DYNAMIND - A SOFTWARE TOOL FOR INTEGRATED MODELLING OF URBAN ENVIRONMENTS AND THEIR INFRASTRUCTURE

CHRISTIAN URICH(1), GREGOR BURGER(1), MICHAEL MAIR(1), WOLFGANG RAUCH (1)

(1): Unit of Environmental Engineering, University of Innsbruck, Technikerstr. 13, Innsbruck, 6020, Austria

The DynaMind Framework is based on VIBe, a tool for generating virtual case studies including the urban environment and the urban drainage system. In DynaMind the computational framework has been enhanced and generalised to enable the simulation of complex dynamic urban environments and their infrastructure. Simulations can be defined by modules (small working task within a simulation) and the data flow between them. A module has in-ports to receive data from the outside. The data are then processed in the module and sent to out-ports. As data format raster and vector data can be used. Different modules can be linked together to a simulation workflow that is executed in parallel by the DynaMind core. DynaMind comes with a set of basic modules. This includes simple modules for data import and export, complex modules (for example the generation and adaptation of combined sewer networks or the placement of ground water heat pump systems) as well as modules for data visualisation. New modules can easily be integrated in DynaMind by using C++ or Python. DynaMind is freely available as open source software using the GPL license. To show the applicability a simple integrated case study to investigate the effects of urban development on an existing combined sewer system is set up in this paper.

INTRODUCTION

To test the effectiveness of new technologies and strategies in urban water management and to tackle the problems induced by climate and urban change, integrated modelling approaches on city scale are required. Traditional modelling tools for urban water systems like SWMM [1] consider the urban networks as static and decoupled from the urban environment. To couple the urban environment with the water infrastructure SWMM has been integrated in GIS based software tools like Mike Urban. Even though, the urban water infrastructure is still considered as a static system and the evaluation of performance is possible only at a certain point in time.

To prepare a case study for integrated modelling on city scale huge amounts of data at a very fine scale are required. Existing data are often either of poor quality or simply not available and therefore the preparation of a case study is a very time consuming and frustrating task. The situation gets even worse when future scenarios are evaluated. Thus often only a small number of simplified story lines can be investigated where only the present state and the future state are evaluated (Semadeni-Davies et al. [2]).
The DynaVIBe approach (Sitzenfrei et al. [4]) tackles the problem by introducing a method for the algorithmic generation of integrated case studies that evolve dynamically in the future. The so called virtual case studies are based on parameters derived from real world case studies. This methodology enables an analysis of future scenarios on a spatio-temporal city scale. Based on the DynaVIBe approach a dynamic environment for integrated modelling – DynaMind has been developed and is presented in this paper.

Aim of DynaMind is to provide an environment where modules can be linked together to prepare data for integrated modelling or to simulate dynamic urban environments. It should be an easy to use, fast and memory efficient and flexible tool. DynaMind is freely available as open source software using the GPL license. To show the potential of DynaMind a simple simulation is set up where the urban development is linked with an infrastructure adaptation module to update the impervious area. The results are used in the SWMM module to assess the performance. This model can be used to investigate the impacts of urban development on an existing combined sewer network.

METHODS

To describe an urban environment raster and vector data are used in DynaMind. This data are modified by modules. A module is a small program in itself, it can receive data via in-ports and the results of the module are sent to its out-ports. To describe the dynamics in an urban environment, several modules are linked together to enable complex simulations. DynaMind has been designed as a fast, memory efficient and easy to use modelling environment. Also the development of new modules is straight forward as well as the exchange of modules.

Figure 1. DynaMind Component Diagram

DynaMind is split up in several components to guarantee a strong and robust code base which is easy to adapt in the future (see figure 1). The heart of the software is the DynaMind Core that provides a fast and memory efficient environment to set up and execute simulations. DynaMind comes already with a number of modules; basic modules
like import and export of shapefiles, to plot raster and vector data or a generic cellular automate; complex modules like modules to analyse networks, to auto design combined sewer systems and modules that integrate external simulation tools like SWMM [1] for the hydraulic performance assessment of drainage networks or UrbanSim [6] to simulate the complex dynamics in an urban environment. DynaMind provides easy to use interfaces to develop new modules in C++ or Python. The Modules are dynamically loaded during runtime from external libraries. To make DynaMind usable for non-programmers a graphical user interface has been developed that provides an easy way to create and execute simulations.

**Modules and Groups**

A Module is a small program (interface class) that is used to modify and or create raster and vector data. It consists of an initialisation and a run method (see figure 2). Within the initialisation method data (raster or vector data) in and output as well as parameters are defined. Supported parameters are boolean, int, double and strings. The DynaMind Core provides methods to modify these parameters during runtime. In the DynaMind GUI standard input dialogues are created to set module parameters at runtime (see figure 2)

```python
class WhiteNoise(Module):
    def init(self):
        Module.init(self)
        self.Height = 200
        self.Width = 200
        self.Cellsizes = 20
        self.RasterData = RasterData()
        self.addParameter(self, "Height", VDBe2.LONG)
        self.addParameter(self, "Width", VDBe2.LONG)
        self.addParameter(self, "Cellsizes", VDBe2.DOUBLE)
        self.addData(self, "RasterData", VDBe2.RASTERDATA_DATA)

    def run(self):
        ...
```

Figure 2. Module Sample Code in Python and Standard GUI

Modules can be lumped together in groups. A group can contain modules as well as groups and can be executed repeatedly - like a for-loop in a programming language. Groups can be used like modules (see figure 4).

**Data management**

Vector data are used to describe a variety of different objects like pipes in a sewer network or houses in a city. A vector data system has been developed that can be used for a variety of different objects that describe the geometry and/or attributes of urban environments like sewer networks, single family houses or households. The simplest object in the vector data system is a component. The component object contains a vector of attributes (double or string). Derived from components are nodes, lines and the system objects. A system consists of a vector of nodes, lines and subsystems and is used to describe complex objects.
Every object is identified by a unique identifier that is used to link objects together e.g. to assign a household (component) to a house (system).

Between modules complex data sets in form of raster or vector data describing the urban environment are exchanged. Therefore a fast and efficient data management system is required. The data are managed by the core and modules only hold references to them. This guarantees that each data set is stored only once during a simulation. The references are managed by the data management system and are updated before a module is executed. This method avoids that huge amounts of data are copied between different modules.

If a module adds or modifies an object in a data set, a new state of that data set is created. A data set state only contains references to objects that are stored in a data base. If a new object is added to the data set, it is added to the data base and the reference is added to the state. If an already existing object is modified, the reference in the new state is changed to point to the modified object. The states are representing some kind of content versioning system to analyse the development of a data set. Furthermore saving all states during a simulation can help to debug a simulation.

The data management system is connected via an interface to a database. At the moment the data is kept in the main memory but in a later step a SQL backend will be developed to handle huge data sets. The main memory will then be used for caching data stored in a SQL data base.

Simulations
Simulations consist of several linked modules. Before a module can be added to a simulation the module needs to be registered in the simulation environment. Therefore interfaces are provided by the DynaMind Core that dynamically load modules (implemented in C++ or Python) from external libraries. When all modules are linked together a simulation can be executed.

Since all modules are linked together the data flow in the simulation is defined and modules can be if possible executed in parallel. As a first step the following simple parallelization strategy has been implemented: within a group of modules the simulation looks for modules that don’t need data from outside (modules with no in-port). These modules are executed in a thread. After a module has been executed, the program checks if all connections of the next downstream modules are satisfied and if so the module is executed in a new thread.

To access generated data during the runtime of the simulation, a data observer can be registered within the simulation. With the help of simulation observers a dynamic adaption of simulations during the runtime is possible. E.g. modifying parameters in downstream modules or adding new modules to the simulation.

Graphical User Interface
On top of the DynaMind core a graphical user interface has been developed. The GUI supports the user by providing an easy way to create and modify simulations. As shown in figure 3 the available modules are shown on the left hand side. By drag and drop modules
or groups from the list of available modules to the right side of the GUI a simulation workflow can be defined. The modules are then represented as boxes and could be seen as an atomic working task within the simulation. Modules are linked together by drawing the line between in and out ports which represent the data flow. By clicking on a module a dialog box appears to set module parameters. The GUI also provides a basic viewer for raster and vector data. A freely available and multi-platform Python text editor called Editra (http://editra.org/) has been integrated in the DynaMind GUI. Therefore new modules can be developed within the GUI.

Figure 3. DynaMind GUI

RESULTS AND DISCUSSION

To demonstrate how DynaMind can be used a simple example of an integrated model is shown in figure 4. Aim of this model is to link an urban environment with a drainage system to investigate the effects of urban development on an existing combined sewer network. For this example only basic modules that come with the DynaMind tool are used.

Before the urban system (urban environment and its water infrastructure) can be evolved the initial system which represents a particular point in time needs to be set up. In this example the urban environment is based on raster maps for land use and population. For the sewer network and the catchments (linked with inlet points of the sewer network) vector data are used. For the initial system we import the data by using the basic modules Import Shapefile and Import Rasterdata grouped together in the Initial Environment.

The Dynamic Environment evolves the urban system one year in the future. It is split up into three major parts, Urban Development, Infrastructure Adaptation and SWMM. For the Urban Development we use a cellular automata model as described in Sitzenfrei et
To set up the model in DynaMind several *Cellular Automata (CA)* modules are linked together. Next we adopt our urban drainage system. In this example we update the impervious area of the catchment. Based on the rules presented in Sitzenfrei et al. [3] a CA is used to generate the impervious area based on land use and population. To update the information in the catchment we use the *Append Data (AD)* module. This module intersects the catchment shape with the raster data of the impervious area, calculates the median value and appends/modifies the attributes stored in the catchment. The performance of the combined sewer system is assessed with the *SWMM* module. This module exports the combined sewer system to the hydraulic solver SWMM [1] and appends the results of the hydraulic simulation to the combined sewer system.

---

**Figure 4. Example Simulation of an Integrated Urban Environment**

The three modules evolve the urban system one year. To evolve the urban environment another year the results of the current simulation are used as input for the *Dynamic Environment*. (link out port – in port). To evolve the urban system 20 years in the future the *Dynamic Environment* is repeated 20 times.
In Urich et al. [5] DynaMind has been successfully applied to set up a complex integrated environment to test the applicability of infiltration trench system as adaptation strategy for urban drainage networks. Therefore UrbanSim (Waddell et al. [6]) has been integrated as Urban Development model. UrbanSim simulates the complex interactions within an urban environment on household level. The Infrastructure Adaptation (IA) module has been enhanced by algorithms that extend the existing network to connect new build up areas and to place infiltration trench systems. As shown in figure 5 the simulation is based on the same structure as the simple example.

![Figure 5. Complex Integrated Model (Urich et al. [5]) and Results](image)

The applicability of DynaMind has also been successfully demonstrated in several scientific projects that enable the simulation of dynamic integrated urban environments like PowerVIBe, DynaVIBe and DAnCE4Water.

**CONCLUSION AND OUTLOOK**

This paper presents the design of the DynaMind framework and graphical user interface for integrated modelling of urban environment their infrastructure. Aim of DynaMind is to provide a software tool where modules, modifying the urban environment, are linked together to a simulation workflow to simulate dynamic urban environments.
DynaMind is split up in several components. A small, fast and efficient core provides interfaces to dynamically load modules from external libraries (implemented in C++ or Python) and to set up and execute simulations. The data to describe the urban environment (raster and vector data) are managed by the core. Based on the core a graphical user interface has been developed to provide the user with a simple tool to create and modify simulations.

To show the potential of DynaMind a simulation setup for a simple integrated environment is shown that can be evolved 20 years in the future. The case study can be used to investigate the effects of urban development on an existing combined sewer system. In this simulation the urban development is linked with a simple infrastructure adaptation module to update the impervious area. This is used in the SWMM model to assess the performance.

The applicability of DynaMind has also been successfully demonstrated in several scientific projects that enable the simulation of dynamic integrated urban environments like PowerVIBe, DynaVIBe and DAnCE4Water.

ACKNOWLEDGMENTS

This work was funded by Austrian Climate and Energy fund within the Energy 2020 program (project PowerVIBe). The authors gratefully acknowledge the financial support.

REFERENCES


