AFRL-OSR-VA-TR-2015-0215

PrIMe Next Frontier Large, Multi-dimensional Data Sets

Michael Frenklach
REGENTS OF THE UNIVERSITY OF CALIFORNIA THE

07/21/2015
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ RTE
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
21-07-2015	Final Technical	05-01-2012 to 04-30-2015

4. TITLE AND SUBTITLE
(U) PRIME NEXT FRONTIER: LARGE, MULTI-DIMENSIONAL DATA SETS

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA9550-12-1-0165

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Michael Frenklach

5d. PROJECT NUMBER
2308

5e. TASK NUMBER
BX

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of California at Berkeley
Department of Mechanical Engineering
Berkeley, CA 94720-1740

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research
875 North Randolph Street
Suite 325, Room 3112
Arlington, VA 22203-1768

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release. Distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The initiative named PrIMe (for Process Informatics Model) is designed to keep track of models, model parameters, and experimental data in a global, integrated framework for the field of Combustion. It is aimed at curation of community data with the objective of collaborative development of reaction mechanisms of scientific explorations and predictive models for practical systems. The present project was a continuation of prior AFOSR grants (FA9550-08-1-0003 and FA9550-10-1-0450).
With the past funding, we focused on moving toward broader cloud-based cyber-infrastructure through distributed data and app sharing. In pursuing this goal, remote-server and browse-based technologies were employed. One of the outcomes is a prototype of a new, redesigned version of the PrIMe platform, Workflow 3.0, which is entirely browser-based and thus can run on all popular platforms.

15. SUBJECT TERMS
Modeling, combustion, global systems, web-based application, collaborative science

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Chiping Li
Unclassified	Unclassified	Unclassified	UL	24	**19b. TELEPHONE NUMBER** *(include area code)* (703) 696-8574

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

PrIMe Next Frontier:
Large, Multi-dimensional Data Sets

Michael Frenklach

Final Technical Report to AFOSR
July 21, 2015

Department of Mechanical Engineering
University of California
Berkeley, CA 94720

Table of contents

1 Introduction

The objective of this project was to sustain and extend the development of the PrIMe cyber-infrastructure (CI) for the practical use by the Combustion community. PrIMe (Process Informatics Model) is a new approach for developing predictive models of chemical reaction systems that is based on the scientific collaboratory paradigm. The primary goals of PrIMe are collecting and storing data, validating the data and quantifying uncertainties, and assembling the data into predictive models with quantified uncertainties to meet specific user requirements. The principal elements of PrIMe include: a data Warehouse which is a repository of data provided by the community, a data Library which archives community-evaluated data, and computer-based tools to process data and to assemble data into predictive models.

Optimizing combustion efficiency and understanding the mechanisms that prevent full energy utilization of fuels relies on detailed knowledge of the underlying physics and chemistry. These systems are generally complex enough that models have been used to explore the effect of different feed and reactor conditions and have been successful in optimizing fuel mixtures and combustor performance. However, the models are extremely complex and often controversial. The data, which parameterize the models and are compared to model predictions, are themselves complex and often open to interpretation. Further, they are developed by multiple labs using different technologies. To keep track of models, parameters, and data in an integrated framework has proven a necessity in the field of Combustion. The PrIMe initiative is designed to fill this need. In its scientific content, PrIMe is a *system approach* aimed at establishing the infrastructure, both scientific and CI, in support of developing *predictive* models of combustion.

The initial phase and development of PrIMe CI has focused on underlying chemical reaction models. There are several important reasons for this strategy. First, modeling of a combustion process begins with a reaction model, which determines the concentrations of chemical specifies and the heat flux, and hence it is only natural to start the new development from this founding stage. It has been our experience[1] that most disagreements between models and experiments and most controversies begin with and trace to the selected reaction model. Another factor for starting with reaction models is the fact that chemical kinetics has accumulated much needed data and the missing data can be evaluated using quantum and reaction-rate theories. And finally, the scientific underpinning of the process, also illustrating the feasibility of the approach, has been piloted by the GRI-Mech project.

The present project was a continuation of a prior AFOSR grants (FA9550-08-1-0003 and FA9550-10-1-0450, Program Manager: Dr. Julian Tishkoff), which enabled, among other things, initial development of one of the principal PrIMe components, PrIMe Workflow Application and brought this development to its stable operational version, Workflow 2.0. With the new funding, we focused primarily on moving toward broader cloud-based cyber-infrastructure through distributed data and app sharing. In pursuing this goal, we employed remote-server and browse-based technologies.

In this Report, we first outline our past-years accomplishments, and then describe the details of the initial version of the redesigned PrIMe platform, Workflow 3.0.

2 Past Three Years Accomplishments and Current Status of PrIMe

The PrIMe infrastructure has the following principal elements, a Data Warehouse, Tools, and Workflow, as well as a community Portal. During the past three years we made progress in the following areas.

2.1 PrIMe Portal

There are currently over 500 registered users.

2.2 PrIMe Data Warehouse

The PrIMe Data Warehouse has been populated with additional data: over 500 new experimental records, mostly with coal and soot data

2.3 PrIMe Workflow Application

The PrIMe Workflow is a centerpiece of the PrIMe cyber-infrastructure. It links data and apps and enables the users to conduct their research activities in a "menu-driven", web-based operation. During the past three-year year, we redesigned the Workflow, which is about to be released under Version 3.0. It is built on modern cloud-based and browser-based technologies and offers a more stable operation, running on all major platforms (Windows, Apple, Android, tablets, etc.), and ability to "link" with all different apps and databases through web services. The technical details are specified below, in Section 3.

2.4 Building Predictive Models through PrIMe

The ultimate goal and purpose of PrIMe is to support development of truly predictive models. During the past three-year period, we employed the app of Bound-to-Bound Data Collaboration (B2B-DC), integrated within the PrIMe infrastructure, in the two new developments described below.

Interval prediction of molecular properties in parametrized quantum chemistry. The accurate evaluation of molecular properties lies at the core of predictive physico-chemical models. Examples can be found in current research in atmospheric, materials, energy, and combustion processes. Quantum chemistry tools provide a solid base for molecular computations. However, the most reliable quantum-chemical calculations are limited to smaller molecular systems while practical interests reside in increasingly larger molecules. A promising approach is to employ quantum-mechanical formalism with simplifications and to compensate for the latter with parameterization. All parameterized quantum-chemical methods share one problem: uncertainty in their predictions is understood in some "averaged" way, as an intrinsic value associated with the specific method, e.g., a mean average deviation over a reference data set. Employing our B2B-DC methodology, embedded into PrIMe, we proposed a new, different strategy: predicting directly the uncertainty interval for a given property of interest based on training-data uncertainties, thereby sidestepping the need for an optimum set of parameters. This work was published in *Physical Review Letters*.

Integrated data-model analysis facilitated by PrIMe instrumental model. Numerical modeling of combustion data has become a valuable tool for technology and science, yielding design and intellectual insights from "proven" physical and chemical models. The ideal proof is agreement of predictions with data from a range of reactors and conditions, but there are uncertainties in both the predictions and the data. The goal of developing uncertainty-quantified predictive models has become broadly accepted. Conventional testing of model agreement uses processed experimental data, such as species

concentrations, which are obtained from raw data like pulse counts, currents, and voltages. As a result, in addition to raw-data aleatoric (statistical) uncertainty, analyzed data acquire epistemic (systematic, nonstatistical) uncertainties introduced via data-analysis assumptions and parameters. Some data analyses may be very precise, like spatial position from the turns of a precision-screw drive, while others may be much more uncertain, such as from factor-of-two calibration factors.

Employing our B2B-DC methodology, embedded in PrIMe, we proposed a new, more-direct protocol of the analysis, by unifying uncertainties from the physico-chemical combustion-experiment model and from the data processing, itself a model. In other words, the flame simulation becomes an integral part of the analysis, which is carried out all the way to the raw experimental data.

The new approach was demonstrated with raw experimental data collected in an acetylene flame mapped with VUV-photoionization molecular-beam mass spectrometry at the Advanced Light Source facilities of Lawrence Berkeley National Laboratory by a research team led by Professor Phillip Westmoreland (of North Carolina State University). The experimental signals were modeled with a premixed laminar flat-flame code augmented with an Instrumental Model, designed to link raw signals to derived properties. The methodology enabled a rigorous comparison between model and data to quantify what is meant by "model predicts the data." The model was shown to be quantifiably consistent with the data, and hence can be further applied to predicting properties. As specific examples, the prediction of an unmeasured property, background water vapor, was quantified and error bounds for weak-signal properties, O, OH, and C_2H_3, were obtained. The sensitivity coefficients produced by the B2B-DC methodology revealed where the main effort for improving model and prediction uncertainties should be placed. The results were presented at the 35th International Symposium on Combustion.

2.5 PrIMe-enabled Collaborative Environment

As a results of collaborative efforts between us and the Combustion Group of KAUST in Saudi Arabia, an application developed by the KAUST team for flame simulations (https://cloudflame.kaust.edu.sa/) was linked with PrIMe Workflow through the PrIMe web services. In this way, a user can perform flame simulations by logging into the central Portal of PrIMe (http://primekinetics.org/), get data (a model and experimental conditions to model) from the PrIMe Data Warehouse, perform flame simulations at a remote server (CloudFlame), and get back the results for further analysis and processing by the PrIMe tools. The success of this undertaking demonstrates the growing capabilities of PrIMe: to link various data sets, private and public, large and small, with various scientific applications.

3 New PrIMe Workflow Architecture

The PrIMe Workflow Application (PWA) is a web-based application that unifies the components of PrIMe into a single interface. The purpose of this document is to describe the new PrIMe Workflow Application architecture and its internal structure. The next sections of the document describe the following aspects of the new PrIMe Workflow Application, PrIMe Workflow 3.0:

- Motivation for changing the PWA
- The new PWA User Interface description
- PWA general architecture description and functionality.

3.1 PrIMe-enabled Collaborative Environment

The increasing popularity of non-Windows OS (MAC, Android, iPad) and tablets necessitates making PrIMe Workflow Application available for these platforms. The current version of PWA, 2.0, has been implemented as a web-based .NET component and was available on the Windows platform only. Within the present project, the infrastructure of the PWA was extended for demonstration of an alternative approach, using the latest HTML5 and JavaScript technologies. This approach is aimed at an entirely Web-based application and thus will be able to be operated on any major platform. A lighter PWA will allow for a wider and broader access by the combustion community, thus enabling the shift to new scientific methods of predictive model developed fostered by PrIMe.

The present development, termed PWA 3.0, is a poof of a concept and has limited functionality. The objective of this initial development was to explore and demonstrate the new internet technologies, and test the community response. PWA 3.0 does not require any setup, runs through most popular browsers and most of the popular platforms.

3.2 Scope

The main issues implemented in designing PWA 3.0:

- REST Web Services layer for the PrIMe Workflow Application 3.0
- Cloning and porting to back-end (PrIMe server) the workflow executor and temporary data storage for workflow calculation results
- Implementation of the Workflow Editor with HTML5 and JavaScript technologies

It is pertinent to note that the previous version, PWA 2.0, is still available and fully operational.

3.3 PWA 3.0 Architecture

Figure 1 shows the previous version of architecture. This architecture is used with PWA 2.0, which was

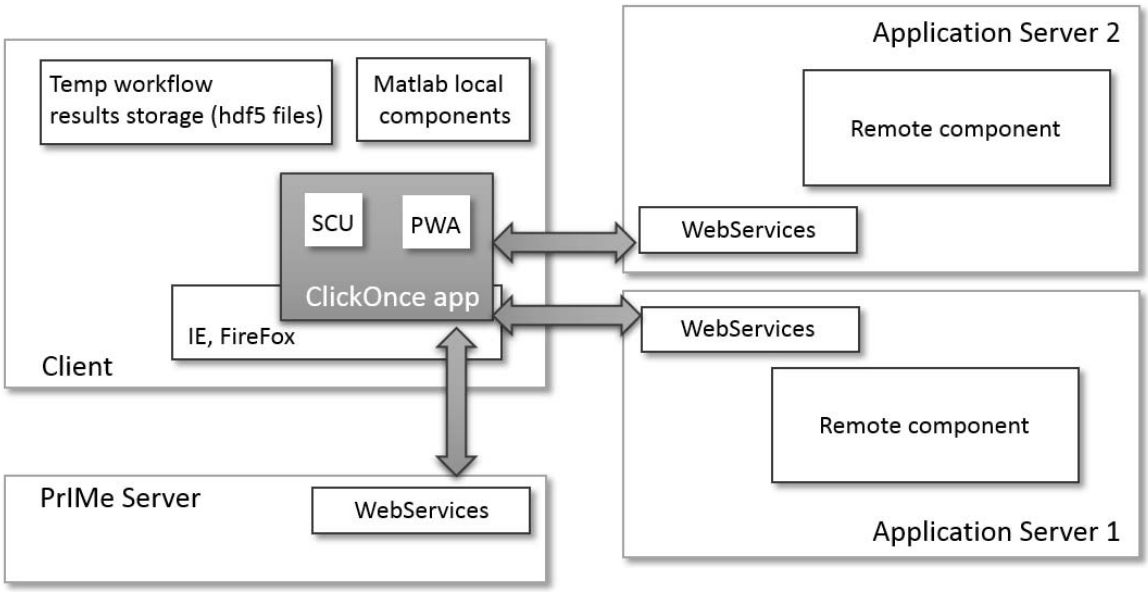

Figure 1. Current PWA 2.0 architecture

built as the client-side ClickOnce application and runs in the Internet Explorer. The main functionality of the PWA 2.0 is implemented on the client side of the system. All the remote components that are included into the workflow interact via SOAP Web Services from this application. The execution process is managed and performed on the client side and interaction with remote components is accomplished from the PWA 2.0 application. The results obtained with each workflow component are stored on the client-side as well. A project folder is used to store all the results (HDF5) files produced during the execution. PWA 2.0 runs in Internet Explorer.

The new PWA 3.0 is implemented with HTML5/JS technologies, and a number of changes in the architecture were required. These changes were motivated by the fact that the HTML5/JS-based front-end applications are running inside of browsers, and the latter have no access to the client-computer resources. Therefore, the new PWA 3.0 has to have distributed architecture:

- Front-end—to implement User Interface for performing all user manipulations and management to create, edit, and run workflow projects;
- Back-end—to implement in part the PWA 2.0 functionality that is responsible for execution and storage of the Workflow projects, as well as communication with remote components.

The front-end is coded in HTML5/JS and hence no temporary data can be uploaded to the front-end during execution. Such approach to implementation of the client-side application makes it impossible to use local Matlab or any other components except the JS components. Therefore, to make the PWA 3.0 fully functional requires changes to be made in the PWA 2.0 component library.

The new infrastructure is shown in Figure 2, where the implemented changes to the infrastructure are colored in green.

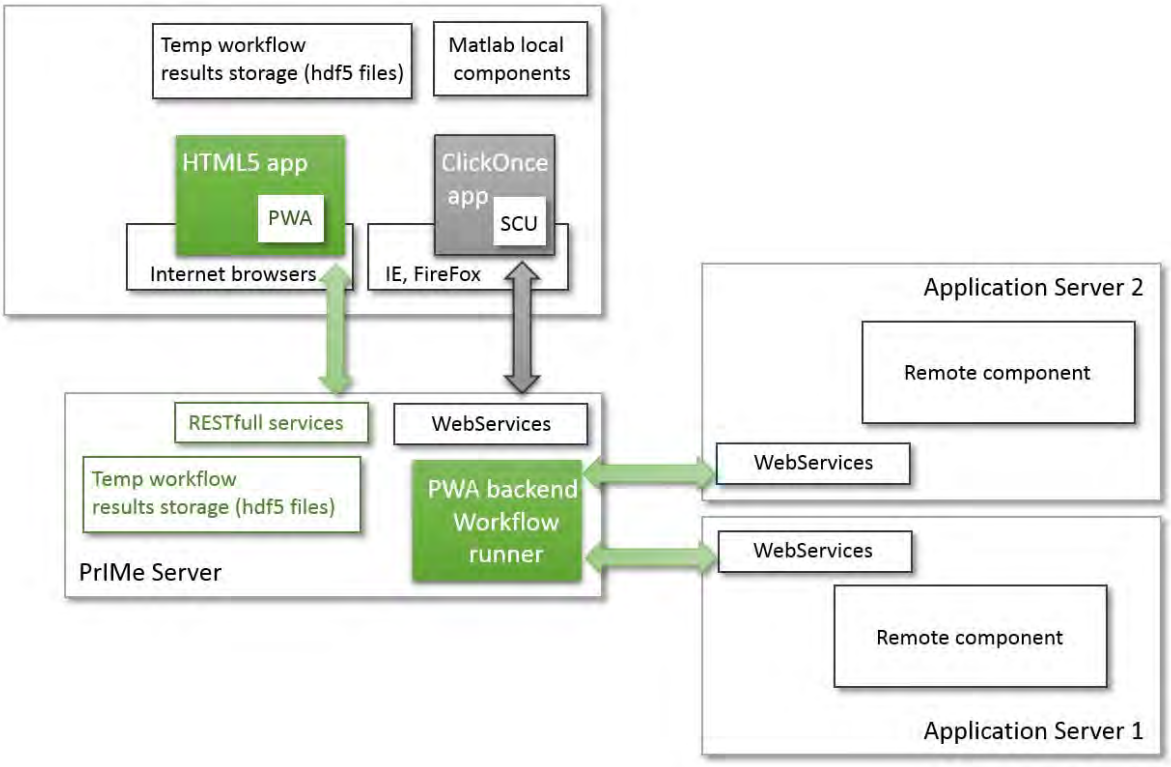

Figure 2. PWA 3.0 architecture

The back-end of the new PWA 3.0 application is implemented with .NET technologies and provides REST API to interact with the front-end. In this version of the architecture, all the core functionality of the PWA was migrated to server-side, and hence a more powerful server is required to run PWA 3.0. All the information and temporary data are stored on the back-end server as well.

3.4 PWA 3.0 Back-end

The back-end part of PWA 3.0 is implemented with the .NET technologies to be compatible with previous versions. The technology stack includes the following tools:

- .NET 4.5—the actual version of the .NET framework.
- OWIN and Katana—flexible set of components for building and hosting Open Web Interface for .NET (OWIN)-based web applications.
- ASP.NET MVC Framework—open-source web-application framework that implements the model-view-controller (MVC) pattern.
- ASP.NET Web API Framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices.
- ASP.NET signalR—simplifies the process of adding real-time web functionality to applications. Real-time web functionality is the ability to have server code push content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data.
- ASP.NET Entity Framework—object-relational mapper that enables .NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write.
- Autofac—addictive Inversion of Control container for .NET 4.5.

3.4.1 Components

The back-end components are shown at Figure 3. The Workflow execution components include all the functionality for execution of the workflow submitted by the user via front-end User Interface (UI).

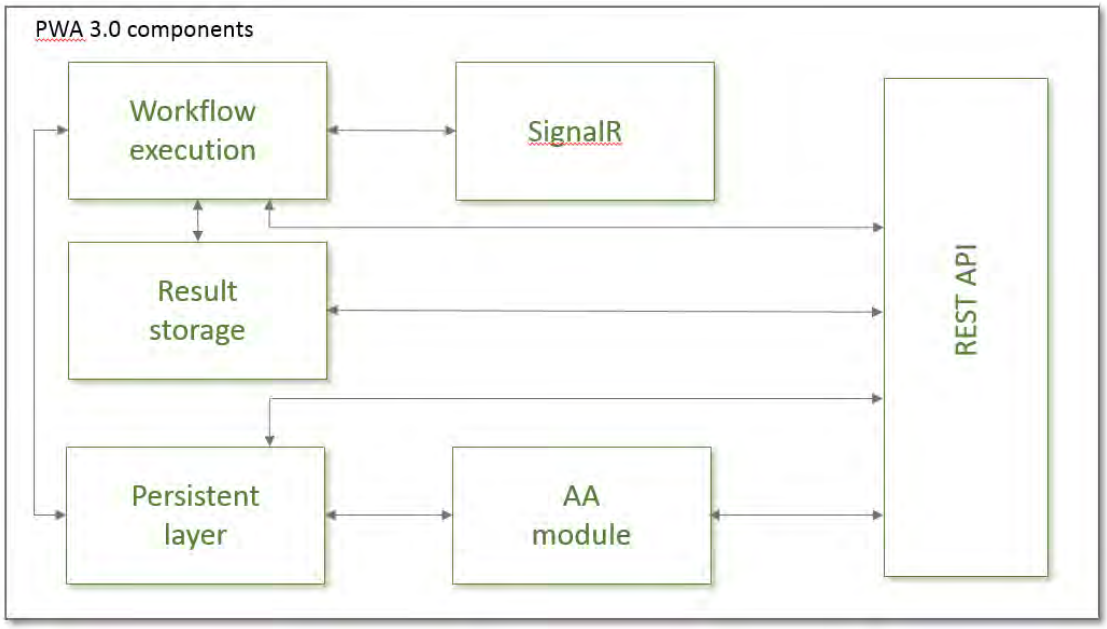

Figure 3. Components of the PWA 3.0 back-end

The **Workflow execution** implements all the logic required for the execution of the Workflow project components, assigns them states, manages processing by giving signals to the component that should be calculated, and informs other components about their states. It receives the Workflow project structure, analyzes dependencies between the components, and then performs calculation for all components, one by one; if the component is local, it suspends back-end calculations and sends start signal to the front-end saying that the given local component should be executed. Also, the Workflow execution component passes the result data of one Workflow component to another.

The **SignalR** component provides two-directional communication between the front-end and the back-end of PWA 3.0 via a socket. When user starts the Workflow execution process by clicking button "Run" at the UI, the web-socket between the front-end instance and the back-end is created and a connection is established. This connection facilitates the exchange of execution states and temporary data. The front-end is listening to the socket and waiting for signals from the Workflow execution component to change the component state or to execute one of the local components.

The **Result-storage** component is intended for organizing and storing the temporary data that are generated during workflow execution. During the workflow execution, each component produces data that are passed to the input of the next component. All the temporary data are stored in the database, in table FileResults. The data are stored in the binary XML format, HDF5, and linked to an instance of the Workflow project component. The HDF5 data could be requested via UI employing REST API.

The **Persistent layer** is the data access and storage component. A database is used for storage, and its structure is described below.

The **AA module** checks the login username and password to identify that the user is registered in PrIMe, and if so, gives access to PWA 3.0. This module interacts with the PrIMe Portal database (DB) that stores user information. The REST API component implements all the functionality for providing integration between the front-end and back-end.

The **REST API** module is a web interface for retrieving data by the front-end from the back-end. The REST API allows to make method calls from JavaScript to the back-end for getting a list of projects, components, and sending projects to the back-end for saving, updating, and launching. A list of REST API calls is provided below.

3.4.2 Entities

Figure 4 is a diagram that represents the datatypes of objects/instances that are used within the PWA 3.0 back-end implementation. These objects are:

AttributePredefinedVaules—an instance with a predefined value for a given attribute; mapped to the database table "AttributePredefinedValues"

ComponentGroup—class for objects that contain data and methods for a group of components; mapped to the database table "ComponentGroup"

ExecutionInfo—an instance of this class represents data about the workflow execution detail; mapped to the database table "ExecutionInfoes"

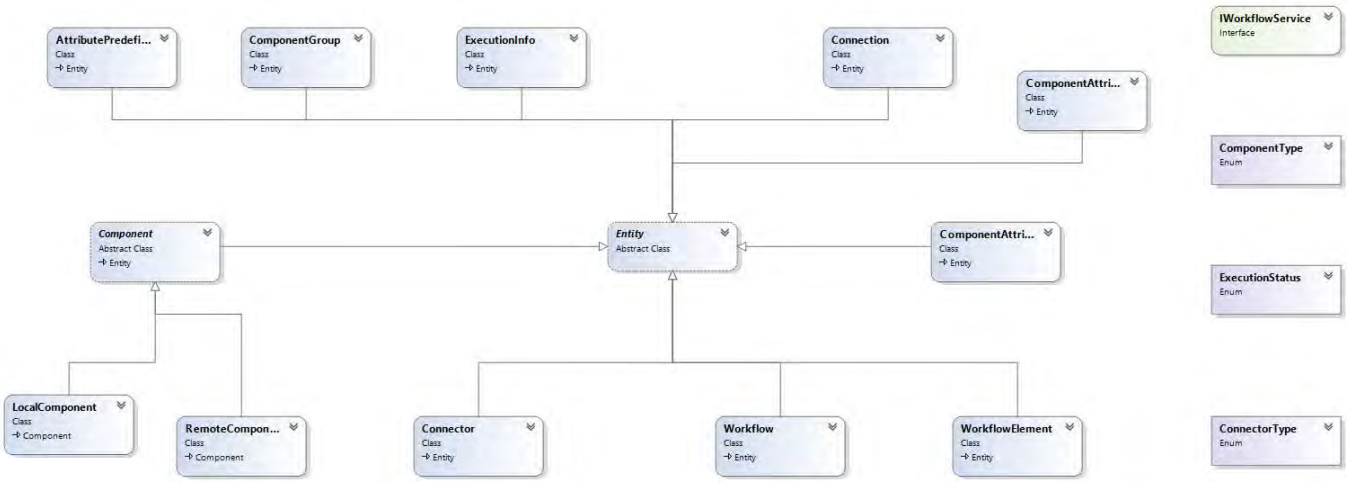

Figure 4. The Entity diagram for the PWA 3.0 back-end

Connection—an instance of this class is created for each link between two components in the Workflow project; mapped to the database table "Connections"
ComponentAttribute— an instance of this class is created for each attribute of the each component; it contains details about the attribute; mapped to the database table "ComponentAttribute"

Component—an instance is created for every component of the Workflow project; mapped to the "Components" table. A component could be local (**LocalComponent**) or remote (**RemoteComponent**), and this specification is stored in the field "Type" of the database table "Components"

Connector—an object for input/output connector of a component instance; mapped to the database table "Connector"

Workflow—a workflow project object, built of Component, Connection, and Connector objects; mapped to the database table "Workflow"

WorkflowElement—an instance for each element of the workflow that contains information about what component this element represents, where it is located at the layout, and its state; mapped to the database table "WorkflowElements"

ComponentAttribute—an instance with the information about possible component attributes; mapped to the database table "ComponentAttributes"

ConnectorType—enum of possible connection types: Input, Output

ExecutionStatus—enum of possible states that are available when workflow project is executed

ComponentType—enum of possible component types, local and remote. Local the one which is executed on the user's side (front-end) as a JS script.

IWokflowService—used for providing CRUD (create, read, update, delete) operations for the Workflow project

3.5 Database Structure

The developed database (DB) structure is illustrated in Figure 5 below.

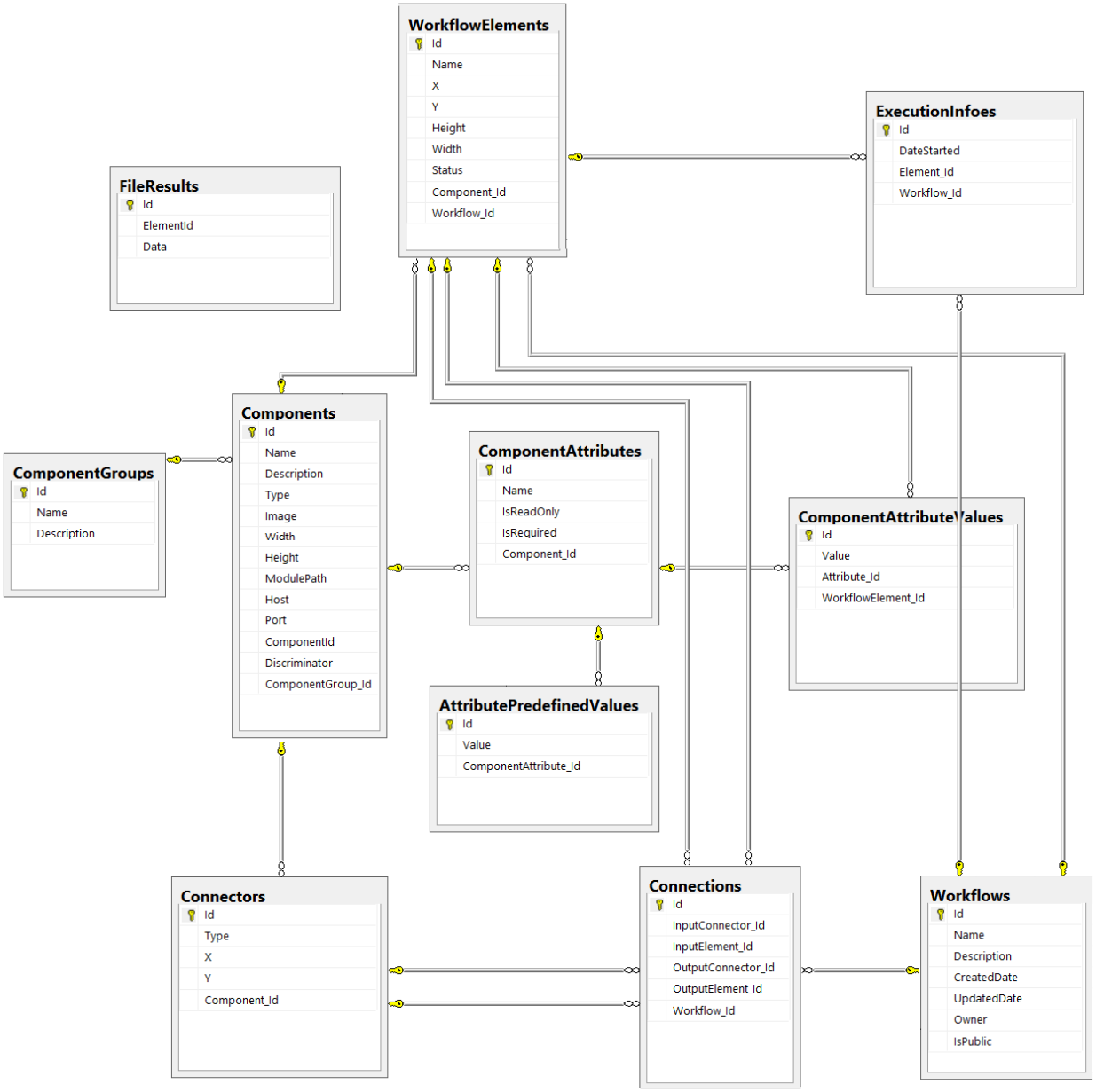

Figure 5 – PWA 3.0 database structure

The **Workflows** table used to store information about the existing Workflow projects is as following:

- Id – the unique identification of the workflow project
- Name – the name of the workflow project given by the user
- Description – the field for storing the description of the workflow
- CreatedDate – the date the Workflow project was submitted to the back-end the first time
- UpdatedDate – the date of the last changes made of the Workflow Project
- Owner – the login name for the primekinetics.org of the user who created and owns the given Workflow project
- isPublic – the identification of the Public/Private access to the Workflow project

The **ComponentGroups** table contains the information on categories of available components:

- id – the unique identification of the group/category
- Name – the name of the group/category which is displayed at UI when splitting components by groups
- Description – the description of the group for future use

The **Components** table contains all the components details that are available for building the Workflow project:

- Id – the unique identifier of the component
- Name – the name of the component which is visible at UI
- Description – the component details that can be used in future versions
- Type – the remote or local component; e.g., implemented as PWA 3.0 front-end JS component
- Image – the illustration file that should be displayed as the component icon (JPG or PNG, 60×60 pixels; other sizes are also acceptable, they will be scaled to 60×60 pixels automatically)
- Width, Height – the width/height of component used for drawing component at user interface
- ModulePath – the path to the location of the component
- Host, Port – the details about the instance on which the component should be executed
- ComponentGroup_Id – the id of the group/category to which the component belongs

The **ComponentAttributes** table contains all defined attributes of the component:

- Id – the unique identification of the attribute
- Name – the name of the attribute which is displayed at UI
- isReadOnly – defines whether the attribute only declarative or can be edited
- isRequired – whether the attribute is mandatory or not
- component_Id – defines the parent component of the attribute

The **AttributePredefineValues** table contains the default values of the attributes:

- id – the unique identification of the default value
- Value – the value which should be used as the default
- ComponentAttribute_Id – the id of the component for which this value should be used

The **WorkflowElements** table contains all the elements, which are used for the given Workflow Project:

- Id – the unique identification of the component instance which is used in the workflow
- Name – the name of the component used in the given workflow project
- X,Y – the position of the component in the Workflow canvas at the Workflow editor page
- Height/Widths – the size of the component used for drawing the component in canvas
- Status – component status at a given moment; also used for displaying the status in UI
- Component_id – defines what component this is
- Workflow_id – the parent workflow project of the component instance

The **ComponentAttributeValues** table contains the attribute values of the given instance of the component in the workflow project:

- Id – the unique identification of the stored value
- Value – the value that is stored
- Attribute_Id – the Id of the attribute for which the value is stored
- WorkflowElement_Id – the Id of the workflow instance of the component for which the value is applicable

The **Connectors** table contains the information about inputs and outputs for each used component:

- Id – the unique identification of the connector
- Type – defines whether this is an input or output connector
- X,Y – position used for drawing the connector in the Workflow canvas
- Component_Id – the parent component instance of the connector

The **Connections** table stores all the connections between elements for each created workflow projects:

- Id – the unique identification of the connector
- InputConnector_Id – the id of the input connector of the connection
- InputElement_Id – the id of the instance of the input component of the connection
- OutputConnector_Id– the id of the output connector of the connection
- OutputElement_Id – the id of the instance of the output component of the connection
- Workflow_Id – the id of the parent Workflow project of the connection

The **ExecutionInfoes** table stores temporary data used by the back-end for execution of the workflow project:

- DateStarted – the start date and time of the execution
- Element_Id – the starting element of the execution
- Workflow_Id – the parent workflow project of the element

The **FileResults** table serves as temporary storage for the data produced after execution of each element of the workflow:

- ElementId – specifies the element that produced the result data
- Data – the results of the execution

3.6 PWA 3.0 Front-end

The PWA 3.0 application was built with the Angular.js framework using jQuery and jQuery-UI, and it has the classical structure of the Angular-based applications. The Angular framework was chosen to build MV* architecture of the web application due to its benefits for future development and extension of the PWA infrastructure. One such benefit, for example, is the two-ways data-binding for settings workflow components, which is more efficient than employing DOM events.

The main structural components of the PWA 3.0 application:

- Controllers describe the logic and model for the pages, page elements, and pop-up windows:
    a) AuthCtrl – login page
    b) ControlsCtrl – the control buttons for managing the project
    c) HeaderCtrl – the top part of the page
    d) SaveCtrl – popup dialog for saving the project
    e) WorkflowCtrl – workflow editor area of the page
- Directives that describe application elements which are isolated repeatable and include the logic for direct changes in the DOM structure of the element:
    a) plumcategories – the list of components by categories
    b) plumbflow – the workflow editor area
    c) plumbinstance – components added to the workflow editor
    d) plumboptions – component parameters
    e) primeproject – project in the list of available projects
    f) primeprojects – list of available projects
- Services containing operational logic of the application and interaction with external resources:
    a) authService – authorization and stored token
    b) componentsService – components uploaded
    c) messageService – uploaded text of error messages and displayed errors
    d) metaService – upload and store information about the current project
    e) projectManagerService – upload and store information on available for-use projects
    f) runService – managing of the current project execution
    g) saveService – applying changes and update the project
    h) schemaService – project upload, storing and updating element position and state
    i) trashService – removed element and its connection with others

For navigation between pages, the Angular route provider is used. All elements of the view are stored as separate HTML files and are uploaded to the page with ng-view asynchronically.

For various purposes, several third-party libraries are used:

- Date Format 1.2.3 – to work with dates and time
- jsPlumb – render links between workflow components and processing events
- FileSaver.js – covert project and workflow to JSON format and upload to back-end

- ASP.NET SignalR – implement web-socket for listening events from back-end
- contextmenu.js – customize context menu control
- TouchPunch – implement reaction to touch-events
- md5.js – for security and encryption
- xml2json – parse XML and retrieve data from it
- ng-dialog – implement the dialog boxes

3.6.1 JS Libraries Comparison

For efficient implementation of the PWA 3.0, the workflow editor was built employing ready-to-use third-party libraries. They were selected based on comparison of important for development criteria as described in Table 1 below. The libraries that were analyzed are:

- **RaphaelJS** http://raphaeljs.com/
- **JSPlumb** https://jsplumbtoolkit.com/demo/flowchart/dom.html
- **mxGraph** https://www.jgraph.com/javascript-graph-visualization-library.html
- **Draw2D** http://www.draw2d.org/draw2d/
- **JointJS** http://www.jointjs.com/

Table 1. Comparison of JavaScript-based libraries for drawing workflow.

Criteria	**RaphaelJS**	**JSPlumb**		**mxGraph**	**Draw2D**	**JointJS**
Lightweight	✔	✔		✔	✖	✔
Fully fit the task	✖	✔		✔	✔	✔
No external dependencies	✔	✔		✔	✖	✖
AngularJS compatable	✔	✔		◐	✔	◐
Free	✔	✔		✖	✖	✔
Welldocumented	✔	✔		✖	✔	✔
Easy-to-use	✖	✔		✖	✔	✔
Size with dependencies	92kB	166kB		4.2Mb	-	221kB
License	MIT	MIT/GPL2		OEM	GPL/Com.	MPL
Chosen		✔				

The best fit for the described task was the JSPlump library. This library can work with Vanilla (built-in js-framework) as well as with JQuery.

3.6.2 JS Local Components

Each component should have a directory associated with it. This directory (folder) should be placed on the server as

[domain]/Scripts/Application/components/[componentID]/

where [domain] is the "root" directory and [componentID] is the component directory named as the ID of that component. The component is specified by the following files, placed into the [componentID] directory:

- <componentID>.js – the file with the implemented logic of the component
- <componentID>.css – the CSS-style for the component; it is not the mandatory
- the HTML code for the Dialog window; it is not the mandatory

The <componentID> should be the same as the component ID. For example, for component #45, the files should be named 45.js and 45.css.

Within the <componentID>.js file, an object with name "localComponent" should be created. This object should have callbacks for the interaction with the PWA 3.0 infrastructure.

onLoad – this callback is activated when the component is being registered in the infrastructure

Example:

```
localComponent.onLoad = function () {
    console.info("component was loaded");
}
```

onInstanceCreate – this callback is activated when the component is being added to the workflow builder

Example:

```
localComponent.onInstanceCreate = function (instance) {
    if (instance.status == 2)
        console.info('Component is ready');
}
```

onInstanceRemove – this callback is activated when the component is being removed/deleted from the workflow builder

Example:

```
localComponent.onInstanceRemove = function (instance) {
    if (instance.status == 4)
         saveChanges();
}
```

onInstanceStartProcessing – this callback is activated at the start of the execution of the component. It receives callback as the second input parameter. When the component processing is finished, this callback should be called with the status of the component "Success" or "Failed". This call will be the initiator for continuation of the workflow execution.

Example:

```
localComponent.onInstanceStartProcessing = function (instance, callback) {
    createWindow(instance.options.graphicalMode, callback)
}
```

onInstanceCompleteProcessing – this callback is activated upon completion of the execution of the component.

Example:

```
localComponent.onInstanceCompleteProcessing = function (instance) {
    if (instance.status == 4) {
         console.log('component was successfully processed');
}
```

Input parameters are supplied for each callback except for the **onLoad**:

- position of the component on the Workflow Editor
- status of the component
- values of the options
- amount of the connections

The component CSS file is uploaded automatically, but it is not a mandatory requirement.

Example of such a CSS file:

```
div.local-output-window1 {
    width: 600px;
    height: 500px;
    background-color: white;
    position: absolute;
    margin: auto;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    z-index: 6001;
    opacity: 0.9;
    box-shadow: 1px 1px 5px #444;
}
div.local-output-window1 div.header {
    margin-left: 20px;
    margin-top: 15px;
    font-size: 26px;
}
div.local-output-window1 div.close-button {
    width: 16px;
    height: 16px;
    position: absolute;
    top: 10px;
    right: 10px;
    background-image:
url('
SUhEUgAAABIAAAAPCAYAAADphp8SAAAAAXNSR0IArs4
c6QAAAAZiS0dEAP8A/wD/oL2nkwAAAAlwSFlzAAALEw
AACxMBAJqcGAAAAAd0SU1FB98CFw0bKXvu+LkAAAAZd
EVYdENvbW1lbnQAQ3JlYXRlZCB3aXRoIEdJTVBXgQ4X
AAAAlUlEQVQ4y2NgGLaAEV1AsfqAAAMDw3sGBob591s
dkpDE5zEwMCQyMDAI3m91+ICujwmL4e+hdCJUM7IhyP
IogAWL2HwkTYmK1QcYkPgwecJew+ICFEuQvUvQIByG4
TQEn4sWMTAwxGKROnK/1cEWmx4mIgxBDhMbxeoDh4ky
CM2QI1DvHEE2jFiDBNG9AaWPoMmPFAAAD7oySsf3Kl0
AAAAASUVORK5CYII=');
```

The UI of the component can be described as HTML code of the Dialog window.

Example:

```
<div class="local-window1">
    <div class="header">Input Component</div>
    <div class="content">
        <button class="apply">Success</button>
        <button class="fail">Failed</button>
    </div>
    <div class="close-button"></div>
</div>
```

Internally, in the component, jQuery-code is available; a window can be created and added. For the upload of the component, $.ajax should be used; for instance:

```
$.ajax({
      type: 'GET',
      url: '1.html',
      dataType: 'text',
      success: function(res) {
          $('body').append(res);
      }
 });
```

Important! The unique names for the component containers must be used, like “input-component1”. The CSS and jQuery selectors should take into account the parent node, like “$('div.input-component1 button#close-button')”. This can help to avoid overlapping of styles and listeners.

If a component uses external data files and/or libraries, they should be uploaded with the $.ajax as well:

```
var libraries = ['/Scripts/Application/components/45/chart.js'];
var counter = 0;
$.each(libraries, function(i, libURI) {
    $.ajax({
        type: 'GET',
        url: libURI,
        dataType: 'script',
        success: function() {
            if (++counter == libraries.length)
                createSpecWindow(callback);
        }
    });
});
```

3.6.3 User Interface (UI)

The UI of the PWA 3.0 is illustrated at the Figure 6. It is one-page web application that works within Internet browsers.

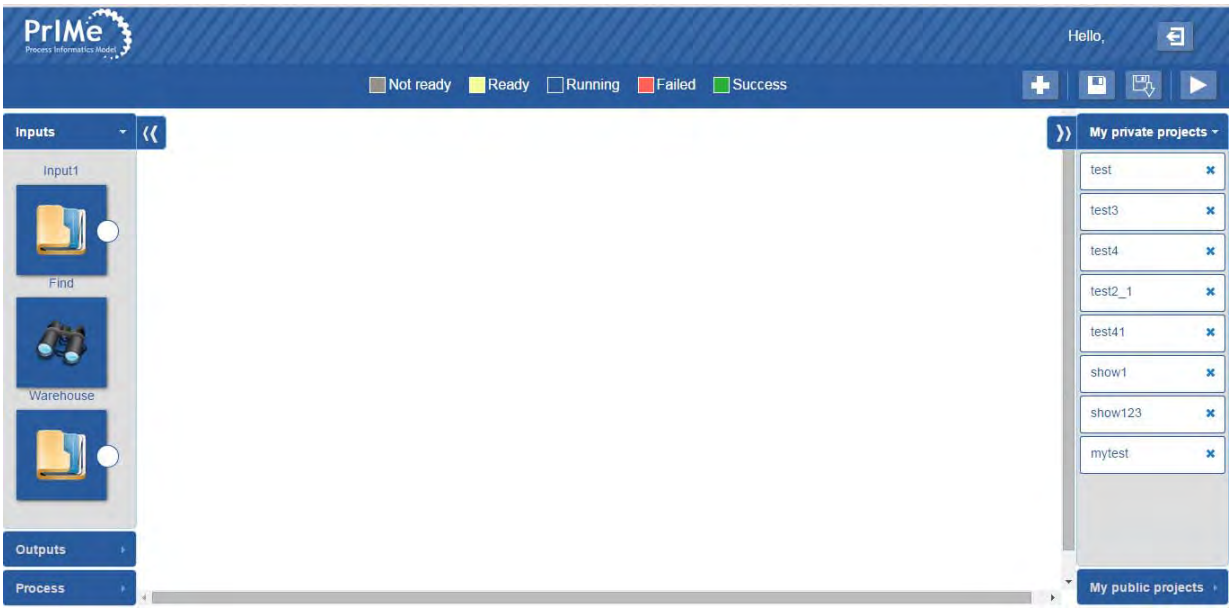

Figure 6 – The general look of the PWA 3.0 user interface

The top panel of the web page has three functional regions:

- component/workflow execution states
- management buttons group (Figure 7)
- user name and logout information

The left hand-side panel of the web page displays all the available components for building the workflow project. The components are organized into the named groups. The component panel can be hidden, which could be used for small screen devices.

The right hand-side panel of the web page lists all available Workflow projects that the user can open, update, execute, or delete. Projects are grouped into two categories: *private* – personal projects of the logged user, and *public* – projects that are available for all PrIMe users.

Figure 7. Management buttons

The management buttons, reproduced in Figure 7, perform the following actions (from left to right):

- start a new workflow project
- save changes made to the workflow project
- “save as” the workflow project
- execute the project

3.6.4 Component/Workflow Execution States

Figure 8 reproduces the top panel of the web page with the set of states that components could have during the execution and editing of the workflow.

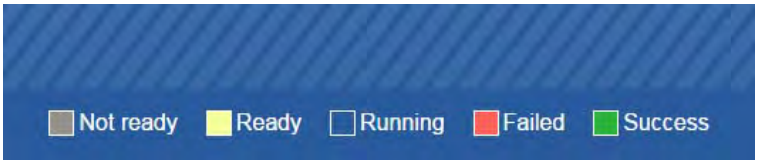

Figure 8. List of states for components

The **Not Ready** state indicates that the component has unconnected inputs or outputs; while at least one component of the workflow has such a state, the workflow project is not ready for execution (Figure 9). When all connections of the component are linked to other components, the component state changes to **Ready** (Figure 9). An attempt to execute a Workflow project with Not Ready components will generate a message asking to connect all components.

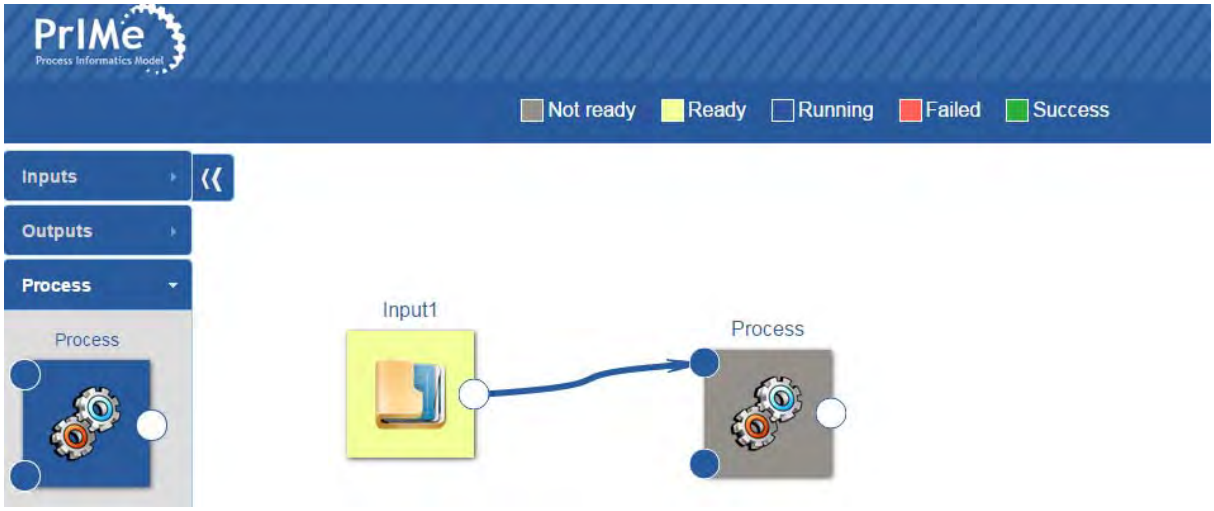

Figure 9. An illustration of a Workflow project with a Ready (Input1, colored yellow) and Not Ready component (Process, colored grey).

As the Workflow execution begins, the component being executed acquires the **Running** state, colored blue. After execution is completed, the component state can be either **Success** or **Failed**; colored green or red, respectively.

3.7 PWA 3.0 REST API

The PWA 3.0 front-end communicates with the back-end via REST API calls. This API allows to pass data about Workflow projects, execution process, and users' related data to visualize the process in UI and to manage the workflow information. The main API calls are as follows.

Authorization:

- POST /token

Submit results of executing a local component to the back-end:

- POST /api/launcher/submit/[workflow-id]/[element-id]

Download all available components from the server:

- GET /api/components

Get the list of the Workflow projects:

- GET /api/workflows/

Delete a Workflow project from the back-end:

- DELETE /api/workflows/delete/[workflow-id]

Start execution of a Workflow project:

- GET /api/launcher/launch/[workflow-id]

Set the execution status for a given component of the Workflow project:

- GET /api/launcher/setstatus/[workflow-id]/[element-id]/[status]

Get the list of Workflow components linked with a given component:

- GET /api/launcher/linkedelements/[workflow-id]/[element-id]

Update changes in the Workflow project structure/settings:

- PUT /api/workflows/update

Save the new Workflow project:

- POST /api/workflows/save

Get results of component execution:

- GET /api/workflows/[workflow-id]/[element-id]

Get the Workflow project from the back-end:

- GET /api/workflows/[workflow-id]

4 Personnel Supported

This project supported mainly the programming consultant, Michael Gutkin, graduate students William Speight and James Oreluk, along with the Principal Investigator, Professor Michael Frenklach.

5 Publications and Presentations

"Interval prediction of molecular properties in parametrized quantum chemistry," D. E. Edwards, D. Yu. Zubarev, A. Packard, W. A. Lester, Jr., and M. Frenklach, *Phys. Rev. Lett.* **112**, 253003 (2014).

"Integrated data-model analysis facilitated by an instrumental model," D. R. Yeates, W. Li, P. R. Westmoreland, W. Speight, T. Russi, A. Packard, and M. Frenklach, *Proc. Combust. Inst.* **35**, 597-605 (2015).

"PrIMe Next Frontier: Large, Multi-dimensional Data Sets," M. Frenklach, Multiagency Coordination Committee for Combustion Research (MACCCR)—5th Annual Fuels Research Review, Sandia National Laboratory, Livermore, CA, September 17-20, 2012.

"Data Needs for VVUQ in Chemical Kinetics," M. Frenklach, Planning Workshop on "Data Collection in Support of Modeling and Simulation," The National Academy of Sciences, Washington, DC, November 2, 2012 (invited).

"PrIMe: Integrated Infrastructure for Data and Analysis," M. Frenklach, session on "Collaboration: Tools and Infrastructure" of the Pacific Neighborhood Consortium (PNC) 2012 Annual Conference and Joint Meetings, University of California, Berkeley, December 7-9, 2012 (invited).

"Deterministic UQ using Rational Quadratic surrogate models: Response Surface Modeling and Inference," R. Feely, T. Russi, C. Meissen, M. Speight, A. Packard, and M. Frenklach, 14th International Conference on Numerical Combustion (SIAM), San Antonio, TX, April 8-10, 2013.

"Active subspace identification in surrogate modeling," T. Russi, A. Packard, and M. Frenklach, 4th International Workshop on Model Reduction in Reacting Flows, San Francisco, CA, June 19-21, 2013 (invited).

"UQ-Predictive Modeling of Chemical Reaction Systems," M. Frenklach and A. Packard, a thematic session on "Uncertainty Quantification in Chemical Kinetics and Thermodynamics" at the *VIIIth Congress of the International Society of Theoretical Chemical Physics*, Budapest, August 25-31, 2013 (invited).

"How many variables does it take to model combustion chemistry" M. Frenklach, Multi-Agency Coordination Committee for Combustion Research (MACCCR)—6th Annual Fuel and Combustion Research Review, Arlington, VA, September 23-26, 2013 (invited).

"A Transdiscipline Partnership: Use of Predictive Software for Textual Study," Lewis Lancaster and Michael Frenklach, an International Workshop on *Korean Studies Technologies*, Sogang University, Seoul, Korea, December 6, 2013 (invited).

"Active Subspace Identification in Surrogate Modeling," T. Russi, M. Frenklach, and A. Packard, SIAM Conference on Uncertainty Quantification, Savannah, Georgia, March 31 – April 3, 2014 (invited).

"Bound-to-Bound Data Collaboration," M. Frenklach and A. Packard, 2nd Conference of the International Society of Nonparametric Statistics", Cádiz, Spain, June 12-16, 2014 (invited).

"A hybrid cloud system for combustion kinetics simulation," G. L. Goteng, M. Speight, N. Nettyam, A. Farooq, M. Frenklach, and S. M. Sarathy, 23rd International Symposium on Gas Kinetics and Related Phenomena, Szeged, Hungary, July 20-25, 2014; Paper No. A30.

"CloudFlame: A hybrid cloud system for combustion kinetics simulations," M. Sarathy, G. L. Goteng, W. Speight, N. Nettyam, A. Farooq, and M. Frenklach, 35th International Symposium on Combustion, San Francisco, CA, August 3-8, 2014, Work-in-Progress Poster W3P069.

"Uncertainty Quantification Framework for Modeling Prediction," M. Frenklach, Columbia University, Mechanical Engineering, October 10, 2014 (invited).

"Comparison of Probabilistic and Deterministic Frameworks of Uncertainty Quantification," M. Frenklach, A. Packard, J. Sacks, R. Paulo, and G. Garcia-Donato, Mini-Symposium on Uncertainty Quantification in Computational Combustion, 15th International Conference on Numerical Combustion, Avignon, France, April 19-22, 2015 (invited).

6 Significant Interactions

During the past three-year period, close ties have been established with the Combustion Group at KAUST in Saudi Arabia and DLR in Germany. The developed infrastructure has been propagating through the combustion community; its features and standards have begun to be adopted worldwide; for instance, in addition to the developments at the Combustion Center of KAUST, Saudi Arabia described above, the Chemical Kinetics Laboratory of the Eötvös University, Budapest, Hungary released their own website for combustion chemistry (http://respecth.hu/) that is built on the PrIMe Data Models, and the group of Prof. Richard West of the Northeastern University is developing a new Python-based interface to PrIMe Data Warehouse. The participation of the community also keeps growing, as evidenced by a recent workshop that took place on May 21-22, 2015, in Cincinnati, OH. Our goal is to continue working on engaging broader participation in developing the community-based approach to predictive modeling in support of solutions to the most pressing issues in combustion. The future work will focus on expanding the PrIMe database and further automation of predictive modeling.

1.

1. Report Type

Final Report

Primary Contact E-mail

Contact email if there is a problem with the report.

frenklach@berkeley.edu

Primary Contact Phone Number

Contact phone number if there is a problem with the report

5106431676

Organization / Institution name

UC Berkeley

Grant/Contract Title

The full title of the funded effort.

PrIMe Next Frontier: Large, Multi-dimensional Data Sets

Grant/Contract Number

AFOSR assigned control number. It must begin with "FA9550" or "F49620" or "FA2386".

FA9550-12-1-0165

Principal Investigator Name

The full name of the principal investigator on the grant or contract.

Michael Frenklach

Program Manager

The AFOSR Program Manager currently assigned to the award

Chiping Li

Reporting Period Start Date

05/01/2012

Reporting Period End Date

04/30/2015

Abstract

The initiative named PrIMe (for Process Informatics Model) is designed to keep track of models, model parameters, and experimental data in a global, integrated framework for the field of Combustion. It is aimed at curation of community data with the objective of collaborative development of reaction mechanisms of scientific explorations and predictive models for practical systems. The present project was a continuation of prior AFOSR grants (FA9550-08-1-0003 and FA9550-10-1-0450). With the past funding, we focused on moving toward broader cloud-based cyber-infrastructure through distributed data and app sharing. In pursuing this goal, remote-server and browse-based technologies were employed. One of the outcomes is a prototype of a new, redesigned version of the PrIMe platform, Workflow 3.0, which is entirely browser-based and thus can run on all popular platforms.

Distribution Statement

This is block 12 on the SF298 form.

Distribution A - Approved for Public Release

Explanation for Distribution Statement

If this is not approved for public release, please provide a short explanation. E.g., contains proprietary information.

SF298 Form

Please attach your SF298 form. A blank SF298 can be found here. Please do not password protect or secure the PDF The maximum file size for an SF298 is 50MB.

FA9550-12-1-0165_form.pdf

Upload the Report Document. File must be a PDF. Please do not password protect or secure the PDF . The maximum file size for the Report Document is 50MB.

FA9550-12-1-0165.pdf

Upload a Report Document, if any. The maximum file size for the Report Document is 50MB.

Archival Publications (published) during reporting period:

"Interval prediction of molecular properties in parametrized quantum chemistry," D. E. Edwards, D. Yu. Zubarev, A. Packard, W. A. Lester, Jr., and M. Frenklach, Phys. Rev. Lett. 112, 253003 (2014).

"Integrated data-model analysis facilitated by an instrumental model," D. R. Yeates, W. Li, P. R. Westmoreland, W. Speight, T. Russi, A. Packard, and M. Frenklach, Proc. Combust. Inst. 35, 597-605 (2015).

Changes in research objectives (if any):

Change in AFOSR Program Manager, if any:

Extensions granted or milestones slipped, if any:

AFOSR LRIR Number

LRIR Title

Reporting Period

Laboratory Task Manager

Program Officer

Research Objectives

Technical Summary

Funding Summary by Cost Category (by FY, $K)

	Starting FY	FY+1	FY+2
Salary			
Equipment/Facilities			
Supplies			
Total			

Report Document

Report Document - Text Analysis

Report Document - Text Analysis

Appendix Documents

2. Thank You

E-mail user

Jul 21, 2015 11:29:08 Success: Email Sent to: frenklach@berkeley.edu