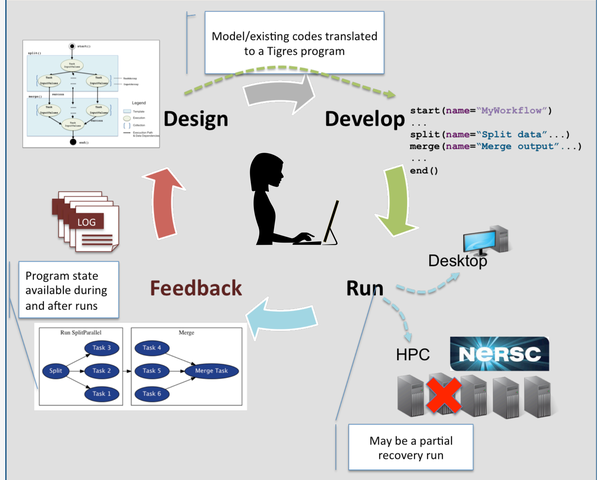
Incorporating Users in Tigres Research and Development

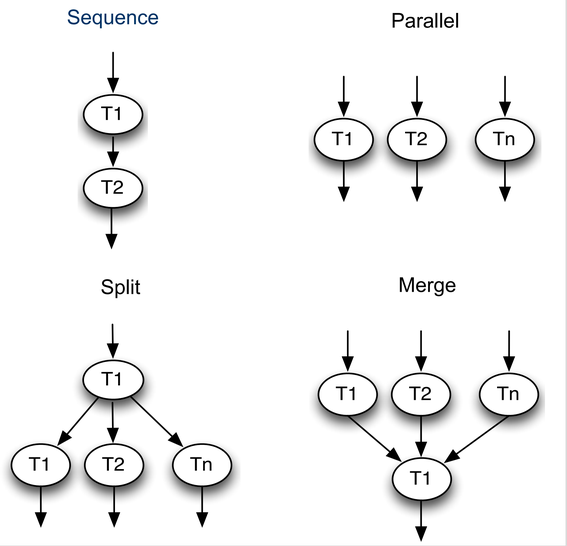
Hernisa Kacorri, Sarah Poon, Valerie Hendrix, Gary Kushner, Lavanya Ramakrishnan

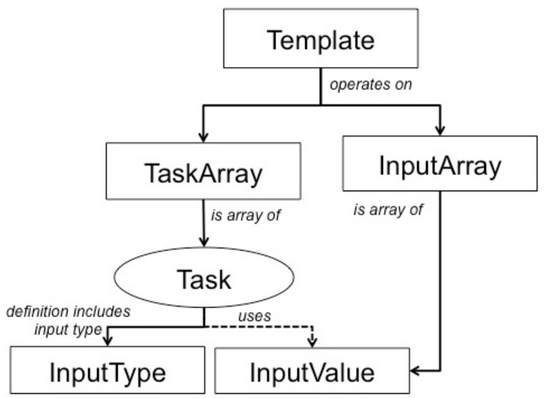
Usable Software Systems Group, Data Science and Technology, LBNL

Tigres



Workflow Composition with Templates





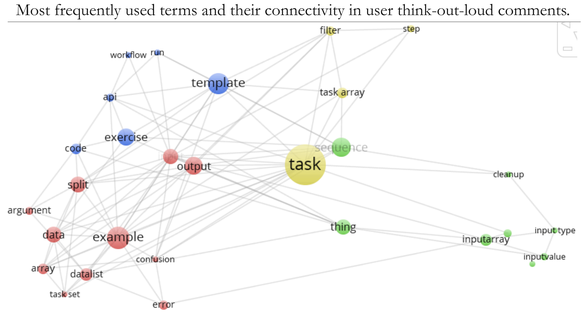
Round 1: Think Out Loud on Scenarios with Pseudo-code API

Evaluation Method: Given a prototype API documentation, a workflow example, and a wizard-of-oz python interpreter, users are asked to think out loud as they compose a workflow in Google docs.

Impact: Prioritized modifications based on user feedback:

- ❑ Adapt terms to map users' mental model
- ❑ Have a single approach for managing dependencies
- ❑ Support implicit specification of data dependencies
- ❑ Make naming for templates and tasks optional

Most frequently used terms and their connectivity in user think-out-loud comments.



Manual 'coding' of user think-out-loud comments into categories.

Workflow concept

- First *declare* your workflow and dependencies then *run*.
- Confusion between parallel and split.
- Overhead of setting up the sequence.

Stitching data with task

- A *task*, its *input*, and *output* should not be separated across the code.

PREVIOUS syntax

- Conceptually difficult.

API complexity

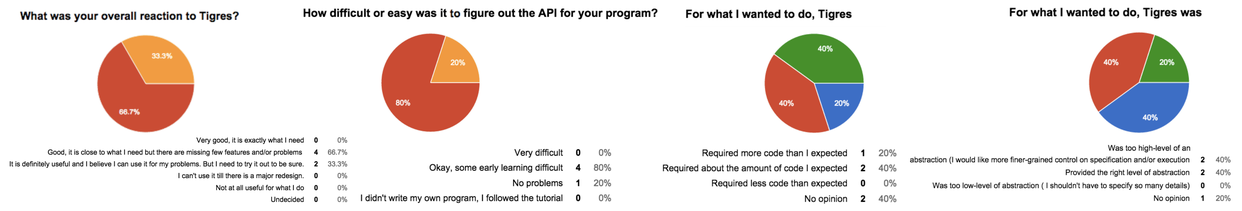
- Verbose syntax: passing strings.
- Complex: e.g. a long list of arguments encapsulated in other lists whose order is dependent with each other.

Type definition

- Extra work for keeping track of input and output types.

Round 2A: Online Questionnaire on Implemented Python API

Evaluation method: Users are asked to try Tigres on a workflow of their own and give feedback on their experience.



Manual 'coding' of user comments in the questionnaire into categories – early results with six participants.

Workflow concept

- Support for nesting, loops, and conditionals.

Stitching data with task

- Managing execution outputs of command-line applications.

PREVIOUS syntax

- Not clear and missing from examples.

New Features

- Pause/resume and recovery mode
- Real-time status on the jobs.

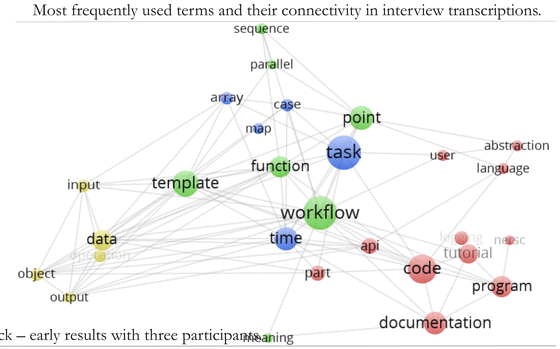
Round 2B: Post-Task Walkthrough and Open Questions Interview

Evaluation Method: Round 2a participants are further interviewed and their feedback is elicited through a post-task walkthrough, open questions, and personalized questions based on their prior responses.

Impact: Prioritized improvements based on user feedback:

- ❑ Support for nested templates
- ❑ Investigation of Tigres support and user logging for templates running in loops
- ❑ Additional examples demonstrating Tigres supported functionalities

Most frequently used terms and their connectivity in interview transcriptions.



Manual 'coding' of positive comments and additional improvements from transcribed user feedback – early results with three participants.

Participant 1

(+): clear instructions, simple model, Python, easy to use.

(-): nesting, mapping a DAG to Tigres templates, PREVIOUS syntax, Template term, NERSC, more documentation examples.

Participant 2

(+): easy to transform a Python execution flow into a task-by-task remote execution, source code structure.

(-): radical change of a template's behavior on inputs, split/merge, PREVIOUS, terms, more documentation examples & DAG, runtime estimation for NERSC.

Participant 3

(+): being able to parallelize functions, analysis and workflow in one place is powerful idea.

(-): difficult transitioning from local to NERSC, non pythonic, hard to understand what is going under the covers, overhead in syntax, more human-readable logs, more HPC-aware.

Nested Templates

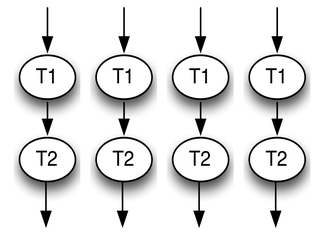
Nested templates, one of the user needs revealed in the usability sessions, is prioritized in Tigres development and we are currently considering two approaches:

- ❑ **ParallelSequential:** a new Tigres template.
- ❑ **Encapsulated Templates:** support inclusion of Tigres templates within Tigres tasks.

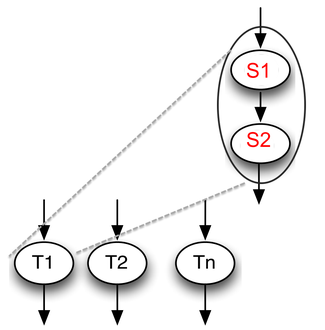
We will compare and choose one of these approaches based on three factors:

- ➢ **Supported DAGs**: Investigate the limitations on the supported workflow structure.
- ➢ **Syntax**: Compare the complexity of the API calls and boilerplate code for exemplar workflow DAGs.
- ➢ **Usability**: Conduct a user study to compare user interaction with each of the approaches on different scenarios.

ParallelSequential



Encapsulated Templates



Summary

In this work, we assess and improve the usability of Tigres, where user feedback is elicited through different evaluation methods. Initial results from empirical user data reveal Tigres' strengths, limitations, new features to be prioritized, as well as insights on methodological aspects of user evaluation that can advance the relatively new research field of API usability.

Acknowledgments

This work was funded by the Office of Science, Office of Advanced Scientific Computing Research (ASCR) of the U.S. Department of Energy under Contract Number DE-AC02-05CH11231.