Designing a Solid Waste Infrastructure Management Model for Integration into a National Infrastructure System-of Systems

Ives, M. C.[1], Coello, J.[2], Turner, D.[2], Watson, G.V.R[2], Stringfellow, A.M.[2], Powrie, W.[2] and Hall, J.[1]
[1] University of Oxford, UK
[2] University of Southampton, UK

ABSTRACT

Solid waste management is arguably one of the most important municipal services provided by government[1]. Given the rapid socio-economic changes that are projected to take place in the UK[2] it is important that we plan our future waste management capacity to ensure the continuance of this valuable service. The Solid Waste Infrastructure Management System (SWIMS) model was designed to model the current solid waste infrastructure requirements (from collection through treatment and disposal) for an area based on its solid waste arisings. SWIMS allows an area's waste treatment capacity requirements to be forecast against future socio-economic change to help decision-makers choose the right solid waste infrastructure given their goals, constraints and ideas about future conditions. The modelling of solid waste management systems has been carried out since the 1970s[3] and such modelling exercises have been undertaken for numerous different geographical areas around the world[4]. However, the SWIMS model is unique in that it was designed to also operate within a larger national infrastructure system-of-systems model, including interdependencies with other infrastructure sectors including energy, water and waste water. To achieve such flexibility the SWIMS model was carefully designed using object-oriented programming (OOP) principles. In documenting this model's design methodology we hope to demonstrate how applying OOP principles enables such models to not only be more flexible and more easily integrated with other modelling efforts, but also more easily understood by system experts and end-users.

Keywords: Solid Waste Infrastructure Modelling, Object Oriented Programming, Model Design, Model Re-use

INTRODUCTION

The SWIMS model was born out of the requirements of the Infrastructure Transitions Research Consortium (ITRC) national infrastructure system-of-systems modelling project[5]. The key aim of the ITRC project is to inform decisions regarding planning and investment in national infrastructure systems (i.e. Energy, Transport, Water, Waste, and

1 Hoornweg, Daniel; Bhada-Tata, Perinaz. What a Waste : A Global Review of Solid Waste Management, World Bank, Washington, DC, World Bank. 2012. https://openknowledge.worldbank.org/handle/10986/17388

2 ONS. "National population projections, 2010-based statistical bulletin", Office for National Statistics (2010).

3 Morrissey, A. J. & Browne, J. Waste management models and their application to sustainable waste management. Waste Management 24, 297-308, doi:http://dx.doi.org/10.1016/j.wasman.2003.09.005 (2004).

4 Beigl, P., Lebersorger, S. & Salhofer, S. Modelling municipal solid waste generation: A review. Waste Management 28, 200-214, (2008).

5 Hall, J. W., Henriques, J. J., Hickford, A. J. & Nicholls, R. J. Systems-of-systems analysis of national infrastructure Proceedings of the ICE – Engineering Sustainability 166, 249–257 (2013).

Information & Communications Technology) by evaluating the performance of national infrastructure strategies in providing infrastructure services under a wide range of future conditions. To achieve this aim the project required the development of a system-of-systems model, known as NISMOD-LP, which enabled the analysis of each of the individual systems, their interdependences and their performance in a wide range of possible future socio-economic and climatic scenarios and management strategies.

The SWIMS model was designed from the beginning to become the solid waste infrastructure component of this larger NISMOD-LP system-of-systems model[1]. SWIMS was also designed to operate as a stand-alone solid waste infrastructure modelling tool, enabling decision-makers from any area or country to analyse their future solid waste infrastructure capacity requirements.

In order to meet these dual aims the model had to meet several key requirements. Firstly, it had to be capable of modelling the capacity requirements of a given area, either autonomously or under instruction. Secondly, the system had to track the infrastructure capacity requirements of an area's waste streams through time from generation to collection, treatment and final disposal as either an exported waste, a new product or as landfilled waste. Thirdly, the system had to be able to model any area of any geographic scale made up of smaller modular areas. It could thus be made to represent a city, a county, or a state including all the regions within a state. Fourthly, the system had to be able to explore the many alternative ways in which waste can flow through the system and analyse the environmental and financial costs and benefits of these alternative options under a range of future conditions (scenarios). It must therefore be capable of providing information on the relative worth of alternative infrastructure strategies in achieving area-specific goals (such as maximising materials recovery or minimising costs), given a set of management constraints (such as EU Landfill Directive[6]), and given a host of possible future scenarios (such as demographic and economic changes). Finally, in order to function effectively as a component within a larger UK national infrastructure system-of-systems the SWIMS model needed to be open to manipulation by NISMOD-LP for the assessment of system-wide management strategies, and interdependencies with other infrastructure sectors (such as energy, transport and waste water). Thus, the SWIMS model needed to be capable of operating both as a stand-alone model of solid waste capacity management and as an integrated component within the NISMOD-LP system-of-systems.

SYSTEM DESIGN

As a stand-alone modelling system, SWIMS model the flows of heterogeneous waste material streams (and their associated material properties) through a waste management system from the point of generation through collection, treatment and disposal based on the principle of mass balance. Potential environmental impacts associated with the management of these waste flows are quantified using Life Cycle Assessment with system expansion (to account for the recovery of materials and energy), whilst financial costs are calculated using Cost-Revenue Analysis. Figure 1 displays a high-level view of the main processes in the model by which the waste that is generated by various waste producer groups is collected, treated and disposed of by a 'waste manager' within the system.

Waste is generated in the model based on a Household Waste Function[7] that relates population and economic growth to arisings of waste types. This waste is then collected via a range of possible collection methods and converted into waste streams based on their means of collection (household kerbside collected residual waste, dry recyclables and organic wastes, etc). Each of these waste streams is treated and converted to either commercial products (energy, metals, and plastics) or another waste stream that is treated until the residual is sent for final disposal. A 'waste manager' agent manages this entire process using the available infrastructure and their costs and capacities to fulfil these waste management requirements within certain specific economic and environmental goals and constraints. The model also forecasts possible changes in demand and capacity utilisation under different socio-economic scenarios and makes decisions regarding new investment using a range of alternative management strategies. Such a system allows decision makers to test these alternative management strategies against future uncertainties allowing them to gain insights into the conditions and strategies that could lead to regional capacity shortfalls or where over-capacity can lead to the phenomena of stranded assets.

In order to best achieve the aims of this project, the SWIMS model was constructed based on real- world solid waste management components and processes through the application of object-oriented programming (OOP) principles.

6 The Landfill Directive (Council Directive 1999/31/EC) http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:31999L0031

7 Beigl, P., Lebersorger, S. & Salhofer, S. Modelling municipal solid waste generation: A review. Waste Management 28, 200-214, (2008).

In simple terms, this meant that the model design contains objects based on the actual objects in the waste management system being modelled - those same processes and components that appear in the process map in Figure 1, such as waste collection decisions, incinerators and landfill sites.

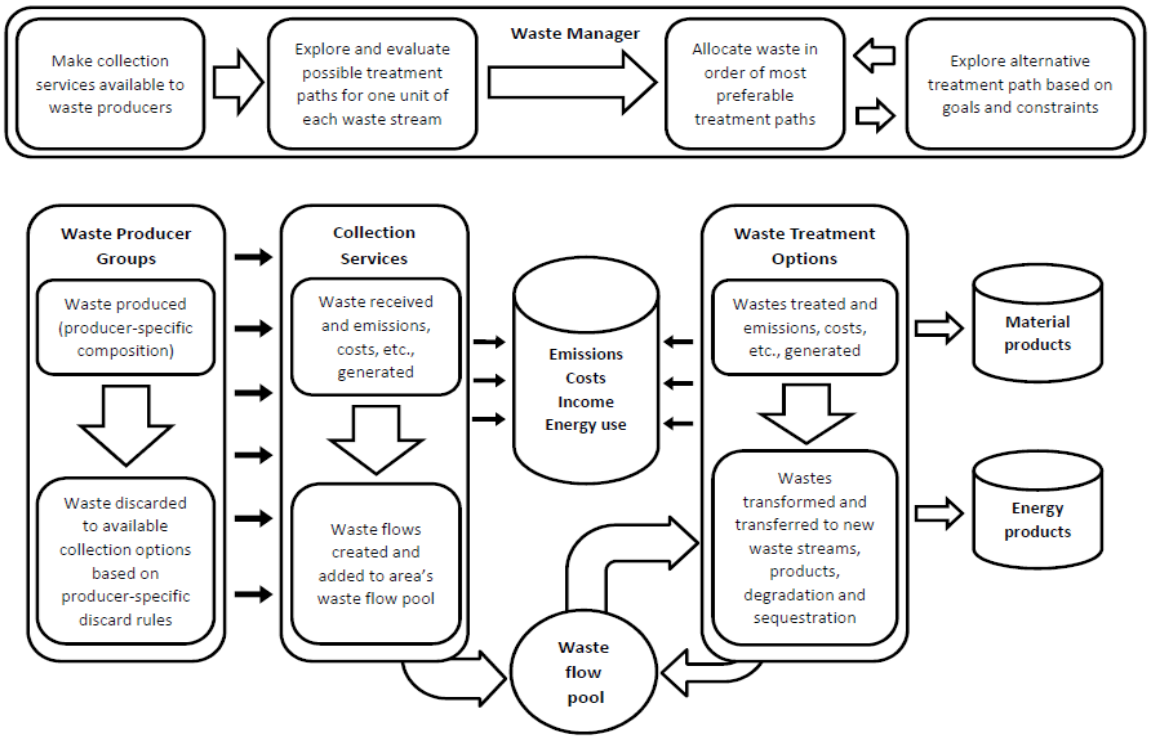

Figure 1: A process map outlining the management of the production, collection, treatment and disposal of waste

Designing the SWIMS system involved the application of the three key OOP principles of *abstraction*, *encapsulation* and *inheritance*. The concept of *abstraction* refers to the style of programming that uses objects and classes that are abstractions or representations of actual objects in the real world. An object can be a physical object, such as a tonne of waste or a landfill site, or it can be intangible, such as a management decision or the price of a commodity. A *class* is the set of rules that define an object. An object is therefore a particular instance of a class. For example, if we had a class that broadly defined all incinerators then each particular incinerator plant would be an object of the incinerator class.

As shown in the class diagram of the SWIMS model (Figure 2) the objects that make up the model are based on actual waste treatment process objects, such as Materials Recovery Facilities and Anaerobic Digesters, that have real world properties and perform functions (methods) such as processing waste and incurring costs. Although these objects only ever manipulate data within the model they still operate on this data using the same principles observed in their real world counterparts. Thus, the modelled system within SWIMS can mimic the functioning of its real-world equivalent using objects and processes that have a direct correlation to objects and processes in the real world.

Encapsulation is an OOP concept that describes a way of organising the many types of information and processes that help the individual objects to be used as efficiently and securely as possible. Encapsulation involves splitting an object up into its public interface - or what is needed to "drive" the object - and its private interface - or what makes the object work "under the hood". For instance, to ensure that the NISMOD-LP system-of-systems model does not 'break' the inner workings or logic of the SWIMS model, only those methods and properties of the objects that are required by NISMOD-LP are exposed in the model's interface.

Finally, object *inheritance* or generalisation is the process of organising the features of different types of objects that share the same purpose. For instance, the **WasteTreatmentOption** class shown in Figure 2 is a parent class from which a number of specialised child classes are derived, such as an **MFR** (Materials Recovery Facility) or an **AnearobicDigester** class. The **WasteTreatmentOption** class contains properties such as *name* and *annualCapacity* and methods such as *processWaste* or *calcCosts* that would be common properties and methods for all child classes derived from this parent class. A derived **AnearobicDigester** child class would therefore inherit all the **WasteTreatmentOption** class's properties and methods as well as its own set of unique properties and methods that are particular to the **AnearobicDigester** class (such as *biogasProduction*).

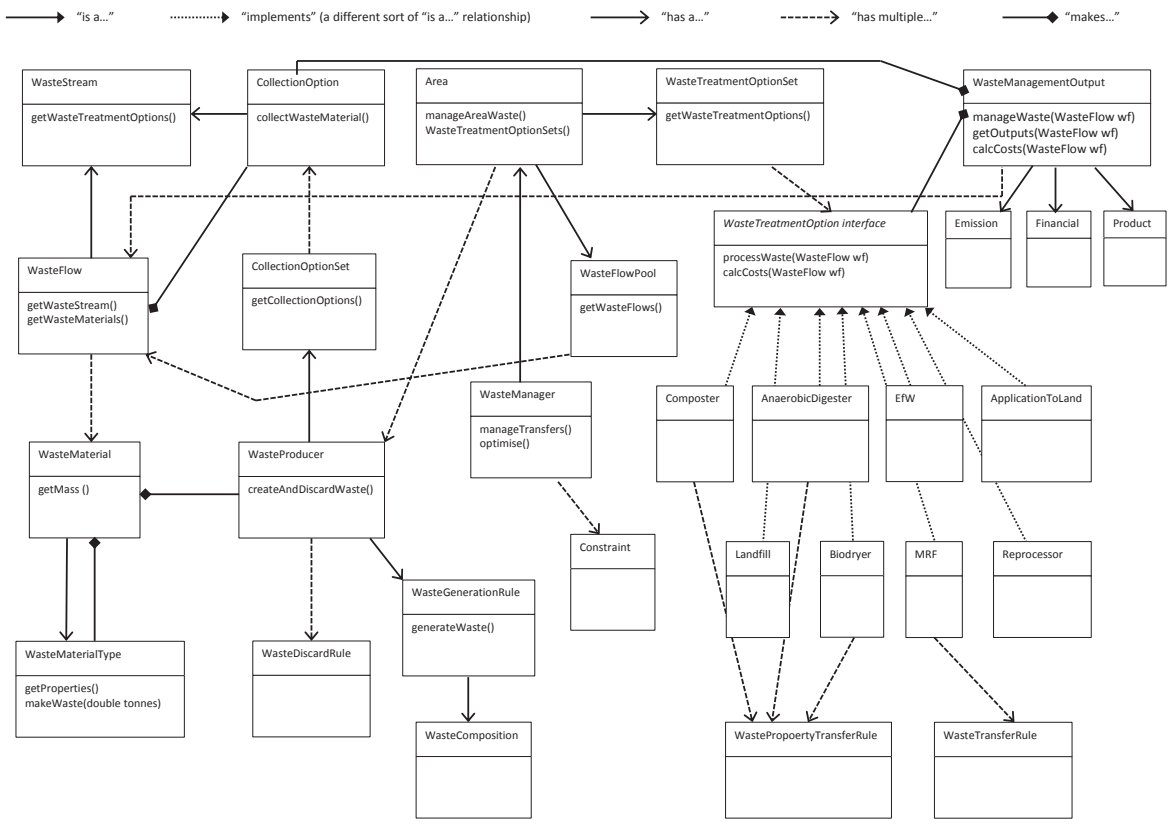

Figure 2: A class diagram for the SWIMS model. Note that to avoid unnecessary cluttering only a subset of the SWIMS model's objects, properties and methods are shown. Note that in order to avoid unnecessary clutter this class diagram does not contain all of the classes, attributes and methods present in the final design of the model.

Object inheritance is one of the key reasons for using the OOP approach. Inheritance provides reusability and a mechanism for adding features to existing classes without the need extensive modifications to the model. This is achieved by deriving a child class from an existing parent class. The new child class then has the combined features of both the inherited interface (properties and methods) of the parent class as well as its own unique properties and methods. The child class can also operate using the codebase within the methods of the parent class or it can override these processes with its own codebase. This not only provides a way of ordering and organising objects, but more importantly allows methods that are common to each child class to be coded within the parent class. This reduces the amount of code that needs to be written, as well as the amount of code that must be debugged – an important time-saving benefit. It also allows for huge flexibility in examining alternative configurations of child objects that can be slotted in and out of the model as required, because other code components that interact with this class within the model can be confident that every child class has the same interface components of the parent class.

When designing such a system using OOP it is very important to understand how the various objects work together and how they will combine to run the necessary system processes. It is therefore necessary to map out the communication and functioning of the various model objects. A number of design methods are available for this task, including writing pseudo-code or building sequence diagrams. Figure 3 shows an example of a sequence diagram for the SWIMS system, showing the process of managing the waste flows that have been generated - including waste collection, treatment and disposal. Note that as waste disposal processes act in much the same way as other treatment options it has been included as a **WasteTreatmentOption**, similar to other intermediate treatment options, such as 'mechanical biological treatment' or 'windrow composting'.

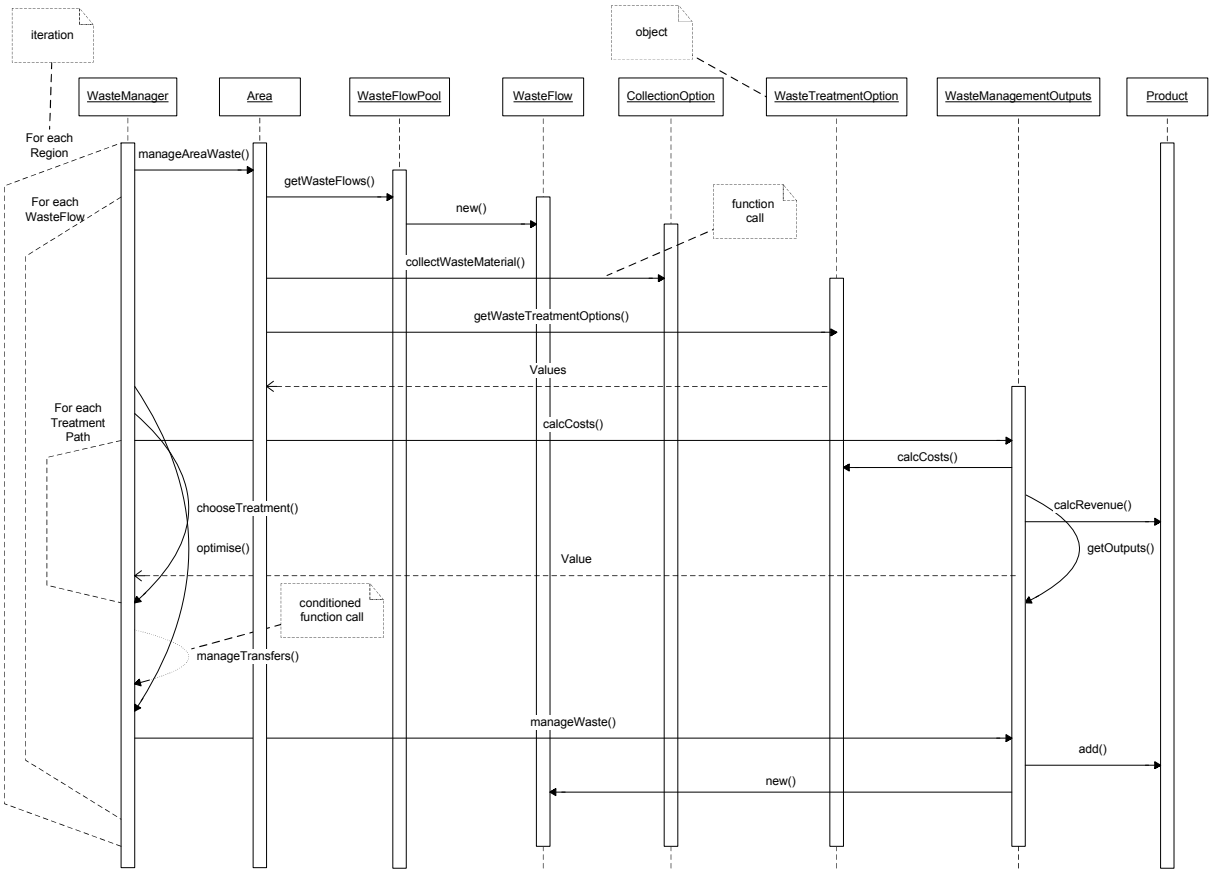

Figure 3: A sequence diagram showing the interaction between the various model classes involved in the process of treating a waste flows in the SWIMS model.

The process of mapping out the classes and functions through a sequence class diagram, or an equivalent method, provides the designers with the information necessary to build an outlay of the classes required in the model. The classes designed for the SWIMS model (shown in Figure 2) were developed through an iterative process of hypothesising the necessary objects and then modelling their communications through sequence diagramming. This can be a laborious process, but one that should greatly increase the speed with which the model code can be written once the designs are completed.

CONCLUSIONS

Building a system that is capable of modelling the management of an area's waste infrastructure is not a trivial task. Adding the ability to tightly integrate this model with a larger, more complex system-of-systems vastly increases the difficulty. Applying OOP design principles was integral to enabling the SWIMS modelling project to meet such demanding, multi-layered objectives. Through abstraction and encapsulation the same objects used to manage an area's waste are provided as objects for manipulation by the larger more complex system-of-systems. As the objects with the SWIMS model represent objects that exist in the real system their functions and attributes are more likely to reflect what the larger system needs in order to integrate it into its structure. By exposing objects, such as the **CollectionOption** object, alternative collection options can be examined for their financial or downstream implications. Allowing any child classes that have been derived from the **WasteTreatmentOptions** object to be dynamically interchanged allows the larger system to examine highly complex alternative management strategies, including those that might only be considered within the context of a larger system-of-system - such as comparing the benefits of reducing fossil fuel use by increasing energy from waste to alternative options for reducing the production of CO_2 within other infrastructure sectors such as energy and transport.

One obvious limitation of the SWIMS model is that it assumes that all future options and decision criteria have already been identified and thus the problem of solid waste management becomes a process of evaluation of

these limited set of alternatives[8]. This limitation exists for all such systems that attempt to model the management of future conditions. However, through its OOP design the inclusion of any future waste management advances into the SWIMS model is made easier as any such new option can be accommodated as child classes of the **WasteTreatmentOptions** parent class, thereby reducing the amount of code that needs to be written.

A key benefit of the OOP methodology that does not necessarily get enough recognition is its ability to facilitate communications between all parties involved in the design, development and application of the model. Building a model based on real world objects facilitates communication between all parties involved in the model development, allowing the model architects to talk to the domain knowledge experts and end-users with reference to objects and processes that are familiar to them. Clearer communications can result in a model that is much more likely to operate as expected and it facilitates the handover of the model, increasing its life and usability.

Many physical systems models are built as single-use models, particularly those used for smaller projects. This trend is partly a reflection of the fact that smaller projects tend to require models for only a very specific problem in a particular context[9]. However, in many cases models are built in a single-use format because the model developers do not have sufficient experience or training in such advanced software development methodologies. Although re-use was a requirement of this project it is our hope that this manuscript demonstrates how, through the application of object-oriented programming principles, physical system models can be built in such a way that not only makes them re-useable but also makes them easier to use. Given the advantages of the OOP methodology, it would be wise for those modelling physical systems to make the effort to learn how to apply this methodology, even if they do not have such complex, multi-layered requirements. OOP is a methodology that has been developed through decades of experience and it is for very good reasons that it is now standard within all mainstream software development languages.

8 Morrissey, A. J. & Browne, J. Waste management models and their application to sustainable waste management. Waste Management 24, 297-308, doi:http://dx.doi.org/10.1016/j.wasman.2003.09.005 (2004).

9 Bollinger, L. A., Nikolic, I., Davis, C. B. & Dijkema, G. P. J. Multi-model ecologies: Cultivating model ecosystems in industrial ecology. Journal of Industrial Ecology (**in press**) (2014).