelling paradigms are minimally required. Thus, the responsibility of developing these domain theories is part of this project.

Eventhough various techniques in compositional modelling exist, the compositional modelling formalisms developed by Falkenhainer and Forbus (1991) are adopted to complement the proposed conversion technique in this project.

1.5 Implementation and testing

Ecological domain is used as a testing platform for the implemented conversion technique. The major intention of this testing application, implemented in the C++ programming language, is to convert a system dynamics model, composed through a compositional modelling technique, to an object-oriented model. The resulting object-oriented model is expected to reflect the

same scenario, but with a different representation, compared to the system dynamics model. For example, in the system dynamics modelling paradigm, a phenomena of interest may be featured via levels and flows between them, while within the object-oriented modelling paradigm, it is represented as object of classes, and the interaction between these objects is provided by the class attributes and methods.

1.6 Outline of dissertation

- Chapter 2 consists of description on compositional modelling. It also contains a description of the two modelling paradigms intended to be used as a testing platform in this work, ecological-modelling with system dynamics and object-oriented paradigms.

- Chapter 3 consists of description on the theoretical design underlying the proposed conversion technique, with the stages involved in this technique.

- Chapter 4 contains a description on how the proposed theoretical design of the conversion technique is implemented. Algorithms of the processes involved and the details of the implementation are discussed.

- Chapter 5 discusses about the testing that have been conducted and the results that have been obtained.

- Chapter 6 contains a summary of this work and some suggestions for future work.

Chapter 2

Background

2.1 Overview of compositional modelling

This section presents a general overview of compositional modelling, with the stages involved in formulating a model that is relevant to the needs of the task and consistent with the operating conditions of the system.

Compositional modelling is one of the predominant approaches for automatically formulating a model, in order to help human analyse a wide variety of physical-system behaviour in a given domain (Iwasaki, 1997). An important feature of Compositional Modelling is that it makes most of the modelling process explicit (Collins and Forbus, 1990). That is, knowledge of the physical world is organised as a domain model, which describes the basic conceptual entities and phenomena. Given a particular physical situation, constructs of the domain model are combined to form a scenario model of the specific situation.

2.1.1 The compositional modelling task

Figure 2.1 presents a generic architecture for compositional modellers. A scenario description and task specification are provided to the compositional

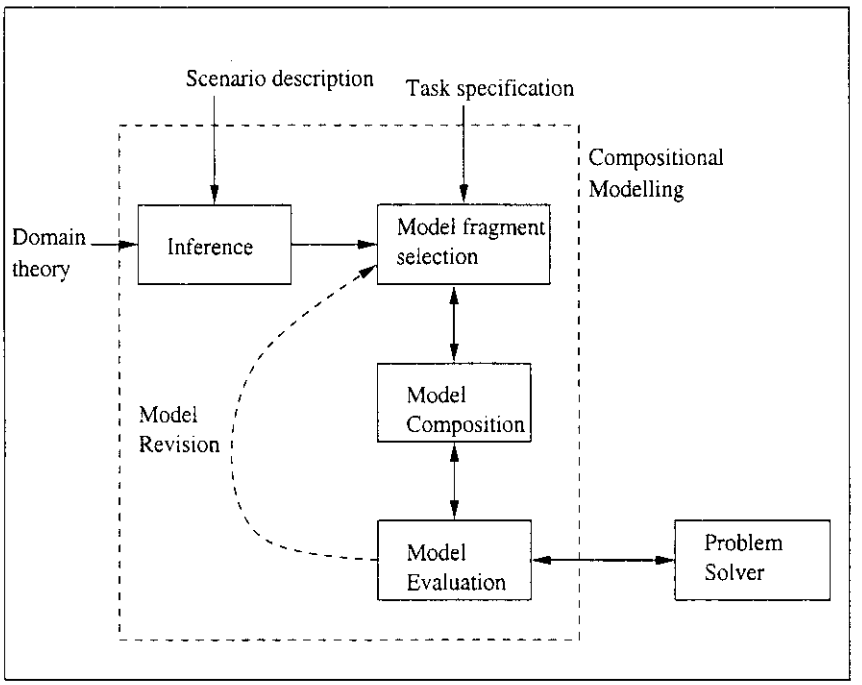

Figure 2.1: Generic architecture of compositional modellers

modellers as input. The *scenario description* constitutes the technical level input and the *task specification* is a formal description of the criteria imposed upon the model to be composed. Task specifications come in a variety of form, and they are usually represented as a *query* or as an *initial state* from which the problem solver must extrapolate future behaviour (Keppens and Shen, 2001).

First, given the scenario description to be modelled, a modeller will make an inference on this information to instantiate the constructs of the domain theory, such as model fragments and rules that apply to the scenario. The *model fragments* represent how certain components, processes or concepts can be modelled. In the *model fragment selection* stage, given the task specification on hand, a subset of the instantiated model fragments generated by the *inference mechanism* is selected. In the *model composition phase*, the selected instantiated model fragments are composed into a model. Vari-

ous techniques, namely *consistency checking* and *causal ordering* techniques, might be employed at this stage in order to produce an appropriate scenario model (Levy et al. (1997); Nayak (1995)). The models generated during the model composition phase are to be used by the problem solver. However, as not all models are equally suitable, each alternative model is assessed and the best alternative is passed on to the problem solver in the *model evaluation* phase. New information may be derived that contradicts earlier assumptions in the model evaluation and problem solving stages. This information is fed back to the model fragment selection phase, which replaces the affected model fragment, hence, revises the model accordingly.

2.1.2 Model fragments

The domain theory of a compositional modeller consists of composable pieces of sub-system models called model fragments, which describe numerous aspects of physical phenomena. The word *phenomena* generally includes the notion of physical objects, behaviour characteristics of objects or combination thereof, and physical processes (Iwasaki, 1997). In Keppens and Shen (2001), the content of a model fragment has been formalised as follows:

A model fragment, μ is a tuple $< P^s, P^t, C^s, C^o, C^t, A >$ where

- $P^s(\mu) = \{p_1^s, ..., p_m^s\}$ is a set of *source-participants*.
- $P^t(\mu) = \{p_1^t, ..., p_m^t\}$ is a set of *target-participants*.
- $C^s(\mu) \cup C^o(\mu)$ is a set of *preconditions*, where $C^s(\mu) = \{c_1^s, ..., c_v^s\}$ is the set of *structural conditions* and apply over the vector of source-participants $p_s^{\rightarrow}(\mu) = \{p_1^s, ..., p_m^s\}$ and $C^o(\mu) = \{c_1^o, ..., c_w^o\}$ is the set of *operating conditions* and apply over the vector of target-participants $p_t^{\rightarrow}(\mu) = \{p_1^t, ..., p_m^t\}$.

- $C^t(\mu) = \{c_1^t, ..., c_u^t\}$ is a set of *post-conditions* which are constraints that apply over $\overrightarrow{p_t}(\mu)$,
- $A(\mu) = \{a_1, ..., a_s\}$ is a set of *assumptions*,

such that for $i = 1, ..., u$

$$\forall p_1^s, ..., p_m^s, \exists p_1^t, ..., p_n^t$$

$$c_1^s \wedge .. \wedge c_v^s \rightarrow (a_1 \wedge .. \wedge a_s \rightarrow (c_1^o \wedge .. \wedge c_w^o \rightarrow c_i^t))$$

The source and target participants in a model fragment are variables representing domain objects. These objects may be entities or subsystems of the real-world system of interest. Alternatively, they may be conceptual entities that, when instantiated, will assume the role of variables within the scenario model, thus representing significant properties of the system. The relations that exist between objects represented by the participants are defined by the conditions in the model fragments.

An example of model fragment is given in figure 2.2. This fragment applies to population P. If applied, it introduces two new target-participants to the scenario model: the biomass B of the population and an intrinsic-rate-of-increase R of the population. The implication that is formalised by this model fragment is that, if B, which defines population size, is greater than 0 and the growth-phenomenon of P is modelled, then the rate of increase of size P at time t (dB/dt) is equivalent to the multiplication R * B (Mackenzie et al., 1998).

2.2 Ecological modelling

In this work, ecological modelling, particularly population ecological modelling, is used as a testing domain for the implemented conversion technique.

Subject	Participant(s)
Source-participant(s)	P
Structural condition(s)	population(P)
Target-participant(s)	B,R
Operating condition(s)	B > 0
Assumption(s)	consider(relevant(growth(P)))
Post-condition(s)	biomass(B,P) intrinsic-rate-of-increase(R,P) dB/dt = B*R

Figure 2.2: Model fragment to define population growth phenomenon

Two ecological domain theories, one designed with the system dynamics modelling paradigm, and the other with the object-oriented modelling paradigm are utilised for this purpose. As population ecology is a broad subject by itself, and consists numerous phenomena of interests which are possible to be modelled, this section briefly decribes the phenomena that this work will focus on. In the next two subsections, the description of the two modelling paradigms used is given.

Colinvaux (1986) defines *population* as a group of organisms of the same species, coexisting at the same time and place and capable for the most part of interbreeding. In this work, two ecological phenomena involving population, namely population growth phenomenon and predation phenomenon are examined.

Models of population growth are basically derived from two theoretical point of views (Mackenzie et al., 1998). Density-independent or exponential population growth is described by a continuous population model where growth is unlimited, and expressed in terms of the rate of change in population numbers at time t :

```
Rate of change of population = Intrinsic rate * Population
size at time t                 of increase      size
```

The above equation is represented with the following simplified notation; $\frac{dN}{dt} = r * N$.

On the other hand, the density-dependent growth, which is also termed as the logistic equation, describes the growth of a simple population in a confined place, where resources are not unlimited. The population increases geometrically until the maximum number of individuals, that the environment can sustainably support, is approached. The maximum number is called the carrying capacity (K). The population growth rate declines to zero as the population becomes more crowded and the population size stabilizes. This can be described as the logistic equation:

```
Rate of change of  = Intrinsic  * Population * Density
population size      rate of      size          dependent
at time t            increase                   factor
```

The above equation is represented with the following simplified notation; $\frac{dN}{dt} = r * N(1 - \frac{N}{K})$.

where the density-dependent factor $(1 - \frac{N}{K})$ approaches zero as the population growth approaches the carrying capacity.

Predator is defined as an organism that uses other life organisms as an energy source and, in doing so, reduces the prey individuals from the ecological system (Colinvaux, 1986). Therefore, the predation of the predator on the prey is one of the crucial aspects that have to be considered in modelling the growth phenomenon on both populations. A variety of models have been developed to explore predator-prey dynamics. The Lotka-Volterra predator-prey model is a simple yet valuable example, and the following equations describe the growth rate of the prey and predator populations:

$\frac{dX}{dt} = r * X - (f * X * Y)$
$\frac{dY}{dt} = f * X * Y - (z * Y)$
where

- X and Y respectively are the prey and predator population size
- r - intrinsic growth rate of the prey
- z - mortality rate of the predator
- f - interaction coefficient giving the feeding rate of the predator on the prey

The Holling-Tanner model (Holling, 1965), is another variation of the Lotka-Volterra predator-prey model. In the basic Lotka-Volterra predator-prey model, the predation rate, i.e. the number of prey consumed per unit

time per predator, is directly proportional to prey density. This relationship is known as the *predator functional response f(N)*, and there is no saturation or upper limit of predation and no account is taken on the effect of predator density on predation. Therefore, it allows for a situation where predation can increase irrespective of how low the predator numbers are. Hence, in order to make the model more realistic, an upper limit on the predation function is imposed, which is known as the Holling's disc equation. This Holling-Tanner model presents a limited prey and predator population growth, together with the Holling disc equation, as a functional response of predation rate to prey density. The equations of this model are as follow:

$\frac{dX}{dt} = r * X * (1 - \frac{X}{K}) - \frac{w*X*Y}{Y+X}$
$\frac{dY}{dt} = f * X * (1 - \frac{b*X}{Y})$
where

- X and Y respectively are the prey and predator population size
- r - relative growth rate of the prey
- f - relative growth rate of the predator
- K - carrying capacity
- b - number of prey required to support each predator at equilibrium
- $\frac{w*X*Y}{Y+X}$ - Holling disc equation, where w is a constant representing maximum rate of predation

Begon et al. (1996), Colinvaux (1986) and Mackenzie et al. (1998) provide more elaborate description on these models of population ecology.

2.2.1 Ecological modelling with system dynamics

A most common modelling paradigm used in ecology is *system dynamics* (Forrester, 1961). In system dynamics, the phenomena of interest are represented as levels and flows between them. The change in the level's unit over time is equal to the total of inflows minus the total of outflows. Additional variables and influences describe the relation between the levels and flows. As such, system dynamics provides an interface to modelling with differential equations and allows features of other paradigms to be integrated (Robertson et al., 1991). The following scenario, based on the work of Keppens and Shen (2000), is used in order to illustrate the system dynamics approach to ecological modelling:

$$population(predator) \wedge population(prey) \wedge feeds(predator, prey)$$

It describes a world consisting of two populations - a predator population that feeds on the prey population. In the given scenario, various phenomena can be considered. For instance, reproduction within both populations and the predation behaviour of the predator with respect to the prey population might be of relevance.

Many models of these different phenomena exist. Consider the predation phenomena involving prey, of which figure 2.3 shows one possible model. In addition, a growth phenomenon conceptualising changes in population size is necessary for both populations. This is because predation affects the change in the level of the prey population since predation kills prey, as well as the change in the level of the predator population since the total amount of available prey affects the sustainable population.

Referring to figure 2.3, the growth phenomenon is represented by a level, an inflow and outflow for both populations. In the given case, a simple linear reproduction model is used which is limited by a maximal sustainable population level. The capacity of the predators depends on the predation

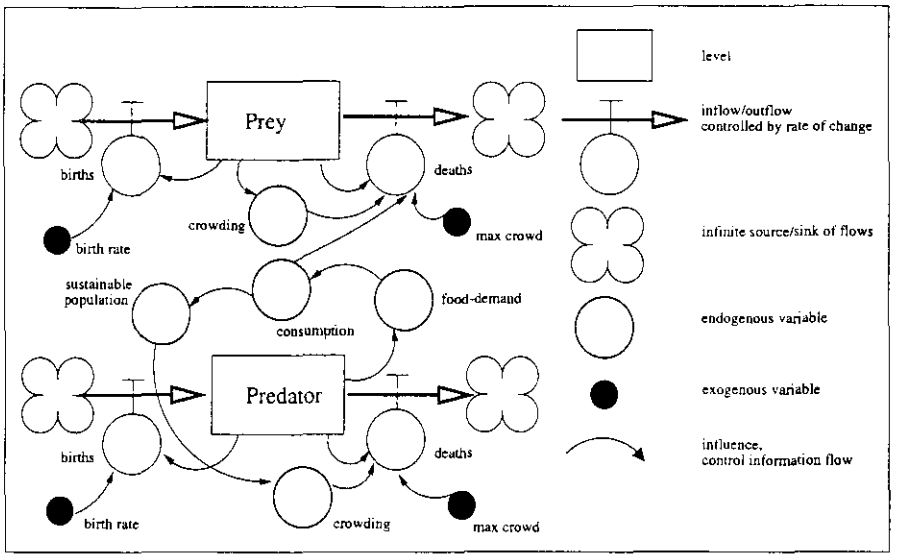

Figure 2.3: System dynamics model of predation

behaviour and the available prey. The total consumed prey by the predator is added to the total outflow of prey. This description is represented in figure 2.3 by variables and influences between them.

2.2.2 Ecological modelling with object-oriented

Object-oriented models consist of objects with complex internal dynamics which interact with each other. The interaction among these objects are determined by rules, in form of methods or procedures, constructed based on the properties of each object. The most essential concept in the object-oriented modelling paradigm is the concept of *class*, which describes both the structure of an object and a set of methods or procedures for initialising and using it (Booch, 1994). *Inheritance* is another important concept based on a hierarchical structure which allows a class defined in the lower level of the hierarchy to inherit all the attributes and methods of the other class or classes above it (Martin and Odell, 1992). Inherited methods can be redefined to suit the structure and purpose of the individual class. Thus, an additional

control for manipulation of local attributes might be possible.

To illustrate the object-oriented approach to ecological modelling, the scenario of section 2.2 is reused.

Based on the given scenario, the representation of individual populations is achieved through the definition of classes, and the interaction between populations is imposed through the definition of methods within each class. The first task is to define the *Population* class which will become the basic building block for the model. Populations can be characterised by attributes such as *Biomass* and *Derivative* (Silvert, 1993). *Biomass* refers to the total quantity or weight of organism in a given area, and it is necessary to include *Derivative* in the class definition to enable objects of this class being used in a model based on first-order differential equations. Thus, it is possible to model the changing of any values over time.

Though each type of population is unique, there exists many similarities between them as all populations are descended from a common ancestor, and differences arise as we move down the family tree. Thus, different classes of population (i.e. *Predator, Prey*) can be treated as descendants of a common ancestor, *Population*, which means they inherit all the attributes and methods defined in *Population*. Those particular features which make them distinct from each other are defined in the class itself (Silvert, 1992). In some cases, a population has different roles, for instance, the population is a predator and at the same time a competitor. This new specialised role as a competitor can be introduced as a new class, therefore, a new set of attributes and methods associated with this competitor class need to be defined, which can be used to establish a relationship with the other defined classes.

In figure 2.4, there are three classes of objects: the basic *Population* class and two derived classes, *Prey* and *Predator*. *Prey* and *Predator* inherit all the attributes assigned to *Population* namely *Biomass* and *Derivative*, as well

as its methods. In each of these subclasses, the *Grow* method is redefined to allow customisation in calculating the *Biomass* growth for each of the individual class, since these two populations are functionally different - for example while the predator is subject to a constant mortality, the prey might have to consider the grazing factor of the predator, in addition to the other natural and endogeneous factors. Furthermore, the *Eat* method is defined in the *Predator* class to represent the predation behaviour of the predator on the prey.

This object-oriented model of predation corresponds to the familiar Lotka-Volterra model on predator-prey system. Depending on the asserting assumptions and given scenario, various aspects of population growth phenomenon, such as the exponential-growth or logistic-growth might be able to be modelled. The conceptual overview of this object-oriented model on ecological predation phenomenon is given in figure 2.4. The detailed implementation, written in the C++ programming language syntax is provided in figure 2.5 and 2.6. Type declarations, class constructors and destructors, and other parts of the code have been omitted for brevity, and all the attributes are treated as *public*, in which they are accessible from outside the class. This detailed implementation follows the work of Silvert (1993) on object-oriented ecological modelling, which was written in the Turbo Pascal programming language.

2.3 Summary

In this chapter, an overview of the compositional modelling technique is given. It covers the processes involved in this technique and a certain design aspects of the domain theory used by the technique. The proposed conversion technique is applied to the ecological modelling domain, therefore, the ecological phenomena that this work will focus are also described. This

chapter also contains a description of the two ecological modelling paradigms intended to be used as a testing platform. They are ecological-modelling with systems dynamic and object-oriented modelling paradigms.

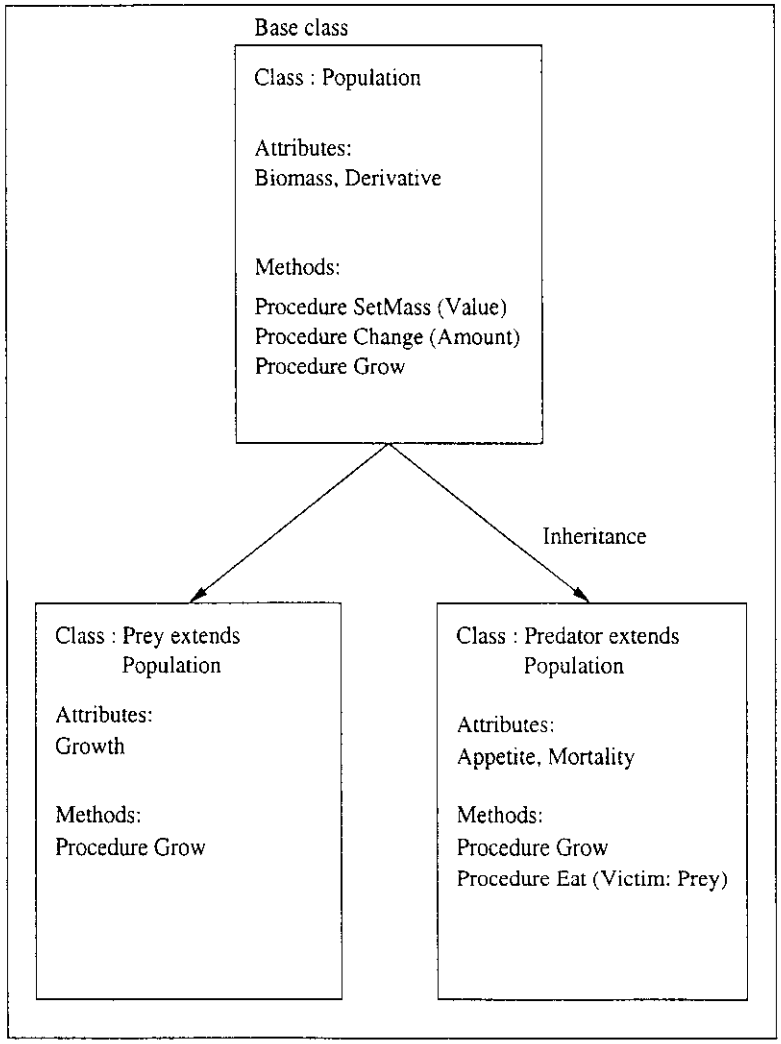

Figure 2.4: Conceptual overview - Object-oriented model of predation

```
class Population {     //this is the base class

Biomass, Derivative;   //attributes
Setmass(NewValue);     //method for setting Biomass
Change (Amount);       //increment the Derivative
Grow ();               //redefined for each class

};

class Prey public Population {  //descendant class

Growth;                         //growth parameter attribute
Grow ();                        //redefined for each class

};

class Predator public Population { //descendant class

Appetite, Mortality;               //attributes
Grow ();                           //redefined for each class
Eat (Prey : Victim)                //predation behaviour

};
```

Figure 2.5: Class definition of an object-oriented model

```
Population::SetMass (NewValue)
{   Biomass = NewValue;
    Derivative = 0.0;
}

Population::Change (Amount)
{   Derivative = Derivative + Amount; }

Population::Grow ()

Prey::Grow ()
{ Change(Growth*Biomass); }

Predator::Grow ()
{ Change(-Mortality*Biomass); }

Predator::Eat (Prey Victim)
{ Victim.Change(-Appetite*Biomass*Victim.Biomass);
  Change(Appetite*Biomass*Victim.Biomass);
}
```

Figure 2.6: Class implementation of an object-oriented model

Chapter 3

Theory

3.1 General overview of conversion technique

Through the proposed conversion technique, models from one modelling paradigm are converted to another, by means of a pair equivalent compositional modelling domain theories. Referring to figure 3.1, the proposed conversion technique can be viewed as an external process, which is only applicable once the model formulation task on *Domain theory I* is completed successfully. The generated output of this completed process is then fed to the proposed conversion technique as an input. From now on, *Domain theory I* is referred to as the source domain theory and *Domain theory II* is referred to as the target domain theory.

Though there exist a number of model formulation frameworks in compositional modelling, for the purpose of this project, compositional modelling algorithm developed by Falkenhainer and Forbus (1991) is adopted within the project. This framework uses explicit *modelling assumptions* to decompose domain knowledge into semi-independent model fragments, each describing various aspects of objects and physical processes. Since one of the important parts of the proposed conversion technique focuses on the manipulation of

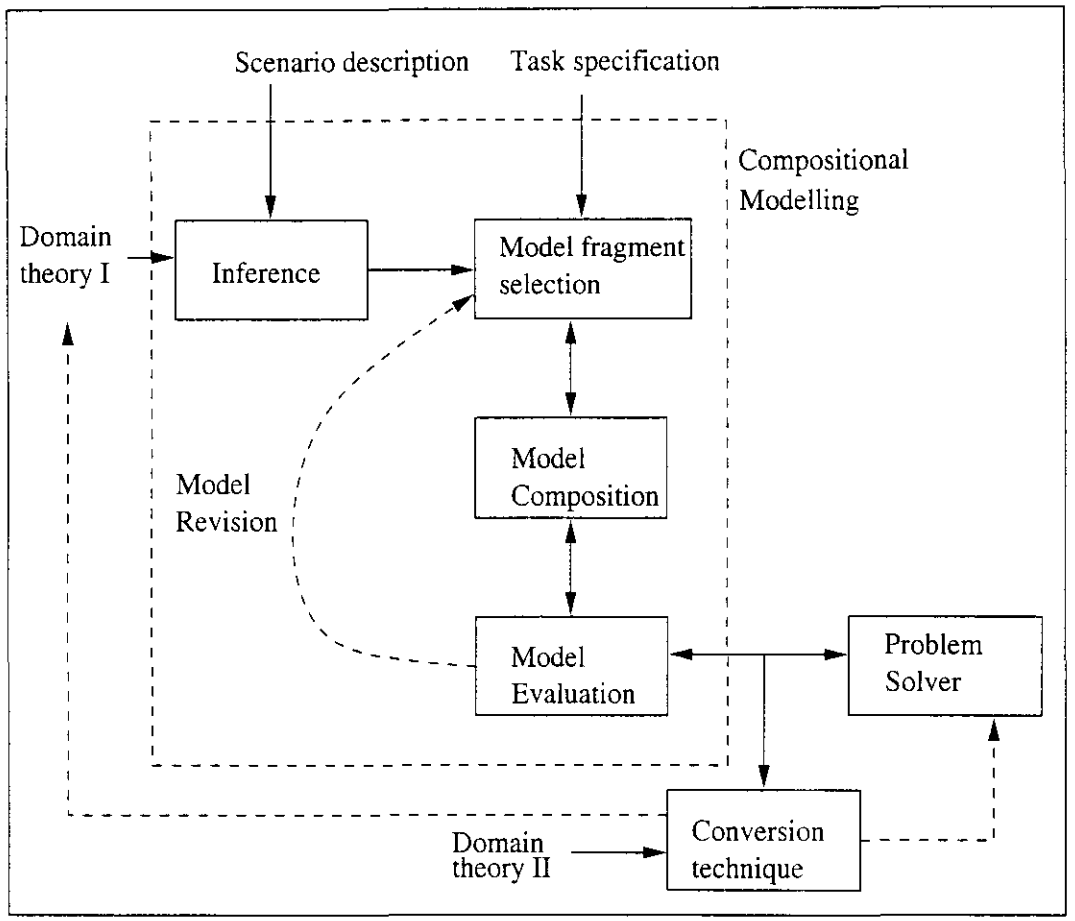

Figure 3.1: The proposed conversion technique, and its relation to the generic architecture of compositional modellers

modelling assumptions, the compositional modelling framework of Falkenhainer and Forbus (1991) naturally complements this requirement.

Therefore, for this project, it is assumed that the compositional modelling process of the source domain theory is performed through this framework. The model formulation task of this framework is to select a suitable set of modelling assumptions. This set of consistent, ground assumption is called the *modelling environment*, from which a sufficient scenario model is derived as the final output. A brief description on the modelling environment and assumption classes are provided in the next two subsections.

Referring to figure 3.1, from the box which represents the conversion process, there is an explicit arrow that points back to the source domain theory. This arrow is used to indicate that a converted compositional model should be able to be translated back to its original scenario model by utilising the same conversion technique. In a way, this suggests that the proposed conversion technique should work in a bidirectional manner. A discussion on this feature is provided in section 3.5.

3.1.1 Modelling environment

In Falkenhainer and Forbus (1991), a detailed description on modelling assumptions, which constitute the modelling environment, is given. These assumptions, usually embedded within the model fragments, provide control over the fragments' instantiation and use, so that only the relevant aspects of a situation are examined. With reference to the model fragment definition in section 2.1.2, assumptions are used to represent specific features that are included in the associated model fragments. For example, assumptions may indicate the inclusion of certain phenomena or distinguish between alternative ways of modelling these phenomena.

In their work, explicit simplifying assumptions to state each model's underlying commitments (e.g. abstraction level, approximations, perspective, and granularity) and the conditions under which they are appropriate are introduced. The problem solver's decision about how to model a particular scenario is largely driven by this type of assumption.

Simplifying assumptions are required to take the form of:

$$\text{CONSIDER}(\text{AsyType}(\text{system}))$$

where Asytype is a predicate denoting the specific kind of assumption, while system is the subject of the assumption. Therefore,

$$\text{CONSIDER}(\text{relevant}(\text{growth}(\text{Penguin} : \text{population})))$$

indicates that the subject of the assumption is the growth phenomenon of an entity of type population (e.g. Penguin), which needs to be considered in the model formulation analysis.

The described assumptions are essential in the conversion process as they provide the necessary information of the scenario descriptions and modelling decisions. Therefore, they can be utilised in the selection process of model fragments, given the target domain theory.

3.1.2 Assumption classes

The possible modelling assumptions for a domain can be described as some collections of assumptions represent mutually exclusive, alternative ways to model the same aspect of an object or phenomenon. In order to represent this important relationship, some assumptions are organised into sets called *assumption classes*, which is crucial in this work. Assumption classes, which are part of a domain theory, represent natural groupings that can be reasoned about as a whole, such as the alternative ways to model the same aspect of an object phenomenon. An assumption class captures one dimension along which a modelling choice must be made. For example, the growth phenomenon of a population can be modelled as either exponential or logistic in terms of the growth rate. Not all dimensions are relevant in all contexts. Therefore, a condition is required to state when an assumption class is relevant. For instance, the Holling-Tanner model of predation will only be relevant if the logistic population growth rate is considered for the populations involved in the predation relationship.

Assumption classes are declared with the form:

$$(\text{defAssumptionClass } c(a_1, ..., a_n))$$

where condition c is an atomic sentence containing (possibly empty) set of free variables v and $(a_1, ...a_n)$ is a mutually exclusive set of atomic sentences

(e.g. CONSIDER statement) whose free variables, if any, must be from v (Falkenhainer and Forbus, 1991). It is logically equivalent to:

$$\forall v\ c \longrightarrow a_1 \vee ... \vee a_n, \forall a_i, a_j, i \neq j, \neg a_i \vee \neg a_j$$

An assumption class is considered *active* when c holds. Any scenario model must include exactly one assumption from each active assumption class. Inactive assumption classes are ignored, and none of their constituent assumptions are included. Intuitively, when the class condition holds, one and only one of the assumptions associated with that class must hold in the scenario model.

3.2 Technique overview

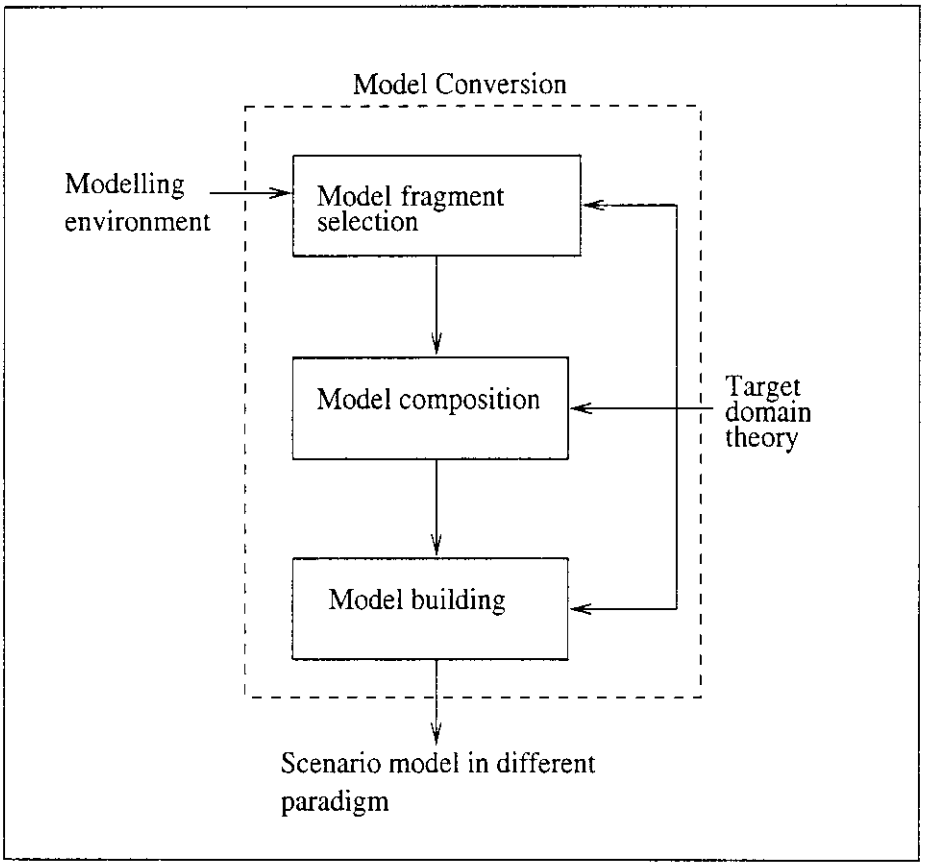

Figure 3.2: Overview of processes involved in the proposed conversion technique

Figure 3.2 presents a generic architecture of the proposed conversion technique. Initially, the modelling environment, which is produced as part of the output of the composition process on the source domain theory, becomes an essential input fed to the conversion process.

Given a set of modelling assumptions, all the relevant model fragments of the target domain theory are selected in the *model fragment selection* stage. However, it is not necessarily a one-to-one mapping between fragments of the two domain theories. It might be the case that a model fragment in one domain theory covers parts of several model fragments in the other domain

theory.

Though the provided modelling environment is adequate and consistent with respect to the source domain theory, the assumptions derived from the selected model fragments of the target domain theory need to be validated with respect to their adequacy and consistency. Therefore, in the *model composition* stage, adequacy and consistency checking are performed.

As the target domain theory is designed based on a different modelling paradigm, it is expected that certain choices of a set of assumptions might raise new choices in turn. Therefore, in adequacy checking, an adequate set of assumptions, capable of generating the intended scenario model from the target domain theory is searched . Consistency checking is then performed to determine whether the underlying assumptions of this selected set are compatible with one another. Deletion or substitution of assumptions of this set may be required to repair consistency. Once an adequate and consistent set of modelling environment of the target domain theory is obtained, this set is passed to the *model building* process. This set is then utilised to build a sufficient scenario model. Throughout the conversion process, the target domain theory is accessible by each stage of this process.

3.3 Model fragment selection

In the model fragment selection stage, the relevant model fragments, intended to be utilised to build up the scenario model are chosen, by means of the provided modelling environment of the source domain theory. Assuming that both domain theories rely on the same syntax and naming convention in specifying their underlying assumptions, each assumption in the modelling environment set is pattern-matched with the assumption construct of the defined model fragments in the target domain theory. The pattern matching process is focussed specifically on the type and subject of each assumption.

For each such collection where the assumption type and subject of the model fragments matched the provided modelling environment, it can be said that the model fragment is *applied* to the scenario to be built. The model fragment is selected and the corresponding assumption object is then used to instantiate the assumption and the source participants construct of this selected fragment. The whole matching process terminates once all the assumptions in the given modelling environment have been evaluated. The formal notation of this whole process can be described as follows:

Given a modelling environment, $E = \{a_1, ..., a_n\}$, a model fragment definition μ is a tuple $< P^s, P^t, C^s, C^o, C^t, A >$ with A(μ)=$\{a_s, ..., a_t\}$, and a database construct $\triangle = empty$. Then,

FOR each $a_i \in E, i = 1..n$ in the modelling environment AND each μ in the target domain theory

If A(μ) MATCHED any set of $a_i \in E$ AND μ is NOT SELECTED yet THEN

- SELECT μ and USE object of $a_i \in E$ to INSTANTIATE corresponding A(μ)
- ADD to $\triangle$

Once the whole process is completed, the database construct, containing the relevant fragments of the target domain theory, is passed to the next process.

3.4 Model composition

Having derived a model space, which consists of related model fragments intended to model a certain scenario, composing individual models from these model fragments is possible. The task of composing a model from these selected model fragments becomes a trivial process if there exists a one-to-one mapping between corresponding model fragments of the two domain theories. However, this may not be the case. Therefore, the task of model composition becomes more complicated, especially in searching for a sufficient and consistent set of model fragments from which the intended scenario model can be deduced.

One approach would be to explicitly reason about combinations of model fragments to ascertain which set are consistent and sufficient. However, there can be many combinations, involving many irrelevant model fragments. A better alternative is to reason about combinations of modelling assumptions, as each model fragment is conditioned on a set of modelling assumptions stating their range of applicability and underlying approximations.

Therefore, reasoning will focus on choosing among the set of possible modelling assumptions, which enable a corresponding set of model fragments, rather than reasoning about each model fragment individually. The task of selecting consistent and sufficient modelling assumption then can be cast as a Dynamic Constraint Satisfaction problem (DCSP).

3.4.1 Dynamic Constraint Satisfaction problem (DCSP)

A constraint satisfaction problem (CSP) is typically defined as the problem of finding consistent assignment of values to a fixed set of attributes given some constraints over these attributes.

The simplest of constraint satisfaction problem (CSP) can be generally specified as triplet $< X, D, C >$ where X is a set of attributes $\{X_1, ..., X_n\}$,

D is a set of domains $\{D_1, ..., D_n\}$ describing the potential values of the attributes and C is a set of constraints relating some of the attributes. Each attribute $x_i \in X$ must be assigned a single value $d \in D_i$, and such attribute assignment will be denoted as $x_i : d$. Each $c(x_i, ..., x_j) \in C$ specifies a subset of D^c of D_i x...x D_j such that $\forall(d_i, ..., d_j) \in D^c$, $C(d_i, .., d_j)$ is consistent with C. The purpose of solving a CSP is to find a tuple $(d_i, ..., d_n)$ such that the attribute assignment $x_1 : d_1, ..., x_n : d_n$ causes all constraints in C to hold.

However, for many synthesis tasks such as configuration and model composition, the set of attributes that are part of the CSP changes dynamically in response to decisions made during the course of problem solving (Mittal and Falkenhainer, 1990). Therefore, as CSPs are not sufficiently equipped to cater for features that can be dynamic in nature, which are frequently present in real-world problems, dynamic CSP (DCSP) is required. DCSP uses two types of constraints, namely *activity constraints* and *compatibility constraints.* Compatibility constraints correspond to those traditionally found in CSPs, the constraints over the values of an attribute. Activity constraints, on the other hand, describe conditions under which an attribute may or may not be actively considered as part of the final solution. These constrain an attribute to be active or not active based on the other attributes' activity and value assignment.

Similar to the work presented in Mittal and Falkenhainer (1990) and Keppens and Shen (2000), this task requires the handling of DCSPs in which the set of relevant attributes is defined by other attributes. In order to solve such CSPs, active predicates are introduced such that:

$$\forall x_i \in X : active(x_i) \leftrightarrow \forall d \in D_i x_i : d$$

From this it follows that $\neg active(x_i)$ implies that no attribute assignment is considered for x_i. The compatibility constraint $c(x_i, ..., x_j)$ is translated as $c(x_i, ..., x_j) \vee \neg active(x_i) \vee ... \vee \neg active(x_j)$. As a result, the determination

of the truth of an activity predicate implicitly results in a set of constraints as well. Activity constraints come in the form of implications where the consequent consists of literal containing the activity predicate of one of the attributes. Mittal and Falkenhainer (1990), Miguel and Shen (1999) and Verfaillie and Schiex (1994) provide more elaborate description on DCSP, from which the model composition stage of this work is based on.

3.4.2 DCSP for model composition

In the *model composition stage*, the DCSP framework is used to express and solve the problem of searching a set of adequate and consistent modelling assumptions. Therefore, in this subsection, an overview on how the problem is defined into a DCSP is given. The next subsection will then show how simple repair and choose techniques can be added to such DCSP and explain how these aid in guiding the search for a consistent model.

Each assumption class of the domain theory can be used to represent a DCSP attribute, and its domain is the set of assumptions defined within that class. For instance, as illustrated in table 3.1, each assumption class in this table, which corresponds to a certain aspect of a population ecological phenomena, can be used to represent a DCSP attribute. The corresponding assumptions within a particular assumption class indicate a set of possible domain values, which might be assigned to a selected DCSP attribute.

Assumption class	Domain
Growth-relevance	{growth-phenomenon}
Growth-model	{exponential, logistic}
Predation-model	{Lotka-Volterra, Holling-Tanner}

Table 3.1: Table of assumption classes and its corresponding domain

A set of minimal required modelling assumptions, derived from the model

fragments selected via the model fragment selection stage, can be used to identify a DCSP's initial attribute. In this work, the dynamic constraint satisfaction task is to extend this initial attribute set to include any additional assumptions which might be required, in order to build the intended scenario model, given the modelling paradigm construct of the target domain theory. Activity and compatibility constraints, defined around the assumption classes of this modelling paradigm, allow an adequate and consistent set of modelling environment to be obtained at the end of the process. Activity and compatibility constraints, which are applied to the attributes defined in table 3.1, are illustrated in table 3.2 and 3.3.

Activity constraints
Predation-model $\overrightarrow{requires}$ Growth-model
Growth-model $\overrightarrow{requires}$ Growth-relevance

Table 3.2: Table of activity constraints

Compatibility constraints
Predation-model=Holling-Tanner $\longrightarrow$ Growth-model $\neq$ exponential

Table 3.3: Table of compatibility constraints

3.4.3 Simple choose and repair techniques

Activity constraints define the conditions under which attributes are active or not active. Given an initial set of attributes, obtained from the model fragments which have been selected via the model fragment selection stage, any new attribute which becomes active, due to the application of the activity constraints on this initial set of attributes, is then added to this set.

Therefore, the initial set now consists of new attributes, which might be assigned or unassigned to a set of corresponding domain values.

A simple *choose* technique, integrated into the DCSP, then repeatedly selects an attribute from the set that is unassigned and assigns a possible value to it. If the assignments cause some constraint violations, an attempt to repair the current set of assignments to resolve the inconsistencies is performed. The *repair* technique works by unassigning the subset of the assigned attributes that is perceived to have caused the inconsistency, and attempting to reassign values to them such that all constraints are satisfied. This assignment involves replacing the identified assumption value which causes inconsistency with another value from the same assumption class. However, the application of the repairs has to be limited to the newly added attributes. Any attempt to change the assignment of the initial attributes might alter the modelling descriptions and decisions obtained from the source domain theory.

For example, consider the task of determining an appropriate set of modelling assumptions for analysing the growth phenomenon of two populations involved in a prey-predator relationship, based on the DCSP definitions provided in table 3.1, 3.2 and 3.3.

In this example, it is assumed that the initial attribute set, consists of V_I = { Predation-model=Holling-Tanner, Growth-relevance=growth-phenomenon} is given to the process. For simplicity, an instantiation of each population is omitted. It is also assumed that the predation phenomenon only involves a single prey and predator populations.

Given the activity constraints, the initial set is extended to include the Growth-model attribute. After this extension, V_I = {Predation-model=Holling-Tanner, Growth-relevance=growth-phenomenon, Growth-model} is obtained. An assignment of exponential value to the Growth-model attribute by the choose technique, causes a compatibility constraint violation. The simple

repair technique then performs a repair on this set of assignment to resolve the inconsistency by replacing the value assigned to the Growth-model attribute with another value from the same assumption class (i.e. logistic). Therefore, a set of consistent assignments V_I = {Predation-model=Holling-Tanner, Growth-relevance=growth-phenomenon, Growth-model=logistic} is obtained, and a coherent and parsimonious scenario model is derivable from this modelling environment.

3.5 Model building

In the model building process, the consequents of the model fragments, which are conditioned on the modelling environment generated by the model composition phase, are organised in a scenario model form, tailored to the modelling paradigm framework of the target domain theory. For instance, given a target domain theory, designed based on the object-oriented modelling paradigm, a model fragment of this domain theory, which is conditioned on the *Growth-relevance=growth-phenomenon* assumption may assert the existence of a class object, and all the relevant attributes and methods to represent the population. Another model fragment which is conditioned on the *Growth-model=logistic* assumption may assert the existence of the logistic growth attributes, defined within the asserted population class construct. The corresponding methods to manipulate these attributes are also asserted. Eventually, a complete scenario model might be obtained by combining and organising these assertions in a specified form, based on the object-oriented modelling paradigm.

Therefore, in this work, the model building process can be viewed as an "organise and display" mechanism, which allows the different consequents of the selected model fragments to be combined into a solid scenario model form that corresponds to the modelling paradigm of the target domain theory.

3.6 Bidirectional conversion

In this work, the focus is on the unidirectional conversion process, in which the main task is to convert the compositional model of the source domain theory to the target domain theory. Nonetheless, in reality, the proposed conversion technique should work in both directions. Once converted, the scenario model produced within the target domain theory should be able to be translated back to its original source domain theory paradigm by utilising the same conversion technique. This whole process, which can be termed as the bidirectional conversion process, can be viewed as a partial cycle and the illustration is given in figure 3.3. In the figure, the model building process is omitted, since it does not significantly influence the whole conversion process.

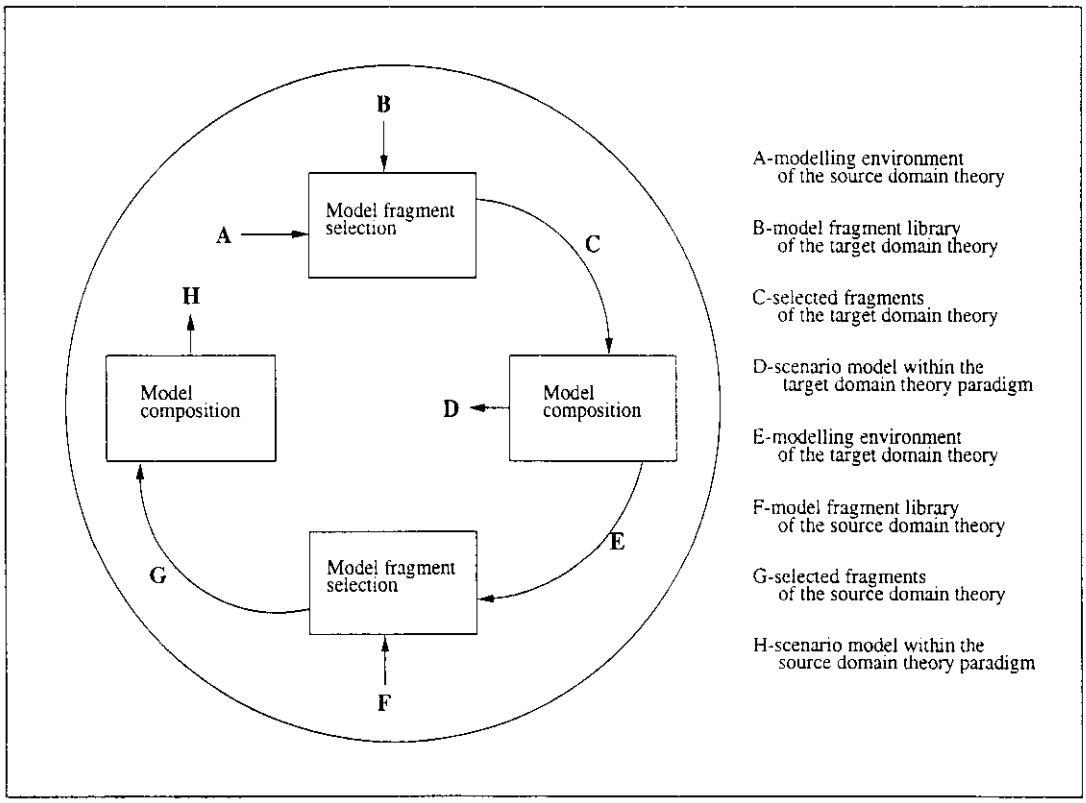

Figure 3.3: The bidirectional conversion process

Initially, a set of relevant model fragments which corresponds to the given

modelling environment of the source domain theory is searched within the model fragments library of the target domain. The selected model fragments are then passed to the model composition phase, in which an adequate and consistent set of modelling environment, to enable the model building process within this new modelling paradigm is further searched. Therefore, assuming that the model composition task on this new modelling paradigm is completed successfully, the generated modelling environment of this process could be fed back to the conversion technique in order to derive the scenario model of the initial modelling paradigm. For this task, another cycle of model fragment selection and model composition processes are applied, as illustrated in figure 3.3.

However, within the limited time, it is not possible to implement this feature.

3.7 Summary

In this chapter, a description on the theoretical design of the proposed conversion technique, with the stages involved in the conversion process, is given. The conversion process basically consists of three subprocesses, namely *model fragment selection*, *model composition* and *model building*. In the model fragment selection stage, the corresponding model fragments of the target domain theory are selected by means of the provided modelling environment of the source domain theory. The adequacy and consistency of the assumptions of these selected and instantiated model fragments are then determined in the *model composition* stage. In this stage, the DCSP framework is used to express and solve the mentioned task. A simple choose and repair techniques are then integrated into the DCSP, which allows the reassignment of an attribute value that causes inconsistency with a new value from the same assumption class. In the model building stage, the different consequents of the

selected model fragments are combined into a solid form of a scenario model, which corresponds to the modelling paradigm of the target domain theory. Though a major part of this work is focused on a unidirectional conversion process, in the real system, the conversion should work in both directions. Therefore, this chapter also provides a discussion on the theoretical aspect of a bidirectional conversion process.

Chapter 4

Design and Implementation

4.1 Knowledge representation

The domain theories used in this work largely consists of a model fragment library, which is a collection of predefined model fragments. These model fragments are designed based on the definition described in section 2.1.2.

The representation of model fragments in this work conceptually follows the general framework of the Compositional Modelling Language (CML) (Bobrow et al., 1996). However, since the C++ programming language is used in designing and implementing the fragments constructs, certain syntactical difference is necessary in order to accommodate the chosen language of implementation.

In Keppens and Shen (2000), a knowledge representation framework is devised that supports ecological modelling within the system dynamics modelling paradigm. For this project, this devised framework is used as a source domain theory, from which the modelling environment is obtained.

For the target domain theory, a knowledge representation framework which supports an object-oriented modelling paradigm is required. Since there is no existing domain theory of compositional ecological modelling

which utilises this particular framework, the task of designing and implementing a domain theory in this paradigm is required. In the next subsection, an overview of Keppens and Shen (2000) work, especially on the assumptions design is given. Following this subsection, a detail overview on the design and implementation aspect of the target domain theory is illustrated.

4.1.1 Assumptions design of system dynamics paradigm

A large part of this model conversion work focuses on assumptions manipulation, where a set of adequate and consistent assumptions to enable the building of a scenario model is searched. Therefore, assumptions design become an important aspect of the knowledge representation. Keppens and Shen (2000) introduce two types of assumptions in their system dynamics framework. The first type of assumption is *relevance assumption*, denoted as relevant($h, p_1, ..., p_q$), states that the associated model fragment describes a phenomenon h, which applies to the participants $p_i, ..., p_j$. Consider, for example the model fragment:

$$\text{population}(p) \wedge \text{relevant}(\text{growth}(p)) \longrightarrow \text{level}(l) \wedge \text{unit-of}(l,\text{population}) \wedge \text{rate}(r) \wedge \text{size-of}(p,l) \wedge \text{flow}(r,\text{source-sink},l).$$

This fragment introduces all objects that are required to represent the phenomenon growth(p): the growth rate r and the population level l, which represents the size of p, and the relation between them, flow(r,source-sink,l). Another type of assumption is *model assumption*, denoted as model(Θ, t), states that the associated model fragment represents the source participants or structural condition Θ in a specific way described by t. Such assumptions are used to distinguish between different ways of describing (or explaining) objects constants or relations between object constants. Consider, for example the model fragment:

$$\text{rate}(r) \wedge \text{level}(l) \wedge \text{unit-of}(l,\text{population}) \wedge \text{flow}(r,\text{source-sink},l) \wedge$$
$$\text{model}(r,\text{exponential}) \longrightarrow \text{birth-rate}(r_0) \wedge (r = r_0 * l).$$

This fragment contains the "exponential" model type for the number of births per time unit r of a population (being $r = r_0 * l$).

Assuming that the model formulation task on this domain theory is completed successfully, a set of adequate and consistent modelling environment, in the form of these two modelling assumptions, is obtained. This modelling environment is then fed to the proposed conversion technique.

4.1.2 Design and implementation of object-oriented paradigm

For this work, a knowledge representation framework that supports an object-oriented modelling paradigm needs to be designed and implemented. Since there is no existing compositional domain theory of ecological phenomena that supports this devised paradigm, Silvert (1992) works on the object-oriented modelling of population ecology are translated into compositional modelling model fragments.

As described in section 2.2.2, object-oriented models are basically characterised by objects known as classes, attributes and methods. Therefore, in the design of the model fragments, these objects become the necessary participants of the fragments. In addition, these fragments are designed based on the ecological phenomena described in section 2.2. Consider an example of a model fragment which utilises an object-oriented modelling paradigm as the following:

$$\text{population(p)} \wedge \text{relevant(growth(p))} \longrightarrow \text{class}(a) \wedge \text{class-of}(a,\text{p}) \wedge$$
$$\text{biomass}(b) \wedge \text{derivative}(d) \wedge \text{attribute-of}(b,a) \wedge \text{attribute-of}(d,a) \wedge$$
$$\text{setmass}(s) \wedge \text{change}(c) \wedge \text{method-of}(s,a) \wedge \text{method-of}(c,a)$$

This fragment introduces all objects that are required to represent the phenomenon growth(p). The class a of type population, two attributes that

characterised the class; biomass b and derivative d, two methods; setmass s and change c to manipulate these attributes, and also the relations between them. The implementations to manipulate the class and attribute participants are integrated within the defined methods.

Syntactically, the assumptions of these two domain theories have to be similar, therefore, the assumptions of the target domain theory are designed based on the notations specified in the source domain theory (i.e. *relevant* and *model*). Since the ecological phenomena covered by this work is quite limited, it is possible to design a one-to-one mapping between corresponding fragments of these two domain theories, however, such attempt definitely defeats the purpose of developing this conversion technique and does not portray the actual situation of model fragments design, involving two different modelling paradigms.

Besides the phenomena described in section 2.2, the scope of the target domain theory is purposely extended to include a fragment which explicitly introduce an intrinsic natural rate of increase for a population. Consider, for example, the following fragment:

population(p) $\wedge$ class(a) $\wedge$ class-of(a,p) $\wedge$ intrinsic-natural-rate-of-increase(r) $\wedge$ attribute-of(r,a) $\wedge$ model(r,closed) $\longrightarrow$ birth-rate(r_o) $\wedge$ attribute-of(r_o,a) $\wedge$ generation-time(T) $\wedge$ attribute-of(T,a) $\wedge$ ($\frac{r=log(r_o)}{T}$)

This fragment contains the "closed" model type of assumption for the population intrinsic natural rate of increase (being $\frac{r=log(r_o)}{T}$). In this type of population, no immigration or emigration occurs. The other alternative would be the "open" population (Mackenzie et al., 1998).

The implementation for each fragment is defined through the structure called a *struct* in C++, which forms a shell around one or more values of the same or different types (Swan, 1999). Each specific part of the fragment then

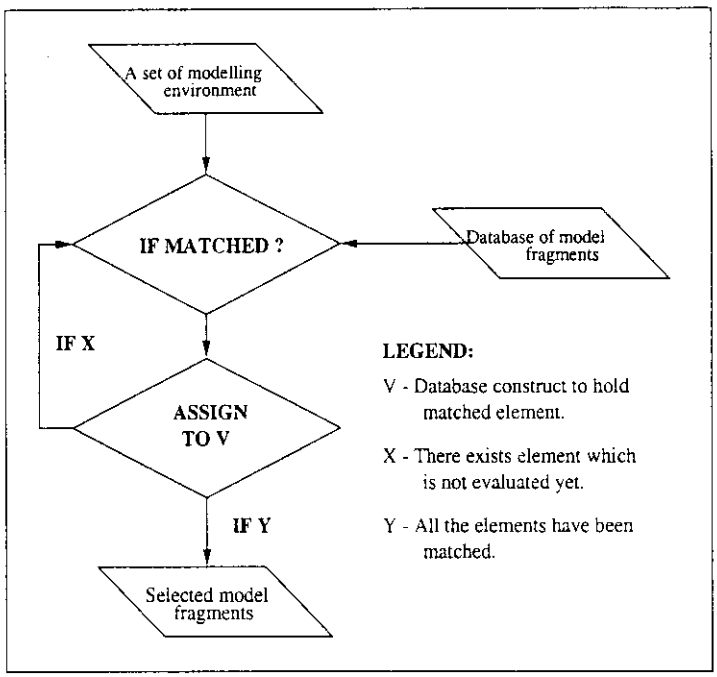

Figure 4.1: Data flow diagram of the model fragment selection process

becomes the element of this structure. However, for ease of manipulation and to easily manage each individual fragment, assumptions are the only element explicitly defined within each fragment. The elements other than assumption are defined in a separate database and a unique numerical value is used to identify them. Each unique numerical value reflects the participant's association with a particular model fragment.

4.2 Model fragment selection

The theoretical design of the model fragment selection process described in section 3.3 is implemented. For this process, the flow of data is illustrated in figure 4.1.

The input to this matching process consists of a set of modelling environment, $E = a_1, ..., a_n$ and a database of model fragments, $\mu_1, ..., \mu_k$ of the target domain theory. The modelling environment input is in the form spec-

ified in section 4.1.1. Therefore, in the actual execution of the system, a text file consisting the assumptions which correspond to $E = a_1, ..., a_n$ is provided. The assumption construct of each μ_i, $i = 1, .., k$ is matched with each $a_i \in E$ of the modelling environment. Any μ_i which matched with a_i, in terms of its type and subject, is selected and assigned to a vector, V. Providing that this μ_i is not selected yet, or the instantiated object of a_i is not similar to the instantiated object of any existing $v_i \in V$, μ will be added to a vector of selected model fragments. The eventual output of this process is a vector V, of model fragments, in which each $v_i \in V$ is relevant to the scenario and modelling decision described in the source domain theory. The algorithm of this process is provided in figure 4.2.

The assumptions are represented as string of characters. Therefore, the C++ programming language predefined standard library functions contained in the header files *string.h* and *string* apply naturally to the matching process. This is one of the important features which makes C++ a preferable choice over the others to be used in implementing the proposed technique. In addition, the author's competency in this language is another important aspect which has influenced this decision.

Swan (1999), Sellappan (1994) and Lippman and Lajoie (1998) provide more detailed description on these built-in C++ string functions and other C++ features, which are largely utilised in this work.

```
Procedure fragment_selection (E,M)
BEGIN
   M <--- Database of model fragments
   E <--- Modelling environment
   V <--- Empty vector construct
   N <--- The number of element in M
   K <--- The number of element in E

   FOR i=1 to N
     FOR j=1 to K

      IF Assumption construct of i_th element of M
         MATCH the j_th element of E AND

         i)  (i_th element of M is NOT assigned yet to V)

         OR

         ii) (the instantiated object of the j_th element of E
              DO NOT MATCH
              the instantiated object of any existing element
              of V)

      THEN

         ADD i_th element of M to V

      END IF

     END FOR
   END FOR

   RETURN V

END
```

Figure 4.2: Algorithm of the model fragment selection process

Assumption class	Domain
Growth-relevance	{growth-phenomenon}
Predation-relevance	{predation-phenomenon}
Growth-model	{logistic, exponential}
Predation-model	{Lotka-Volterra, Holling-Tanner}
Intrinsic-rate-of-increase-model	{open-population,close-population}

Table 4.1: Table of assumption classes and its corresponding domain

Activity constraints
1. Predation-relevance $\overrightarrow{requires}$ Growth-model
2. Growth-model $\overrightarrow{requires}$ Growth-relevance
3. Predation-model $\overrightarrow{requires}$ Predation-relevance
4. Growth-model $\overrightarrow{requires}$ Intrinsic-rate-of-increase-model=close-population

Table 4.2: Table of activity constraints

4.3 Model composition

The theoretical design of the model composition process described in section 3.4 is implemented. In the implementation stage, assumption classes of the target domain theory need to be defined in a DCSP framework. In table 4.1, each assumption class of the target domain theory and its corresponding domain values are defined. Each assumption class represents a DCSP attribute. The activity and compatibility constraints that correspond to these attributes are defined in table 4.1 and 4.2 respectively.

The model composition processes consists of two major subprocesses. The first subprocess consists of a main propagate cycle in which the constraints relevant to the initial problem statement are checked, and the consequences and dependencies due to this constraints are propagated. A similar process

Compatibility constraints
5. Predation-model=Holling-Tanner $\longrightarrow$ Growth-model $\neq$ exponential
6. Predation-model=Lotka-Volterra $\longrightarrow$ Growth-model $\neq$ logistic

Table 4.3: Table of compatibility constraints

is applied to all derivable active attributes, until all relevant attributes are identified.

Once this is complete, the second subprocess is applied. This second subprocess consists of the integration of *choose, propagate and repair* steps. Each choose step selects an active unassigned attribute and assigns it a value that has not been ruled out. Each propagate step then checks the constraints relevant to the new attribute value assignment and propagates their consequences and dependencies. Constraint checking is ordered to take advantage of the differing scope of each constraint type. Activity constraints are checked first since they apply to attribute activity, encompassing all their possible value assignments. Then, compatibility constraints are examined to see if the new attribute assignment is consistent. If an inconsistency occurs, a repair procedure is invoked that will unassign this value assignment and replace it with a value that has not been ruled out. The algorithm of these two subprocesses are provided in figure 4.3 and figure 4.4.

Once an adequate and consistent set of modelling environment is obtained, this set is written into a text output file. This file becomes an essential input fed to the next stage of the conversion technique, which is the model building process.

```
Procedure first_subprocess_model_composition (Vi)
BEGIN

  Vi          <--- A set of initial attributes
  V           <--- Initial state - empty
  Conflict?   <--- false
  error?      <--- false
  Ci          <--- a particular constraint from the set of defined
                   constraints in the problem

  Check all applicable activity and compatibility constraints on Vi
     IF Conflict? == true
        THEN return fail (initial problem statement is inconsistent)
     V <-- Vi (Assigning element of Vi to V)

  FOR each attribute vl of V

     IF there is an active Require constraint Ci applicable to vl
        run Ci, add new active attribute to V

     ELSE IF there is an active Require-Not constraint Ci applicable
     to vl then
        run Ci, rule out new active attribute


     END IF

   END FOR

   RETURN V

END
```

Figure 4.3: Algorithm of the first subprocess of the model composition stage

```
Procedure second_subprocess_model_composition(V)
BEGIN
Conflict? <--- false
error?    <--- false

FOR each attribute vl of V not yet assigned a value

value(vl) <--- Choose(vl) (choose an assignment for vl)

Check all applicable compatibility constraints
on assignment

 IF Conflict? == true
  rule out assignment
     DO
    repair(value(vl)) <--- (new value assigned to vl)
    check all applicable compatibility
    constraints on new assignment

     UNTIL all constraints are satisfied or error? == true
 END IF

END FOR

RETURN V

END
```

Figure 4.4: Algorithm of the second subprocess of the model composition stage

4.3.1 Example trace

An example partial trace of the model composition algorithm, described in figure 4.3 and 4.4, is provided in table 4.4. This partial trace is based on the DCSP framework of the target domain theory described in table 4.1, 4.2 and 4.3. In this trace example, assume that an initial attribute set provided to the algorithm is V_I = {Predation-model=Lotka-Volterra}. For simplicity, the names of all attributes involved are shortened as shown in the following legend:

Pr - Predation-relevance
Gr - Growth-relevance
Pm - Predation-model
Gm - Growth-model
Rm - Intrinsic-rate-of-increase

In the figure, the active attributes and value assignments are shown in bold font, constraint propagation and value choices in italics, and explanatory comments in roman. A constraint propagation is represented as C_i, in which i is a numeric value referring to an individual constraint out of the six constraints (i.e. 1-6) that have been defined in table 4.2 and 4.3.

Tracing example
V_i = **{Pm=Lotka-Volterra}**
C3 runs (requires make Pr active)
V = **{Pm=Lotka-Volterra, Pr}**
C1 runs (requires make Gm active)
V = **{Pm=Lotka-Volterra, Pr, Gm}**
C2 runs (requires make Gr active)
V = **{Pm=Lotka-Volterra, Pr, Gm, Gr}**
C4 runs (requires make Rm=close-population active)
V = **{Pm=Lotka-Volterra, Pr, Gm, Gr, Rm=close-population}**
choose Pr=predation-phenomenon
choose Gm=logistic
C6 runs (leading to a conflict, perform repair on Gm)
choose Gm=exponential
choose Gr=growth-relevance
V = **{Pm=Lotka-Volterra, Pr=predation-phenomenon, Gm=exponential Gr=growth-relevance, Rm=close-population}**

Table 4.4: Tracing example

4.4 Model building

The theoretical design of the model building process discussed in section 3.5 is implemented and the process is centered on the task of transforming the adequate and consistent set of modelling environment, generated by the model composition phase, into an acceptable scenario model form, tailored to the modelling paradigm of the target domain theory. To accomplish this task, the consequents of each model fragment that is conditioned on this set of modelling environment need to be selected and organised into an acceptable predefined form.

As described in section 4.1.2, the participants of each model fragment are defined in a separate database, and a unique numerical value is used to associate each particular model fragment to the corresponding participants. Therefore, in this model building process, the input to the process consists of a database of model fragments participants, $R_1, ..., R_k$, and a set of modelling environment, $E = \{a_1, ..., a_n\}$, which is obtained from the model composition process.

For each $a_i \in E$ which matched with $R_j, j = 1..k$, in terms of R_j's association with a particular model fragment, it is said that the model fragment to which this participant belongs to, is conditioned on a_i. Therefore, R_j is selected and added to a vector, V_p. This process is repeated until all of $a_i \in E$ has been evaluated. At the end of this process, a vector V_p, which consists of all the consequents of the model fragments which are conditioned on the generated modelling environment is obtained. In order to display the content of V_p, the display procedure which is integrated within this process has to consider the modelling paradigm of the target domain theory (i.e. object-oriented modelling paradigm). The defined constructs of the modelling paradigm (i.e. class, attributes and methods among others) need to be considered, as different constructs have a different hierarchy of precedence.

For instance, it is not possible to define a particular attribute or method which is associated with a class, unless this class has been defined first. In fact, within the class itself, there exists a category of base and derived classes.

The algorithm of this model building process is provided in figure 4.5.

4.5 Actual implementation

In the actual implementation, the model fragment selection and the model composition processes are combined as one application, and the model building process is set up as a separate application, which stands on its own. However, in order to execute the latter application, it requires a text file of modelling environment produced by the former application. The decision to split these three processes into two separate C++ programs is due to the following reasons:

- Unlike the other two subprocesses, the influence asserted by the model building process on the model conversion task of this project can be considered insignificant and very minimal. Its core objective is mainly geared towards organising and displaying the consequents of the selected model fragments into a scenario model form. Therefore, based on this, it is appropriate to establish the model building process as a separate application entity.

- The volume of programming codes has to be maintained at a manageable and reasonable level. The splitting allows this aim to be achieved.

4.6 Summary

This chapter describes how the theoretical design of the proposed conversion technique is implemented. It includes a description on the knowledge

representation aspect of the domain theories used for the testing, with high focus on how the model fragments of the target domain theory are designed and implemented. Algorithms of the three processes involved in this proposed conversion technique; *model fragment selection*, *model composition* and *model building*, are also provided. An example of a simple tracing of attributes and their domain values which are involved in the *model composition* process is also given. This chapter ends with a brief description on the actual implementation of the proposed conversion technique.

```
Procedure model_building (E,R)
BEGIN
   R  <--- Database of model fragments participants
   E  <--- Modelling environment
   Vp <--- Empty vector construct
   N  <--- The number of element in R
   K  <--- The number of element in E
   FOR i=1 to K
     FOR j=1 to N

      IF i_th element of E
         MATCH the unique numeric value of j_th element of R
      THEN

         ADD j_th element of R to Vp
      END IF

     END FOR
   END FOR

   //To display the selected consequents
   FOR each v of Vp
     DISPLAY v based on the following hierarchy;

        IF inheritance is considered
         i )  Base class
                    - Attribute of base class
                    - Method of base class

         ii)  Derived class
                      - Attribute of derived class
                      - Method of derived class
        ELSE
                  Class
                  - Attribute of class
                  - Method of class
        END IF
   END FOR
END
```

Figure 4.5: Algorithm of the model building process

Chapter 5

Testing

5.1 Testing description

The implemented conversion technique is tested on its ability to reliably convert an ecological scenario model generated from the system dynamics modelling paradigm of the source domain theory to the object-oriented modelling paradigm of the target domain theory. Given a set of modelling environment which enables a scenario model to be built within the framework of the source domain theory, the performance of the system is measured based on its ability to identify relevant model fragments of the target domain theory that correspond to the similar scenario. As there is not necessarily a one-to-one mapping between corresponding fragments of the two domain theories, a consistent and adequate set of modelling environment, to enable the building of a scenario model within the target domain theory framework, is searched.

Therefore, two textbook's standard ecological modelling scenarios have been identified to be used in the testing (Mackenzie et al., 1998). The description of these scenarios, the input and output involved in the whole process, are provided in the next two sections.

5.2 First scenario

In the first scenario, the growth phenomenon of a single population (i.e. Rabbit) which considers a logistic growth rate is modelled.

5.2.1 Input description

Within the source domain theory framework, all the participants of the system dynamics modelling paradigm which correspond to this scenario are introduced by two model fragments. These participants include a population level and growth reproduction, with an equation involving the participants, which is relevant to the logistic growth model. Figure 5.1 depicts these participants and shows how they are related to their underlying assumptions.

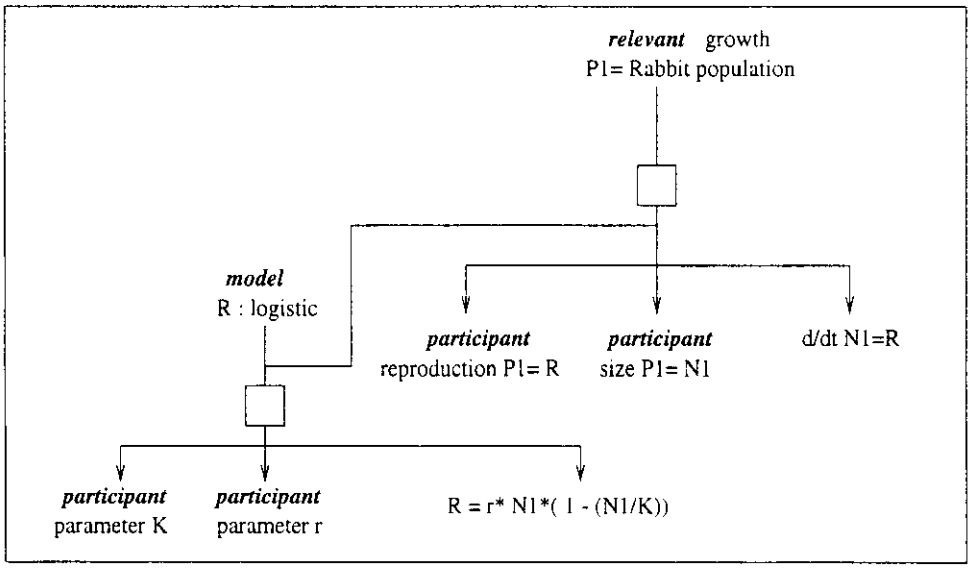

Figure 5.1: Participants of the selected model fragments and their related underlying assumptions

The selected model fragments are basically conditioned on the following assumptions:

(relevant growth ?population)

and

(model logistic ?population_reproduction)

Assuming that the model formulation task on the source domain theory is completed successfully, the generated modelling environment is simply these two assumptions applied to the *Rabbit* population and its growth rate. These assumptions, listed in a text input file as illustrated in table 5.1, are then fed to the conversion technique.

A text file of modelling environment
relevant growth Rabbit model logistic Rabbit_reproduction

Table 5.1: Text file consisting the modelling environment

5.2.2 Generated output

In the *model fragment selection* stage, given the set of modelling environment obtained from the source domain theory, the corresponding model fragments of the target domain theory are selected. The instantiated assumptions of these selected fragments are extracted and fed to the *model composition* phase. The initial assumptions set obtained from the model fragment selection stage, and the set obtained after the model composition process has been completed are illustrated in table 5.2. For brevity, the instantiated object of the assumptions (i.e. Rabbit) is omitted.

This modelling environment, generated by the model composition process, enables the building of a scenario model within the object-oriented framework. This model is expected to correspond to the initial scenario model produced within the system dynamics framework. This generated

Status of assumptions set	Content of assumptions set
Pre model composition process, V_I	{Growth-relevance=growth-phenomenon, Growth-model=logistic}
Post model composition process, V	{Growth-relevance=growth-phenomenon, Growth-model=logistic, Intrinsic-growth-rate-model=close-population }

Table 5.2: Table of assumptions set and its content

modelling environment is then fed to the *model building* process, which allows the consequents of the selected model fragments which are conditioned on this generated modelling environment to be combined and displayed. The object-oriented model produced by this conversion technique is illustrated in figure 5.2 and 5.3.

```
class Population {          //the class definition

public:                     //members of class are public

float Biomass;              //attributes
float Derivative;
float Growth;
float Carrying_capacity;

Population();               //class constructor
void Setmass(NewValue);     //method for setting Biomass
void Change (Amount);       //increment the Derivative
void Grow ();               //method to calculate growth rate

};
```

Figure 5.2: Class definition of an object-oriented model

```
Population::Population() { }

Population::SetMass (NewValue)
{   Biomass = NewValue;
    Derivative = 0.0;
}

Population::Change (Amount)
{   Derivative = Derivative + Amount; }

Population::Grow ()
{ Change(Growth*Biomass*(1-(Biomass/Carrying_capacity))); }


Population Rabbit;
```

Figure 5.3: Class implementation and object instantiation of an object-oriented model

5.3 Second scenario

The second scenario involves the modelling of a predator population (i.e. Fox) and a prey population (i.e. Rabbit), which considers a Lotka-Volterra model of predation.

5.3.1 Input description

Within the source domain theory framework, all the participants of the system dynamics modelling paradigm which correspond to the scenario are introduced. These include two population levels (i.e. prey and predator), their growth rates, and the Lotka-Volterra equation involving the mentioned participants. Figure 5.4 depicts these participants and shows how they are related to their underlying assumptions.

The selected fragments of this scenario are basically conditioned on the following assumptions:

(relevant growth ?population1) <— { The prey population}
(relevant growth ?population2) <— { The predator population}
(relevant predation ?population2 ?population1)
(model lotka-volterra (predation ?population2 ?population1))

Assuming that the model formulation task on the source domain theory is completed successfully, the generated modelling environment is simply the four assumptions applied to the *Rabbit* and *Fox* populations. These assumptions, listed in a text input file as illustrated in table 5.3, are then fed to the conversion technique.

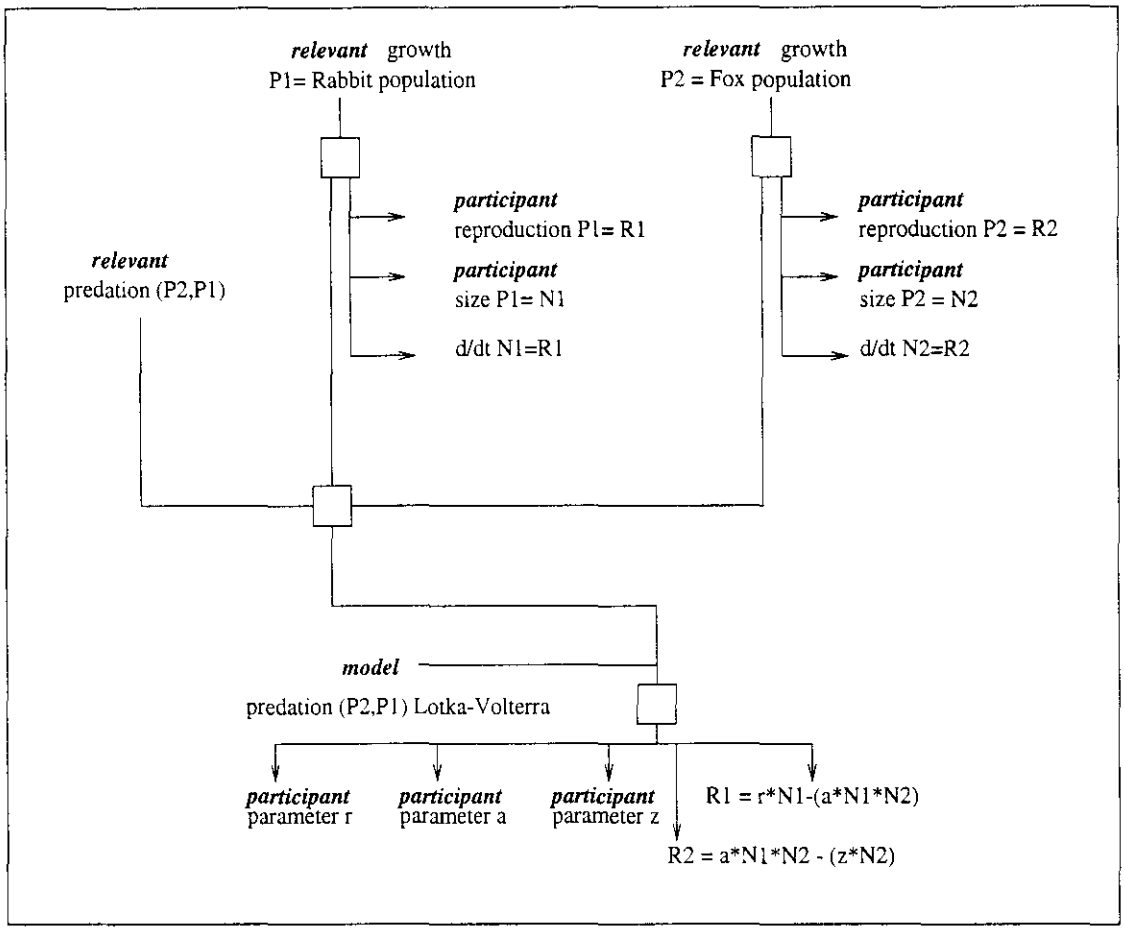

Figure 5.4: Participants of the selected model fragments and their related underlying assumptions

A text file of modelling environment
relevant growth Rabbit
relevant growth Fox
relevant predation Fox Rabbit
model lotka-volterra Fox Rabbit

Table 5.3: Text file consisting the modelling environment

5.3.2 Generated output

The initial assumptions set and the set obtained after the *model composition* process has been completed are illustrated in table 5.4. For brevity, the instantiated objects of the assumptions (i.e. Rabbit and Fox) are omitted.

Status of assumptions set	Content of assumptions set
Pre model composition process, V_I	{Growth-relevance=growth-phenomenon, Predation-relevance=predation-phenomenon, Predation-model=lotka-volterra}
Post model composition process, V	{Growth-relevance=growth-phenomenon, Predation-relevance=predation-phenomenon, Predation-model=lotka-volterra, Intrinsic-growth-rate-model=close-population, Growth-model=exponential}

Table 5.4: Table of assumptions set and its content

This set of generated modelling environment enables a scenario model which utilises the object-oriented paradigm to be built. This generated modelling environment is then fed to the *model building* process, which allows the consequents of the selected model fragments that are conditioned on this set to be displayed in an object-oriented modelling form. The object-oriented

model produced by this conversion technique is illustrated in figure 5.5 and 5.6.

```
class Population {          //the base class definition

public:                     //members of class are public

float Biomass;              //attributes
float Derivative;

Population ();              //class constructor
void Setmass (NewValue);    //method for setting Biomass
void Change (Amount);       //increment the Derivative
virtual void Grow ();       //to be redefined in descendant class
};

//the derived class definition

class Prey public Population {

public:                     //members of class are public

float Growth;               //attribute

Prey(): Population();       //derived class constructor
void Grow ();               //grow method of Prey
};

class Predator public Population {

public:                    //members of class are public


float Appetite;             //attributes
float Mortality;

Predator(): Population(); //derived class constructor
void Eat (Prey Victim);     //predation of the Predator
void Grow();                //grow method of Predator
};
```

Figure 5.5: Class definition of an object-oriented model

```
Population::Population() { }

Population::SetMass (NewValue)
{   Biomass = NewValue;
    Derivative = 0.0;
}

Population::Change (Amount)
{   Derivative = Derivative + Amount; }

Population::Grow () { }

Prey::Prey() : Population() { }

Predator::Predator() : Population() { }

Prey::Grow ()
{ Change(Growth*Biomass); }

Predator::Grow ()
{ Change(-Mortality*Biomass); }

Predator::Eat (Prey Victim)
{ Victim.Change(-Appetite*Biomass*Victim.Biomass);
  Change(Appetite*Biomass*Victim.Biomass);
}

Predator Fox;
Prey Rabbit;
```

Figure 5.6: Class implementation and object instantiation of an object-oriented model

5.4 Discussion

Based on the resulting models of the two scenarios produced within the object-oriented modelling paradigm, it can be said that the conversion of compositional models from the system dynamics modelling paradigm to the object-oriented modelling paradigm is satisfactorily performed by the implemented conversion technique. The conclusions that can be made on these results, by comparing the models produced within these two modelling paradigms (refer to figure 5.1, 5.2 and 5.3 for the first scenario; and figure 5.4, 5.5 and 5.6 for the second scenario) are as follow:

- In the first scenario, the logistic reproduction rate of the Rabbit population, which is modelled within the system dynamics modelling paradigm as a direct first-order differential equation involving the Rabbit population level and the other relevant participants, is satisfactorily reflected in the model produced within the object-oriented modelling paradigm, particularly in the defined *Grow*() and *Change*() methods, which utilised the *Growth*, *Biomass* and *Carrying-capacity* attributes in order to achieve the similar aim.

- In the second scenario, the Lotka-Volterra model of the predator-prey system involving the Fox and Rabbit populations is satisfactorily represented within the scenario model produced in the object-oriented modelling paradigm. In this paradigm, the Lotka-Volterra equations involving the two classes (i.e. Prey and Predator) are explicitly integrated within the *Grow*() and *Change*() methods of both population classes, and also the *Eat*() method of the Predator class.

- Based on the two examples above, it is shown that the implemented conversion technique works in the given scenarios, which allow similar scenarios to be represented satisfactorily in both modelling paradigms.

The main focus of this testing is to prove that that the proposed conversion technique works, given a pair of equivalent domain theories, each utilising a different modelling paradigm. Therefore, the constructed small-scale domain theories of ecological phenomena, used in this testing, are considered adequate in fulfilling this objective.

One of the problems faced by this project is the absence of existing large-scale compositional domain theories which utilise multiple modelling paradigms. Therefore, the testing cannot be extended to include a larger scale domain theory. Given this problem, one of the solutions is to expand the existing compositional domain theories of ecological modelling to cover many more possible phenomena within this domain. However, for such task, further research time and commitment are required, and it is beyond the scope of this project.

5.5 Summary

This chapter describes the testing performed on the implemented conversion technique. For the testing, two ecological scenarios are used. The input provided to the applied system and the output generated by the system in both scenarios are described.

The first scenario deals with a population growth phenomenon that considers a logistic reproduction rate. In the second scenario, a predation phenomenon involving two populations is examined. The Lotka-Volterra model of predation is considered for the modelling of this second scenario.

Chapter 6

Conclusion and Future work

6.1 Conclusion

This dissertation has presented an initial work towards the development of a technique for converting compositional models between different paradigms or representations. Assuming that the model formulation task in a particular domain theory is completed successfully, the generated modelling environment, from which the whole modelling process is conditioned, is fed to the proposed conversion technique. This work, therefore, relies on this set of modelling environment to select and instantiate the corresponding fragments of the other domain theory.

One of the major conversion problems is that the mapping between corresponding fragments of this pair of domain theories is very unlikely to be in a one-to-one manner. It might be the case that a model fragment in one domain theory covers part of several model fragments in the other domain theory. Given this situation, further adequacy and consistency checking need to be performed on the assumptions set from which the selected model fragments of the other domain theory is conditioned. In this work, this task is cast as a dynamic constraint satisfaction problem (DCSP). Each assumption

class of the other domain theory is used to represent a DCSP attribute, and the corresponding domain values of each attribute is the set of assumptions defined within that class. Activity and compatibility constraints, applied around these assumptions, allow an adequate and consistent set of modelling assumptions to be obtained at the end of the process. A *choose* and *repair* techniques are integrated within the DCSP. The choose technique allows any unassigned attribute of the set to be assigned with a value from the corresponding domain. On the other hand, the repair technique allows any value assignment that causes inconsistency to be unassigned, and a new value assignment is performed.

As a prove of concept, the theoretical design of this conversion technique is implemented in the C++ programming language, and the compositional modelling of ecological systems is used as a testing domain for the implemented conversion technique. For this work, system dynamics and object-oriented are the two modelling paradigms adopted for the knowledge representation framework of the domain theories used. The major intention of this conversion application is to convert a system dynamics compositional model, to an object-oriented compositional model. The resulting object-oriented model is expected to reflect the same scenario, but with a different representation, compared to the model produced within the system dynamics modelling paradigm.

In the testing, it is shown that the proposed conversion technique works in the given selected scenarios.

6.2 Future work

It is not possible, within the time available, to extent this project beyond the objectives described in chapter 1. Some suggestions for future work in the area of compositional model conversion are outlined below in two different categories; what would be a short term work, and what would require a much more substantial project:

Short term work:

- In this work, though the conceptual design of the proposed conversion technique is not restricted to ecological modelling, the implementation is specifically focused around the ecological phenomena constructs of the domain theories used for the testing. As a consequence, the application's applicability is highly restricted on this domain. In order to apply the current work on the other domains, a number of modifications need to be performed on the existing code of this application. Given this situation, the attention for a short term work in the near future is to consider a more generic architectural approach, in terms of the implementation aspect of this proposed conversion technique.

- Two ecological phenomena involving populations, namely population growth phenomenon and predation phenomenon are examined in the current work. Future extension that might be considered for this existing list will include a competition phenomenon between populations that rely on the same source for food (e.g. Fox and Hawk that feed on Rabbit). This competition phenomenon can be modelled as either "intraspecific" (i.e. competition within the same species) or "interspecific" (i.e. competition between different species) (Mackenzie et al., 1998).

Substantial project:

- In order for the conversion technique to work, a set of consistent and adequate modelling environment is required. In this work, to obtain this mentioned set, the compositional modelling algorithm of Falkenhainer and Forbus (1991) is assumed to be applied to the source domain theory. In addition, the knowledge representation framework of the domain theories involved in this conversion also follows the notation introduced in the Falkenhainer and Forbus (1991) work. This indicates an explicit dependency of the proposed conversion technique on this compositional modelling framework. However, as there exist a number of other compositional modelling approaches (e.g. QPC, Causal approximation, TRIPEL, DME, Probabilistic), future work should consider extending the conversion technique on these frameworks.

- The major part of this work focuses on resolving the adequacy and consistency issues which rise due to the fact that the mapping of the corresponding model fragments between these two domain theories are not in a one-to-one manner. A dynamic constraint satisfaction technique is used to guide the search for a model that meets the adequacy and consistency requirements. In this work, the task of translating the given modelling environment and the target model fragment library into a dynamic constraint satisfaction problem is done manually. Future work should consider incorporating the method proposed by Keppens and Shen (2000) to automate this translation process.

- The conversion technique should also consider other forms of input, besides the one used in this work (i.e. a set of modelling environment). For instance, given a complete scenario model, how the process of identifying the scenario description and modelling decisions will be

handled.

- A number of research groups, working in different domains, have constructed large knowledge bases with extensive domain theories for compositional modelling. These include Collins and Forbus (1990) work on the construction of a large-scale domain theory of engineering knowledge about thermodynamics; Porter et al. (1988) work on the construction of a large, multi-function domain theory about botany; and Catino (1993) who has developed a large domain theory within the chemical engineering domain. However, these constructed domain theories only consider a single modelling paradigm. Given the rigorous efforts and commitments, directed into the creation and refinement of high-quality knowledge bases of model fragments for specific domains in science and engineering, it is hoped that domains which utilise multiple modelling paradigms might be available in the near future. This will greatly benefit future work in the area of compositional model conversion. In short, before the area of compositional model conversion can be fully explored, one of the crucial factors that has to be considered is the research and development in the area of multiple modelling paradigms domain theories. The development of the former definitely relies on the progress made on the latter.

Bibliography

Begon, M., Harper, J., and Townsend, C. (1996). *Ecology: Individuals, Populations, and Communities - 3rd edition.* Blackwell Science Ltd. Cambridge, MA.

Bobrow, D., Falkenhainer, B., Farquhar, A., Fikes, R., Forbus, K., Gruber, T., Iwasaki, Y., and Kuiper, B. (1996). A compositional modelling language. *Proceedings of the 10th International Workshop on Qualitative Reasoning about Physical Sytems*, (12-21).

Booch, G. (1994). *Object-Oriented Design with Applications-Second Edition.* The Benjamin/Cummins Publishing Company, Inc.

Catino, C. (1993). *Automated Modeling of Chemical Plants with Application to Hazard and Operability Studies.* PhD thesis, Department of Chemical Engineering, University of Pennsylvania, Philadelphia, PA.

Colinvaux, P. (1986). *Ecology.* John Wiley and Sons.

Collins, J. and Forbus, K. (1990). Building qualitative models of thermodynamic processes. Technical report, Beckman Institute, University of Illinois.

Falkenhainer, B. and Forbus, K. (1991). Compositional modeling: finding the right model for the job. *Artificial Intelligence*, 51(95-143).

Forrester, J. (1961). *Industrial Dynamics.* MIT Press.

Holling, C. (1965). The functional response of predators to prey density and its role in mimicry a population regulation. *Mem. Entomol. Soc. of Canada*, 46(60).

Iwasaki, Y. (1997). Guest editor's introduction: Real-world applications of qualitative reasoning. *IEEE Expert*, 12(3)(74-82).

Keppens, J. and Shen, Q. (2000). Towards compositional modelling of ecological systems via dynamic flexible constraint satisfaction. *Proceedings of the 14th International Workshop on Qualitative Reasoning about Physical Systems*, (74-82).

Keppens, J. and Shen, Q. (2001). On compositional modelling. *The Knowledge Engineering Review*, 16(2)(171-214).

Kuipers, B. (1994). *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge.* MIT Press.

Levy, A., Iwasaki, Y., and Fikes, R. (1997). Automated model selection for simulation based on relevance reasoning. *Artificial Intelligence*, 96(351-394).

Lippman, S. and Lajoie, J. (1998). *C++ Primer - Third Edition.* Addison-Wesley.

Mackenzie, A., Ball, A., and Virdee, S. (1998). *Instant Notes in Ecology.* BIOS Scientific Publishers Ltd.

Martin, J. and Odell, J. (1992). *Object-Oriented Analysis and Design.* Prentice-Hall, Englewood Cliffs, New Jersey.

Miguel, I. and Shen, Q. (1999). Hard, flexible and dynamic constraint satisfaction. *Knowledge Engineering Review*, 14(3)(199-220).

Mittal, S. and Falkenhainer, B. (1990). Dynamic constraint satisfaction problems. *Proceedings of the 8th National Conference on Artificial Intelligence*, (25-32).

Nayak, P. (1995). *Automated Modeling of Physical Systems - Lecture Notes in Artificial Intelligence.* Springer.

Porter, B., Lester, J., Murray, K., Pittman, K., Souther, A., Acker, L., and Jones, T. (1988). Artificial intelligence research in the context of a multifunctional knowledge base: The botany knowledge base project. Technical Report TR AI-88-88, Artificial Intelligence Laboratory, Department of Computer Science, University of Texas at Austin.

Robertson, D., Bundy, A., Muetzelfeldt, R., Haggith, M., and Uschold, M. (1991). *Eco-logic. Logic-Based Approaches to Ecological Modelling.* MIT Press.

Sellappan, P. (1994). *C++ Through Examples.* Federal Publications Sdn. Bhd.

Silvert, W. (1992). Object-oriented simulation programming in ecological modelling. *8th ISEM International Conference on Ecological Modelling.*

Silvert, W. (1993). Object-oriented ecosystem modelling. *Ecological Modelling*, 68(91-118).

Swan, T. (1999). *GNU C++ for Linux.* Que Corporation.

Verfaillie, G. and Schiex, T. (1994). Solution reuse in dynamic constraint satisfaction problems. *Proceedings of 12th National Conference on Artificial Intelligence*, (307-312).

Appendix A

User manual

A.1 The program files

The three subprocesses of the proposed conversion technique, namely the *model fragment selection*, the *model composition* and the *model building*, are split into two separate C++ programs. The model fragment selection and the model composition processes are combined as one program, and the program file is named **Conversion.cpp**. The model building process is set up as a separate C++ program which stands on its own, and the program file is named **Model_building.cpp**.

A.2 Input text files

The modelling environment (i.e. set of assumptions) of the source domain theory (i.e. system dynamics modelling paradigm) is defined in a text input file named **source_modelling_environment.txt**. Each assumption should be written as a line of characters corresponding to the format defined in the following, with a space in between the different element constructs of the assumption.

```
Legends:

At  <---- Assumption type
As  <---- Assumption subject
Ob1 <---- Instantiated object 1
Ob2 <---- Instantiated object 2

Format of definition:

[At] [As] [Ob1] [Ob2]
```

Certain *assumption subject* can only be applied to a single instantiated object, therefore the *instantiate object 2* part of these assumptions can be omitted.

Table A.1 summarised the *assumption type* and *assumption subject* that correspond to the ecological modelling phenomena covered in this work. For more details on how these assumptions are derived, please refer to chapter two on the section of ecological modelling, and chapter four on knowledge representation.

Based on the assumptions summarised in table A.1, in order to define a *model* assumption type which corresponds to the logistic growth rate of a particular Penguin population, the following statement is written to the text file:

```
model logistic Penguin_reproduction
```

For more example, please refer to chapter four on the section of input description.

Assumption type	Assumption subject
relevant	growth
	predation
model	logistic
	exponential
	lotka-volterra
	holling-tanner
	open-population
	close-population

Table A.1: Table of assumption type and its corresponding subject

A.3 Compiling the program files

In order to execute the program files, they need to be compiled first. The GNU C++ compiler, which is installed in the AI machines in South Bridge and Forrest Hill sites can be used for the compilation. There are many ways to compile the program. The easiest method, and the one explained in this manual is by running the GNU C++ compiler. To compile **Conversion.cpp** program file for instance, enter the following command:

```
$ g++ Conversion.cpp
```

The dollar sign represents the shell prompt. Line preceded with the dollar sign is those you type at the console. *Don't type the dollar sign.* To run the compiler, you may either type **g++** or **c++**. In some version of UNIX, these commands actually run the GNU C compiler, gcc, with options selected for C++ programming. In newer release, GNU C++ is the stand-alone compiler, egcs (Experimental GNU Compiler System) which does not rely on gcc.

After you have compiled **Conversion.cpp**, a directory listing produced by the **ls** command shows a new file named **a.out**. This file contains the executable code that g++ has created. It is a finished code file, ready to run. To run the program, preface its name with a period and slash. This tells the shell to look for the file in the current directory. For example, in order to run the executable code, enter the following command:

```
$ ./a.out
```

A.4 Expected screen displays

Consider a text file consisting the modelling environment as described in table 5.3, is provided to the **Conversion.cpp** program. After compiling and executing this program, the following screen displays are expected to appear on the screen (comments for each segment of the screen display are provided in bracket) :

```
-------------------------------------------------------------

(The following indicates the generated output produced by the
 model fragment selection process. Two growth-phenomenon
 fragments are selected and instantiated, corresponding to the
 two populations - Rabbit & Fox. The other two model fragments
 which are applicable to these two populations are
 predation-phenomenon and lotka-volterra-model.)

The number of selected fragment : 4
 The fragment's name is : growth-phenomenon
 The fragment's name is : growth-phenomenon
 The fragment's name is : predation-phenomenon
 The fragment's name is : lotka-volterra-model

-------------------------------------------------------------
```

```
(The assumptions obtained from the selected model fragments of
 the model fragment selection process become the initial
 attribute set. Each assumption class (i.e. an attribute) and
 its corresponding assumption value are displayed until all
 defined constraints have been satisfied (refer to table 4.1
 for description of the displayed attributes and domain values).
 In displaying the current content of V, the data structure used
 in holding the attribute set, the instantiated objects which
 are associated with different particular assumptions are
 omitted. Therefore, though the same assumption class and value
 assignment might exist more than once (i.e. applicable to
 different populations or instantiated objects of the problem),
 it is only displayed once. Each active constraint out of the
 six defined activity and compatibility constraints, is identified
 and displayed (refer to table 4.2 and 4.3 for description on these
 defined constraints). This program also provides a step-by-step
 tracing of the model composition process, which requires a user to
 key in any keyboard character value and then press the <ENTER> key
 to move to the next stage. The process is ended when all attributes
 have been assigned a domain value.)


INITIAL  CONTENT OF V :
----------------------
Growth-relevance = growth-phenomenon
Predation-model = Lotka-Volterra
Predation-relevance = predation-phenomenon


Please press any character (ie. A-Z, a-z)
and then press <ENTER> to continue....


Active constraint : C1


CURRENT  CONTENT OF V :
----------------------
Growth-relevance = growth-phenomenon
```

```
Predation-model = Lotka-Volterra
Predation-relevance = predation-phenomenon
Growth-model = unassigned


Please press any character (ie. A-Z, a-z)
and then press <ENTER> to continue....


Active constraint : C4


CURRENT  CONTENT OF V :
----------------------
Growth-relevance = growth-phenomenon
Predation-model = Lotka-Volterra
Predation-relevance = predation-phenomenon
Growth-model = unassigned
Intrinsic-rate-of-increase-model = close-population


Please press any character (ie. A-Z, a-z)
and then press <ENTER> to continue....

New assignment: Growth-model = logistic


CURRENT  CONTENT OF V :
----------------------
Growth-relevance = growth-phenomenon
Predation-model = Lotka-Volterra
Predation-relevance = predation-phenomenon
Growth-model = logistic
Intrinsic-rate-of-increase-model = close-population


Please press any character (ie. A-Z, a-z)
and then press <ENTER> to continue....


Active constraint : C6
```

```
CONFLICT ALERT : A conflict on value assignment
of Growth-model = logistic

New assignment : Growth-model = exponential


CURRENT  CONTENT OF V :
----------------------
Growth-relevance = growth-phenomenon
Predation-model = Lotka-Volterra
Predation-relevance = predation-phenomenon
Growth-model = exponential
Intrinsic-rate-of-increase-model = close-population


Please press any character (ie. A-Z, a-z)
and then press <ENTER> to continue....

No existing conflict on all attribute assignments


FINAL  CONTENT OF V :
---------------------
Growth-relevance = growth-phenomenon
Predation-model = Lotka-Volterra
Predation-relevance = predation-phenomenon
Growth-model = exponential
Intrinsic-rate-of-increase-model = close-population


Please press any character (ie. A-Z, a-z)
and then press <ENTER> to continue....

-----------------------------------------------------------------
```

```
-------------------------------------------------------------

(The following is the eventual set of modelling environment
 generated by the model composition process. This set will be
 written to a text file named target_modelling_environment.txt.
 Model_building.cpp program relies on this text file to
 generate a compositional scenario model within the
 object-oriented modelling framework. The object-oriented model,
 described in figure 5.5 and 5.6, is generated based on this
 set of modelling environment.)

relevant growth Fox
relevant growth Rabbit
model exponential Fox-reproduction
model exponential Rabbit-reproduction
model close-population Fox
model close-population Rabbit
relevant predation Fox Rabbit
model lotka-volterra Fox Rabbit

-------------------------------------------------------------
```

A.5 Concluding remark

These two programs (i.e. **Conversion.cpp** and **Model_building.cpp**), are developed as a prove of concept for the proposed compositional model conversion technique. To a large degree, the objectives specified in chapter one have been fulfilled. However, in terms of the implementation, it should be noted that these two programs are far from complete, and further work in refining and enhancing these two programs are required. Therefore, for any comment, further query or useful discussion, feel free to contact the author at the following e-mail address:

fadzil@eudoramail.com