Improving Hydraulic Model Compatibility and Encouraging Multi-Platform Collaborative Innovation

Goals

- Improve compatibility between hydraulic modelling platforms while demonstrating and maintaining consistency of hydraulic simulation results.
- Reducing the duplication of work building tools for specific sets of APIs.
- Building additional APIs for static analysis (without simulation requirement/penalty).
- Building standard APIs to OFWAT specification so water company model requirements can be tested on a level playing field.
- Embracing open source for fast development life cycle and developer communities.
- Encouraging competition between software vendors.

Brief history of Sewage Hydraulic Simulation software

Hydraulic modelling of sewers in the UK was first done in the 1970s prior to privatisation of the water industry. Back then software named WASSP-SIM was used to perform hydraulic modelling on small networks of ~240 pipes. This software was designed by the government research departments for use within local authorities and became the standard for all local authority hydraulic models.

The Hydraulic Research department in Wallingford was later privatised to “HR Wallingford” who later sold the software to Innovyze. WASSP is the ancestor of InfoWorks ICM, which is the most used hydraulic simulation and analysis software amongst the British water industries.

InfoWorks ICM implements an API named IExchange. The IExchange API can, at current, only be ran via the Ruby programming language. To this day the main advantage InfoWorks boasts is its ability to have an interlaced 1D-2D model out of the box.

There are also a number of other software packages which allow the building and running of 1D (and 2d) hydraulic models. These software packages include:

SWMM – an open source hydraulic modelling package – 1D only (although 2D can be bodged on top using interconnected network of 1D channels). Also, it isn’t strictly designed for waste, however with inflows it can be used for waste. – Offers an API named PySWMM.

TUFLOW – a closed source proprietary (but cheaper than InfoWorks) software package which offers 1D, 2D and 3D interlaced hydraulic modelling. TUFLOW is strictly a computation engine and has no UI. – Due to its open format specification TUFLOW has no dedicated APIs but relies mostly on communities to build APIs for them. E.G. CrayFish QGIS package and TUPython.

Today hydraulic models are a critical part of our capital delivery, risk prioritisation and incident verification processes.

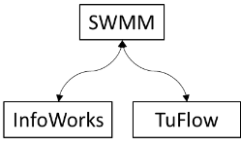
Problem Statements

1. Non-existent Interoperability between InfoWorks, SWMM and TUFLOW software packages. Leading to a lack of competition, and lack of competition between UK hydraulic modelling vendors (leading to significantly higher license costs (175% increase with Autodesk takeover)).
2. API Inconsistencies:
    a. InfoWorks APIs focus on project housekeeping, model data import and extraction, and running simulations and simulation data extraction.
    b. TUPython focuses on simulation data extraction.
    c. PySWMM attempts to create a standardised API interface, but only has 1 implementation, which is built for SWMM. PySWMM also allows for flexible user defined simulation behaviour.
3. APIs are not only inconsistent, but many do not include any standard analysis tools outside of simulation data extraction. I.E. Common Analysis reports like RPA, Network storage volume at spill, PFF at first spill, spill count etc, are common model needs but are usually counted by humans rather than being built in API tools.

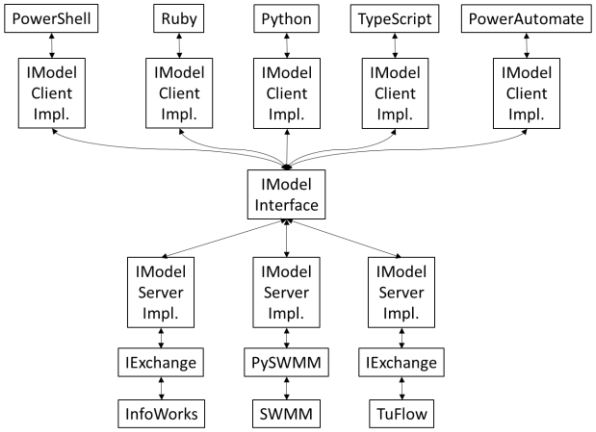
These problems are industry wide and therefore perfect candidates for collaboration.

Long Term Solutions Statements

1. Build a set of Open Source interop libraries for InfoWorks, SWMM and TUFLOW model databases. It is likely best to take SWMM as standard (as this is already the open standard) and make software for IExchange and Tuflow which interop to and from SWMM database format:



   A key measure of success of this software will be how the simulated results differ between models in their original runtime engine and in their competing engines.
2. Make an open source unified API interface for these technologies, such that tools built upon them will work for all platforms improving compatibility, increasing competition between software vendors, and reducing duplication of tools that solve specific issues built on top of specific API implementations.
   I suggest this to be implemented as a client-server API architecture. The server implementations to be in PySWMM, IExchange and TUFLOW. And a sample client implementation in Python. *Note: Advanced feature losses are inevitable with this compatibility architecture.*



   Some similar standardised technologies have been attempted in the past for hydraulic 2d river modelling. No standardised APIs have been developed which covers all functionality across applications.
3. What happens with the 3rd problem depends largely on whether the above two projects are taken up by innovation.
   a. If #2 were implemented, then an *IModel-IModelEx interface could be* implemented in between server and client offering a larger API set for larger scale analysis both inside and outside of simulation.

Short term immediate solutions / projects

The hope is that with short term solutions / projects we might be able to step towards our long-term goals without biting off more than we can chew.

1. Open Source APIs and Reporting Tools for InfoWorks ICM

I have been developing numerous libraries for InfoWorks Ruby API which offer extended utility. These APIs could be made open source and shared with other water companies as a test bed for a full solution.

The API is only maintained on an ad-hoc basis currently, as software development and standards are not part of my job description. The APIs are currently being used by modellers within the organisation, so benefit to our own modellers will be small, unless the APIs are extensively developed by the community / collaborating business partners.

Pros:

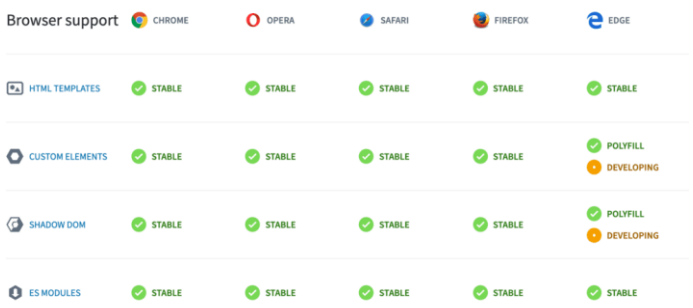
- It is something that can be initiated immediately without extensive collaboration.
- It would help push a template for the requirements of a long-term API.
- An Open Source library at least indicates that we are willing to collaborate.

Cons:

- Short term, this library will only further stifle competition and pushes the narrative that InfoWorks is the only viable solution.

2. Create a platform comparison table for software and features

Similar to https://caniuse.com/flexbox for Web Browser compatibility, a similar thing could be done for the major Hydraulic Modelling software packages/platforms and key features.



This would be a good way to establish which features are inherent, missing or where work arounds are available. E.G.

Feature	IExchange	PySWMM	RAG	Comments
RTC	Yes, system not very powerful.	Yes, full control with python.	Y	Standard API must be built to weakest software.
Variable speed pump	Yes	Yes	G	
Direct Connectivity	Yes	Yes, with workarounds	G	Implementation can easily add this feature to PySWMM.
Sewage Compatibility	Yes	Yes, with workarounds	G	Custom nodes in SWMM for population and trade, make this easy.

Note: feature here will be varied. Will be useful to have modeller's input on these as well as technology's input. E.G. Ability to fully separate UI from simulation engine is a big feature of all APIs (allowing web front-ends); Ability to generate tile service end-points is not a feature of any API (as far as I am aware). These features are just as important as modelling concepts such as VSPs, RTC, Storage nodes, 1D/2D modelling etc.

This would assist greatly in determining how much time it'd take to create a standard API, and what the limits of said API will be.

Pros:

- It is something that can be initiated immediately without extensive collaboration.
- It would assist greatly in determining how much time it'd take to create a standard API interface.
- Potential for Encouraging competition by demonstrating pitfalls of existing software if published.

Cons:

- Maintenance of table might need collaboration from software vendors.