Our Land, Our Future
Tō tātou whenua, mō āpōpō
An Interoperable Low-Code Modelling Framework for Integrated Spatial Modelling

Alex Herzig, Jan Zoerner, John Dymond, Hugh Smith, Chris Phillips
Overview

• Smarter Targeting of Erosion Control Research Programme
• BMI – Basic Model Interface Interoperability Standard
• Model Integration Platform - LUMASS
  - Integration of LUMASS models into a BMI composite model
  - Integration of BMI-compliant models into LUMASS composite models
Smarter Targeting of Erosion Control

2018-2023 funded by NZ Ministry of Business, Innovation and Employment (MBIE)

• **Event-Scale Soil Erosion Modelling**
  – High spatial detail so that erosion mitigation can be modelled
  – Temporal variation so that sediment and water quality can be modelled

• **In-Stream Sediment Transport**
  – Temporal (hourly) modelling of suspended sediment, deposition, and re-suspension so that water clarity can be assessed

Research Aim 1.3 - Modelling
Smarter Targeting of Erosion Control

Benefits of adding BMI coupling capabilities to LUMASS

• Seamless integration of 'external' model components
• Reduced component development effort through
  – provision of data read/write and other processing components
  – provision of sequential processing
  – provision of multi-threaded processing (bmi-c++)
  – Provision of fine-grained data provenance tracking (PROV-N)
• Simplification of multi-scale model development through
  – provision of hierarchical (processing) workflow engine
  – visual development (coupling) environment
BMI – Basic Model Interface

- Introduced by Peckham et al. (2013)\(^1\)
- Set of functions\(^2\) for
  - Model control
  - Model information
  - Variable information, getter and setter
  - Time
  - Model grid functions

Land Use Management Support System - LUMASS

How does the system work?

How does it react to management?
- Spatially explicit system dynamics modelling
- System understanding
- Impact assessment

What do we do?

Where do we do it?
- Optimal spatial resource/land-use allocation
- Land-use development scenarios
- Limits testing
- Resource-use efficiency

LUMASS modelling & optimisation
https://bitbucket.org/landcareresearch/lumass
Mapping Spatial System Dynamics Modelling
LUMASS Processing Pipelines

support
- hierarchical
- sequential
- parallel
- iterable
- multi-dimensional

array processing and SQLite database processing

Basic model mechanics: https://www.youtube.com/watch?v=1DN9LcyT9ic
Spatial DaisyWorld: https://www.youtube.com/watch?v=aN2H-izUyCE
LUMASS to C-BMI

LUMASS model .lmx

ModelConfig .yaml

loads

loads

loads

loads

fine-grained control of model components

instantiate configuration execution data exchange

LumassBMI. dll/so

initialize()

update()

finalize()

communication

loads

LUMASS Engine

BMI application
C++ BMI to LUMASS

ImageReader Component

BMIModel Component

ProcessObject

ModelBMI .so/.dll

ModelConfig .yaml

Processing Pipeline

Instantiation Configuration
- bmi::BMI*
- streamable
- threadable

communication

initialize()
finalize()
update()

loads

parses

streamable: true
threadable: true
Python BMI to LUMASS

Process Component

BMIModel Component

Instantiation Configuration
- bmi::BMI*
- streamable
- threadable

ProcessObject

BMIModelFilter

ProcessObject

Processing Pipeline

LUMASS PythonBMI .so/.dll

ModelConfig .yaml

PythonBMI .py

C++ <-> Python

imports calls

data exchange

multi-threading needs to happen here

streamable: true
threadable: false

initialize()
finalize()
update()
# configuration for LUMASS' BMIModel component

**LumassBMICConfig:**

```
# bmi interface type <bmi-cxx | bmi-python>
type: bmi-python
# path to bmi library
#   python: path to *.py module file
#     e.g.: /home/python/watyieldbmi.py
#   native: path to *.dll/*.so
path: /home/alex/garage/python/watyield/bmi/watyieldbmi.py
# name of class/module, implementing the BMI interface
#   python: name of python module / class
#       e.g.: WatYieldBMI
#   native: name of bmi::Bmi subclass
name: WatYieldBMI
# whether or not the model component produces and output that can
# be fetched by a downstream component, i.e. the component doesn't
# write its output
issink: false
# whether the component's implemented algorithm supports streaming,
# i.e. the sequential processing of parts of the input array;
# conceptually, algorithms working on a single pixel, local, or
# focal neighbourhoods are streamable;
# if issink == true, the component becomes executable and will be
# called by the lumas model controller depending on the time level
# of its host BMIModel component
streamable: true
# whether or not the component is threadable, i.e. can be called
# safely from multiple threads of the processing pipeline;
# note: for python-bmi we assure that any threading is done
# within the component, e.g. using numba, since the python interpreter
# cannot be called safely from multiple threads
threadable: false
```
LUMASS
modelling & optimisation
https://bitbucket.org/landcareresearch/lumass