AFRL-AFOSR-VA-TR-2015-0303

MISSION-CENTERED NETWORK MODELS: DEFENDING MISSION-CRITICAL TASKS FROM DECEPTION

Yolanda Gil
UNIVERSITY OF SOUTHERN CALIFORNIA LOS ANGELES
UNIVERSITY GARDENS STE 203
LOS ANGELES, CA 90089-0001

09/29/2015
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/RTA2

Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
04-09-2015	FINAL REPORT	06/01/2011 - 03/31/2015

4. TITLE AND SUBTITLE
Mission Centered Network Models: Defending Mission-Critical Tasks from Deception (WORKFLOW-NET 2)

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA9550-11-1-0104

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Yolanda Gil

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Southern California
3720 S. Flower Street, Third Floor
Los Angeles, CA 90089-0001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AF Office of Scientific Research
875 North Randolph Street, RM 3112
Arlington, VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR

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

12. DISTRIBUTION/AVAILABILITY STATEMENT

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Traditional cybersecurity has focused on techniques to analyze and eliminate vulnerabilities in a network, often in response to actual security breaches of previously unknown weaknesses. A key desired capability is to be able to accomplish a mission even while the network is compromised and subject to deception. This project investigated a new framework for representing models of mission goals and tasks, and to exploit those models to make a mission more robust to deception operations co-occurring in the network. These mission-centered network models (MCNMs) build on and extend current two-layered (logical/physical) network models by integrating a new layer of task-level representations of the mission into those models. In this new task-oriented layer, a mission can be characterized as a set of goals, each accomplished by a set of interdependent tasks that place requirements on the network resources. The system can then dynamically control the mappings of those tasks onto network resources using a variety of algorithms that take into account which resources are currently compromised. As a result, a mission can be protected from ongoing intrusion and deception activities by dynamically reallocating resources as they become compromised.

15. SUBJECT TERMS
Misson-centered network models, cybersecurity, mission representation, mission planning, mission protection, workflows, semantic workflows

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(Include area code)*

MISSION-CENTERED NETWORK MODELS: DEFENDING MISSION-CRITICAL TASKS FROM DECEPTION

Funded by Air Force Office of Scientific Research
Area: Information Operations and Security
Program Manager: Dr. Robert L. Herklotz, Dr. Tristan Nguyen

Award Number: FA9550-11-1-0104

Final Project Report

Period: June 1, 2011 – March 31, 2015

1. Abstract

Traditional cybersecurity has focused on techniques to analyze and eliminate vulnerabilities in a network, often in response to actual security breaches of previously unknown weaknesses. Recognizing that in practice network operations can never be fully secure, a major focus of recent research is on intrusions that are assumed to be ongoing in the network by one or more malicious parties. In this new view on cybersecurity, a key desired capability is to be able to accomplish a mission even while the network is compromised and subject to deception. However, traditional network models lack a representation of the mission and of how network resources are utilized to accomplish various aspects of the mission. This project investigated a new approach to develop a general framework for representing models of mission goals and tasks, and to exploit those models to make a mission more robust to deception operations co-occurring in the network. These mission-centered network models (MCNMs) build on and extend current two-layered (logical/physical) network models by integrating a new layer of task-level representations of the mission into those models. In this new task-oriented layer, a mission can be characterized as a set of goals, each accomplished by a set of interdependent tasks that place requirements on the network resources. The system can then dynamically control the mappings of those tasks onto network resources using a variety of algorithms that take into account which resources are currently compromised. As a result, a mission can be protected from ongoing intrusion and deception activities by dynamically reallocating resources as they become compromised and by examining provenance records of task outcomes to determine their reliance on compromised resources. MCNMs can be used to determine which resources are critical for any given mission, to prioritize the use of uncompromised resources, to accomplish and estimate the trust on mission tasks when resources are compromised, and to determine the practical impact on the mission of deception activities. MCNMs enable a new approach to cybersecurity in network-based operations.

2. Research Products

Our research focused on four major areas:

1. Development of Task-Centered Network Models (TCNMs)
2. Robust mapping of TCNMs to the execution environment
3. Graph-based clustering algorithms to generate workflow abstractions
4. Algorithms for resource criticality, vulnerability, and reallocation

2.1 Development of Task-Centered Network Models

Network operations offer crucial capabilities to virtual organizations. They offer the ability to assemble a customized network of resources by drawing from disparate pre-existing organizations. For example, when various organizations have relevant data sources for a task, and computing cycles to analyze the data, they may agree to form a virtual organization with the datasets and the computing resources in order to enable its members to carry out data analysis tasks. Such virtual organizations are commonplace in science, business, and military operations. Network operations are vulnerable to attacks. For example, some science applications may use private or sensitive data that might be target of attacks (consider a network of hospitals in the Los Angeles area sharing patient data that likely includes celebrities). In military applications, networked operations offer an effective way to reduce the footprint of a force, but become a center of gravity for the enemy and therefore a potential target for attack and deception.

To address attacks to the network, traditional cybersecurity has focused on techniques to analyze and eliminate vulnerabilities, often in response to actual security breaches of previously unknown weaknesses. Recognizing that in practice network operations can never be fully secure, a major focus of recent research is on intrusions that are assumed to be on-going in the network by one or more malicious parties. The focus is on continuing to conduct operations in the network while it is under ongoing attack. In this new view on cybersecurity, a key desired capability is to be able to accomplish tasks even while the network is compromised. That is, the virtual organization should be able to function even while the network is partially controlled by a malicious party. Therefore, a recent focus for new research is on combining normal network operations to accomplish tasks with mechanisms for defending those tasks against intrusion. Note the shift from protecting the network to a new focus on protecting the tasks.

Unfortunately, protecting organization tasks is not possible in current models of network operations because they current models lack the ability to represent the tasks themselves. Traditional network models represent resources at the physical and logical level. At the physical level are computing hosts, storage facilities, communication lines, and other physical resources. At the logical level are software systems and data sources, which reside in physical resources. These models are appropriate to develop mechanisms to defend those resources at the physical and logical levels. They can be used to protect the network at the physical level in terms of securing connections and physical resources. They can be used to protect the network at the logical level by detecting software attacks and by replicating software resources into

various physical locations to make the network more robust to threats. However, traditional network models lack a representation of how these resources are utilized to accomplish various tasks. Therefore, current network models are limited when the focus is on protecting the tasks rather than the network resources.

The goal of our work is to develop a general framework for representing and exploiting semantic models of high-level tasks in virtual organizations. We define task-centered network models (TCNMs) for representing tasks, building on and extending current two-layered (logical/physical) models. Tasks are represented as *semantic workflows*. Semantic workflows enable the specification of abstract steps that are later on specialized and assigned to network resources.

In prior work, we demonstrated the use of semantic workflows to represent declaratively the application software components and the dataflow among them. We developed semantic representations of workflows that can be used to reason about the data and the processing steps to automatically generate workflows as well as metadata descriptions of new data products that result from the computations. We have combined workflow reasoning with the dynamic assignment of workflow resources. This has involved investigating how conceptual tasks are mapped to available software and data and subsequently mapped to computational resources. We have also explored the implications of this work for compiler research, resource-bound computations, and application performance. We also explored the use of policies in workflow systems to bias the system to create workflows that comply with a given set of policies.

The goal of this project was to examine the research challenges of managing tasks within virtual organizations relying on network operations while the network is under attack. We also investigated how semantic workflows could be used to extend current network models to incorporate representations of those tasks.

Tasks serve the goals of the organization. For example, a virtual organization of scientists may study how ecosystem changes (eg air pollution or water quality) affect the health of patients. Carrying out tasks in a virtual organization involves accessing a variety of resources in the network. Performing the tasks requires accessing three major kinds of resources:

- Data sources: Many distributed data sources are consulted as tasks are being performed. Each data source may specialize in a type of information. For example, a data source may provide access to GIS data, another may provide access to data about personnel available. The same data source may be replicated in several physical resources to allow for load balancing and robust distributed operations.
- Services: In addition to data sources, services are consulted that provide a useful function. Services are accessed remotely by invoking them with a programmatic API. Similar to data sources, the same service may be available from several physical resources.
- Software: Software components may be executed as part of the task. An example is a component for extracting features in an image that may indicate air pollution. The execution often creates new data (in our example an annotated version of the image). There may be several alternative physical resources to run a given software component. For example, the component may require certain libraries to be installed, but many computing resources may have them available.

In service-oriented architectures, services provide a level of abstraction over the implementation of the function they provide. That is, services are defined at the logical level, specifying how to access the resource and what functionality it provides. Services are implemented in a set of physical resources, which can be changed without affecting their specification at the logical level. Therefore, the distinction between physical and logical resources provides the flexibility of separating implementation from function. We exploit this important feature in our approach.

The goal of our work is to protect tasks while the network is undergoing a malicious attack. Current approaches exploit network models that focus on the network and its resources and connections. There is no representation of the tasks themselves, and therefore the system does not have a basis to manage them when the underlying network resources are under attack. In order to protect tasks, we face important research challenges:

- *We need the ability to control how a task as a whole is mapped to physical (and logical) resources.* Moreover, although we currently are aware of the mapping of individual software and data to physical resources, we need in addition to be able to control that mapping and change it. Moreover, we should control how an entire task is mapped to resources, so we can keep the task away from potentially compromised resources.
- *We need the ability to determine what are the critical resources for accomplishing a given task within the mission.* We should be able to know at any point whether and to what level a task is threatened by an intrusion in terms of how its subtasks will be accomplished.
- *We need the ability to manage a task in a network under a threat and move it to a different subset of resources that are not compromised.* We should be able to dynamically reassign resources for a task.
- *We need the ability to prioritize the use of uncompromised resources based on relative criticality of the mission tasks.* We need to allocate trusted resources based on mission-level task criticality and time to completion.
- *We need the ability to measure task accomplishment based on the level of trust that can be placed in task outcomes when the network is compromised.* When we learn about specific resources being compromised, we should be able to examine whether they were used in executing tasks, and determine whether and to what level each task was compromised and therefore whether we can trust its outcome.
- *We need the ability to determine progress, status, and threats to tasks at any given time.* We should be able to track the accomplishment of various individual tasks, the potential threat to accomplishment of upcoming tasks, and the status of ongoing tasks. We should be able to retract tasks that may be untrustworthy when we learn that the resources assigned may have been compromised. Furthermore, we should be able to understand the dependencies of other tasks on those tasks, and be able to re-execute any portions of the mission in new resources.
- *We need the ability to predict the practical impact on tasks for any deception activities in specific (logical or physical) resources.* We should be able to assess whether a given task is mildly, severely, or fatally affected by a specific intrusion activity. We also need to determine whether the task needs to be dynamically changed in terms of what tasks should be reformulated or added in order to make the mission possible given the ongoing threat.
- *We need the ability to manage the threat while guaranteeing the accomplishment of tasks.* We should be able to practice deception by camouflaging the mission's true tasks under unnecessary ones that the intruder might be expecting to see.

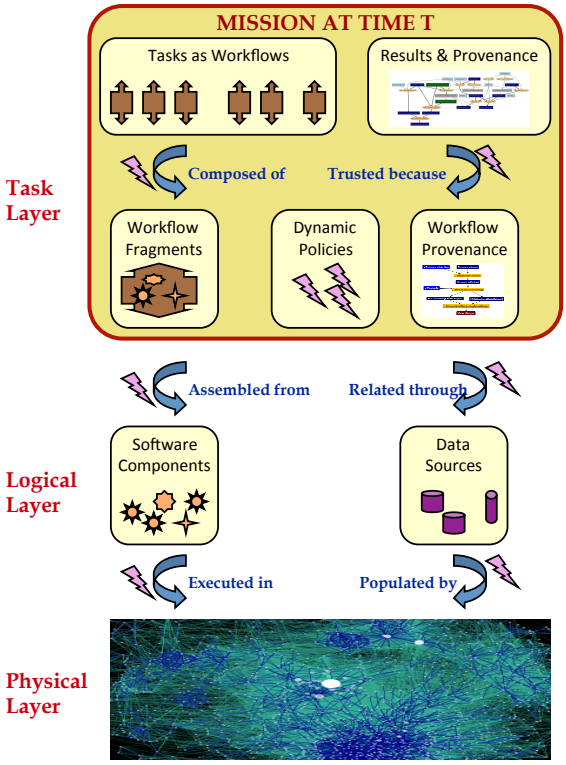


Figure 1. A task-centered network model adds a task-centered layer to traditional network models, enabling better management of tasks through policies that control resource selection and support dynamic reallocation in light of intrusion activities.

- *We need the ability to detect high-level patterns of deception.* We should be able to detect whether and how the patterns of intrusion follow task activities in the network. We should be able to orchestrate simulated tasks mirroring real ones, and learn deception patterns by observing intruder behavior.

In summary, in order to protect tasks in a possibly compromised network we need the ability to do nimble task allocation leading to unpredictability to the intruder. Current network models do not represent tasks explicitly, limiting their ability to meet these challenges. Existing approaches focus on concerns such as denial of service attacks [Mirkovic et al 09; Yu et al 07], worms [Stafford and Li 10], anomaly detection [Shanbhag 10], and intrusion detection [Li et al 07]. Some work is focused on analyzing communications at the network communications level (e.g., [Benzel et al 09]). We build on this work to develop a representation for tasks and their execution in a network.

In summary, in order to protect tasks in a possibly compromised network we need the ability to do nimble task allocation leading to unpredictability to the intruder. Current network models do not represent tasks explicitly, limiting their ability to meet these challenges. Existing approaches focus on concerns such as denial of service attacks [Mirkovic et al 09; Yu et al 07], worms [Stafford and Li 10], anomaly detection [Shanbhag 10], and intrusion detection [Li et al 07]. Some work is focused on analyzing communications at the network communications level (e.g., [Benzel et al 09]). We build on this work to develop a representation for tasks and their execution in a network.

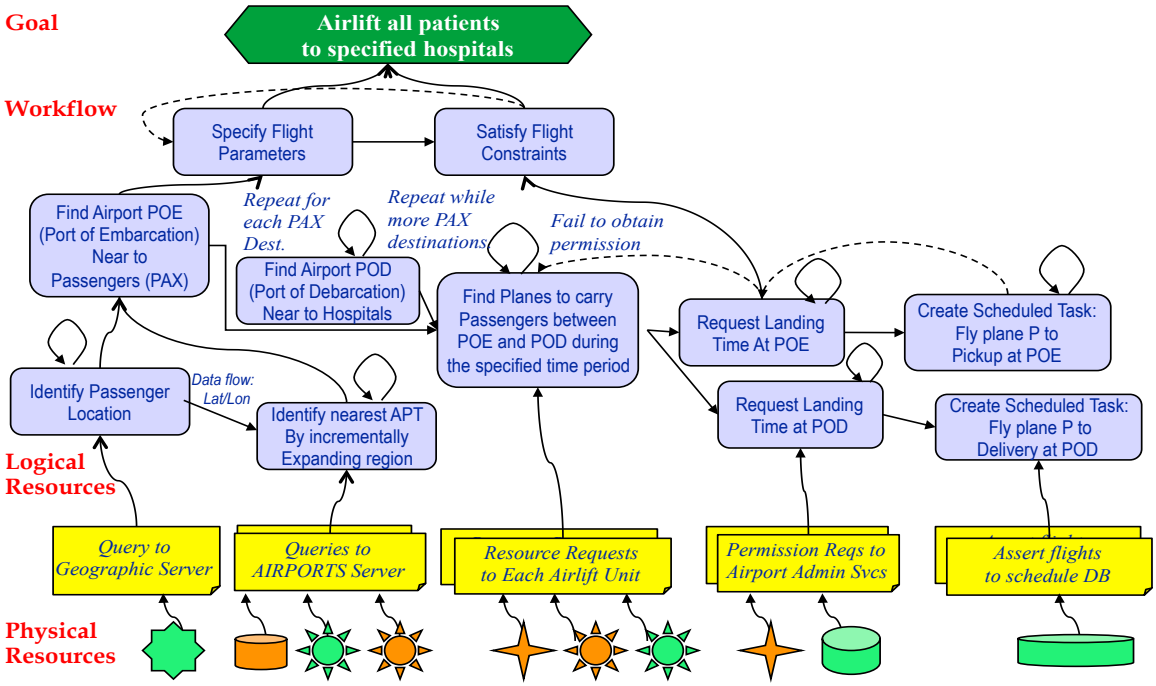


Figure 2. An example of a workflow for airlift of injured personnel (adapted from [Burstein et al 2008].

We propose a new concept of *task-centered network models (TCNMs)* for representing tasks, and exploit those models to make the organization more robust to deception operations co-occurring in the network. TCNMs build on and extend current two-layered (logical/physical) models by integrating task-level representations of the mission into those models. At this task-oriented level, a task can be characterized as a set of goals, each accomplished by a set of sub-tasks. The system gives sub-tasks explicit mappings to logical and physical resources. The system can then protect tasks by assigning them to resources in ways that take into account whether they have been compromised.

A diagram of task-centered network models is shown in Figure 1. At the bottom, we show the Physical Layer and the Logical Layer representing network resources available for the mission. At the Logical Layer, both data and software are identified, and realized at the Physical Layer. The Task Layer represents tasks of the virtual organization, and builds on the Logical Layer by identifying the logical resources utilized by the mission.

Figure 2 illustrates a workflow of airlift tasks adapted from [Burstein et al 2008]. The overall goal of the workflow is to airlift all injured personnel to specified hospitals. The workflow specifies tasks and subtasks to accomplish that goal. The tasks are mapped to logical resources in terms of queries to various queries to types of data sources. These queries are then mapped onto physical resources that are the specific data sources (databases, services, etc.) to be used.

There are three major challenges in designing workflow specifications at the domain layer:

1. A language must be developed to express domain tasks with sufficient generality to encompass equivalent application codes and with sufficient detail to provide meaningful descriptions to scientists.

2. There needs to be a mechanism to map domain tasks into application codes.
3. Automating the generation of executable workflows from workflows of domain tasks.

We use *semantic workflow representations* as the central paradigm for the task level layer. Workflows provide a framework for managing distributed data access and computations, and have been used successfully in a variety of applications.

The term "workflow system" is commonly used in the literature to refer to software that has a diversity of functions and typically includes a workflow execution capability of some sort. We wish to take a broader view on workflow software, and include other software that has useful workflow capabilities, for example a workflow browser, a workflow editor, or a workflow mining system. Therefore we introduce here the terms "workflow tool", "workflow function", and "workflow ecosystem". We refer to a ***workflow tool*** as any kind of software that consumes or generates workflows, and are typically called "workflow systems" in the literature. Workflow tools include Taverna, Pegasus, Moteur, Kepler, and Wings among others. But in our terminology, workflow tools also include software that is less comprehensive than a workflow system, such as workflow browsers, workflow mining tools, etc. A ***workflow function*** indicates a unique capability regarding workflows, such as workflow execution, workflow browsing, or workflow mining. A given workflow tool can have several functions. For example, Taverna includes a workflow editor function, a workflow execution function, and a workflow provenance recording function. When workflow tools are integrated, the integration may involve only a particular function of the tool. For example, when workflow tool A sends a provenance record to workflow tool B which stores it, the execution engine of A would be involved but not its workflow editor or other functions. As workflow technologies and workflow interchange standards mature, these kinds of integration efforts will give raise to ***workflow ecosystems*** that scale up integration efforts.

Our goal is to support the creation and execution of workflows in a way that they can be run interchangeable in a workflow ecosystem consisting of different workflow tools with different workflow functions. This will enable the development of novel cybersecurity frameworks that support TCNMs.

We distinguish three major types of workflow structures at different levels of specificity:

- *Workflow Instance (WI)*: A workflow that specifies the application algorithms to be executed and the data to be used. Workflow instances are sometimes called abstract workflows because they do not specify execution resources. A workflow instance can be submitted to a workflow mapping and execution system, which will identify and assign available resources at run-time, submit it for execution, oversee its execution, and return a workflow execution trace. Because different resources may be available at different times or in different execution environments, the same workflow instance could result in very different workflow execution traces.
- *Workflow Execution (WE)*: Also known as a workflow execution trace or a provenance trace, a workflow execution contains the details of what happened during the execution of a workflow, including what resources it was executed on, execution time for each step, links to the intermediate results, and possibly other execution details such as steps for data movements. When workflow steps fail to execute, a workflow execution

contains annotations of what failed and in this way its dataflow structure may be different from the dataflow in a workflow instance.

- *Workflow Template (WT)*: A generic reusable workflow specification that indicates the types of steps in the workflow and their dataflow dependencies. A workflow instance can be created from a workflow template when datasets are identified. A workflow generation tool can take the types of step specified in the workflow template (e.g., "sort") and specialize them to implemented algorithms and codes (e.g., "Quicksort in Python") to create the workflow instance. Since a type of step can have different implementations, the same workflow template could be used to generate very different workflow instances.

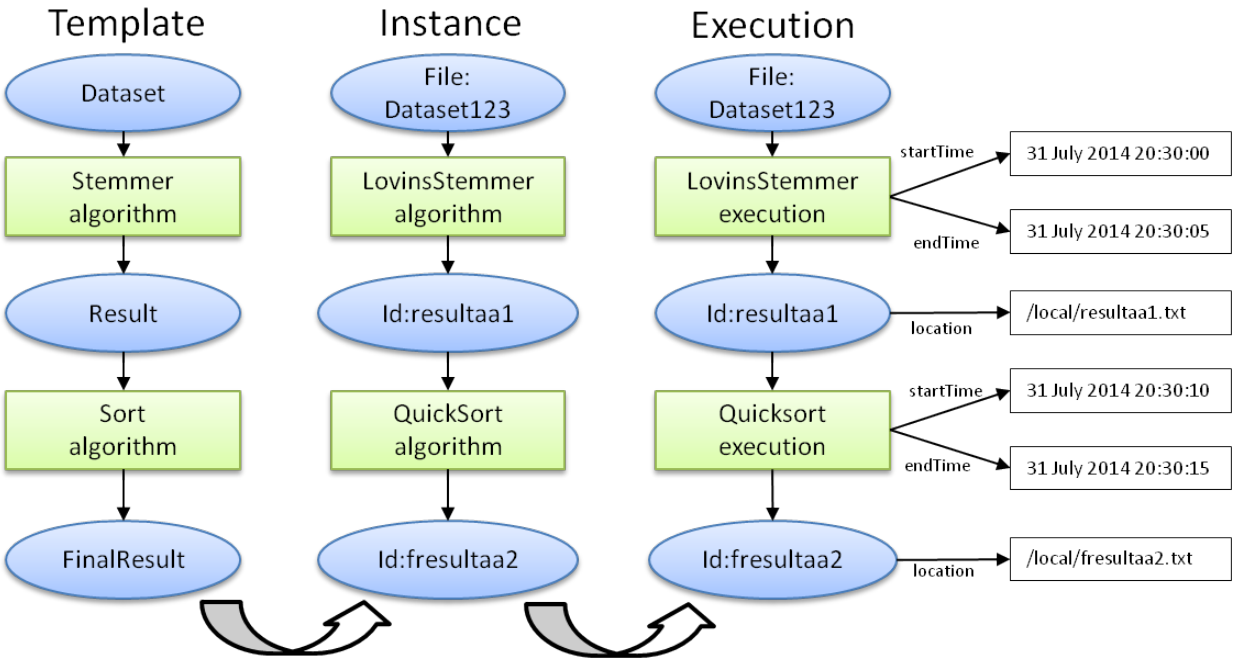


Figure 3. A Workflow Template (left), a Workflow Instance (center), and a Workflow Execution (right).

Figure 3 illustrates with an example the difference between templates, instances and executions. On the left of the figure we can see a template with two steps (Stem and Sort), an input (Dataset) and an output (FinalResult). A workflow instance built from this template is shown in the middle of the figure, specifying Dataset123 as the input and specific executable codes for each of the steps (the LovinsStemmer algorithm and Quicksort algorithm respectively). When this workflow instance is executed, the workflow execution engine produces the execution trace shown on the right (each step has its start and end time, and each intermediate result identifier, e.g., Id:resultaa1, has associated the path of the file it represents).

We use the W3C PROV and the OPM standards to represent workflow executions as provenance records. In addition, we use two extensions of these languages that are more specific to workflows and enable us to represent workflow templates and instances as well as further details of workflow executions: P-plan (an extension of PROV) and OPMW (an extension of OPM).

References

More details about this work can be found in the following articles:

- Representations of task-centered network models as semantic workflows: [Gil-ISI-13] [Garijo-et-al-WORKS-14]
- Provenance-based representations of mission executions: [Garijo-Gil-LISC-12] [Garijo-Gil-WORKS-11]

[Gil-ISI-13] Towards Task-Centered Network Models through Semantic Workflows Gil, Y. In *Proceedings of the IEEE Conference on Intelligence and Security Informatics (ISI)*, Seattle, WA, 2013.

[Garijo-Gil-LISC-12] Augmenting PROV with Plans in P-PLAN: Scientific Processes as Linked Data Garijo, D.; and Gil, Y. In *Second International Workshop on Linked Science: Tackling Big Data (LISC), held in conjunction with the International Semantic Web Conference (ISWC)*, Boston, MA, 2012.

[Garijo-Gil-WORKS-11] A New Approach for Publishing Workflows: Abstractions, Standards, and Linked Data Garijo, D.; and Gil, Y. In *Proceedings of the Sixth Workshop on Workflows in Support of Large-Scale Science (WORKS'11), held in conjunction with SC 2011*, Seattle, Washington, 2011.

[Garijo-et-al-WORKS-14] Towards Workflow Ecosystems Through Semantic and Standard Representations Garijo, D.; Gil, Y.; and Corcho, O. In *Proceedings of the Ninth Workshop on Workflows in Support of Large-Scale Science (WORKS), held in conjunction with the IEEE ACM International Conference on High-Performance Computing (SC),* New Orleans, LA, 2014.

2.2 Robust Mapping to the Execution Environment

Our work in this area focused on representing semantic workflows of domain tasks that abstract the specification of steps from the particular application codes to be executed as well as the particular workflow execution engines used.

This work makes workflows more portable to new execution environments, where the availability of application codes and run-time libraries may vary. This work also allows users to specify MCMNs at a domain level, which enables the system to allocate the execution resources dynamically according to the execution environment.

Workflows have been used to allow users to specify computations in a manner that is independent from the execution environment, where the workflow system automatically maps the codes to whatever execution resources are available at run time. This can be seen as managing a separation between the physical layer and the logical layer of the computation. A workflow specification at the logical layer has a specification of the codes to be executed, and there is no mention of the actual resources where the execution will take place. A workflow specification at the physical layer does mention execution resources that are to be used to run the computations. This separation between the logical and the physical layers is also common in web services frameworks and has been adopted in some workflow systems. A workflow specification at the logical layer is sometimes called an "abstract workflow". Workflow

systems automatically map the workflow specification at the logical layer to a workflow specification at the physical layer. This is an important benefit, as it enables users to run their workflow in different execution environments, bringing flexibility to their applications.

However, workflow representations are still very tied to the execution environment because they specify the application codes that are to be run at each step. For example, depending on the workflow system a step may specify the MATLAB routine or Java code to run, or the signature of the service that needs to be invoked. In this respect, workflows are still tied to particular application codes and software environments. When published in workflow repositories, this limits their reuse by others who may use different software. It also limits their validity over time when code becomes obsolete and no longer runs. Ideally, workflow specifications would be independent of the particular code and software environment, specifying only the domain task to be carried out rather than what application codes to run. Previous work has focused on interoperability of workflow systems and workflow representations, but not on creating more abstract representations that address the domain layer.

Data analysis tasks are implemented by application codes. They represent abstractions of the functions that those codes perform. An example of a domain task is linear regression, which could be implemented with a variety of application codes such as a MATLAB routine, an R routine, a Java code, etc. A semantic workflow specification at the domain layer would have steps that specify domain tasks. A semantic workflow system would then map automatically the workflow specification at the domain layer into a workflow specification at the logical layer. A workflow execution engine would, in turn, map the logical layer (application codes) to the physical layer (i.e., execution resources).

Semantic workflows at the domain layer are closer to specifications of scientific methods that are described in scientific publications. These methods describe steps in ways that are not dependent on the application codes, and are therefore more reusable by other researchers who may use different software.

Workflow specifications at the domain layer would provide scientists with additional flexibility:

- **Facilitate method design and testing**: Scientists would design semantic workflows at the domain layer, and submit the same workflow in a local host or in a shared resource. For example, scientists often use smaller datasets to test different workflow designs on a local host, then when one of the workflow designs looks promising it can be submitted to execution with larger datasets in shared high end computing resources. Their local resource and the shared resource may not have the same codes. For example, a shared resource may have an efficient MPI version of a clustering algorithm, while a local host may have an inefficient one that is easy to install locally and is sufficient for testing purposes.
- **Provide new failure recovery mechanisms**: When the execution of a particular application code fails, the semantic workflow system could select an alternative application code for the same domain task specified in the step that failed.
- **Portability and reproducibility of methods across research groups**: Different research groups often use different software environments for their analyses. Some research groups prefer to use proprietary software for their work (e.g., MATLAB), perhaps because it is more reliable or more efficient. Other groups prefer using open software that implements similar functionality. Other labs prefer to develop their own software. A

semantic workflow specified at the domain level could be more easily transferred and reused across research groups.

- **Archival publication of methods**: The software base that is used today may not be available tomorrow. Software libraries evolve, implementations are revised into new versions, and commercial software has new periodic releases. Libraries and packages are abandoned in favor of new ones. Describing a semantic workflow at the domain layer enables the method to survive the test of time more easily, making workflows more portable over time.
- **Improve provenance and metadata annotations**: Workflow systems automatically annotate provenance and metadata of workflow data products. If the provenance annotated is specific to the application codes then it has limited generality. A semantic workflow system would record provenance in terms of domain tasks independent of the codes executed.
- **Facilitate the transition of methods from research into operational environments**: A research environment may use more exploratory application codes, while an operational environment may use more robust and efficient implementations. Transitioning methods from research to operations could be facilitated with semantic workflow specifications at the domain level.

Our work addresses three major challenges in providing semantic workflow specifications at the domain layer:

1. Representing domain tasks with sufficient generality to encompass equivalent application codes and with sufficient detail to provide meaningful descriptions to scientists.
2. Mapping domain tasks into application codes.
3. Automating the generation of executable workflows from workflows of domain tasks.

We create representations of domain tasks using ontologies. We define a concept (class) for each domain task. Each of the arguments of a task is represented as a property of that class. When used in a workflow, the dataset or parameter for an argument is the value of the corresponding property. These classes form ontologies of domain tasks, containing hierarchies that organize them from more general abstract ones to more specific ones. These ontologies form a *Catalog of Domain Tasks (CDoT)*. The domain tasks in the CDoT are then used to represent workflow steps at the domain layer.

Figure 4 illustrates the representation of domain tasks in a CDoT for machine learning. Domain tasks for machine learning such as modeling and classification, shown in the top part, can be done using different approaches, such as decision trees and Bayesian methods. Two decision tree algorithms are shown: ID3 and J48, each represented as a subclass. Training a classifier using ID3 is a domain task, represented in this CDoT as "ID3-Modeler". This domain task can be implemented in codes in different languages and libraries, and all can be used to execute that task as we describe next.

Domain tasks are ultimately implemented by application codes. We assume that a workflow execution engine has a *Catalog of Application Codes (CAC)*, concerned with the logical layer and that includes implemented codes available to run for individual workflow steps. Therefore, we need a mechanism to map domain tasks in semantic workflows into application codes.

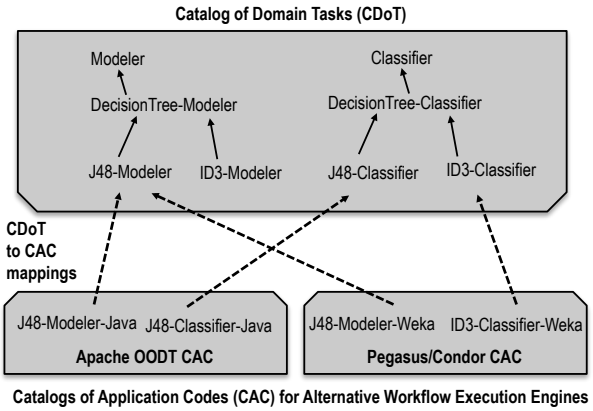


Figure 4. The Catalog of Application Codes (CAC) for each workflow execution engine is mapped to the Catalog of Domain Tasks (CDoT).

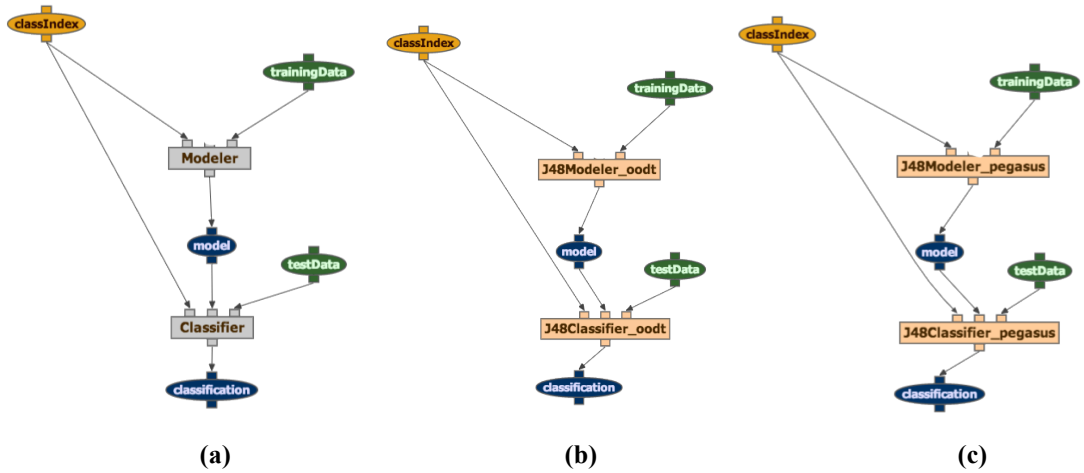


Figure 5. A semantic workflow is defined using component classes from the CDoT, shown in (a), which is independent of the workflow execution engine as well as the application codes available within its execution environment. Two different workflows are generated depending on the particular execution environment and the CAC that is available, shown in (b) and (c).

Our approach to facilitate the mapping of semantic workflows to application codes available in an execution engine is through the use of ontologies. Application codes in the CAC are represented as a class similar to the representation of domain tasks. A CAC must always refer to a CDoT, which can be done through importing the CDoT ontology. Then, each application code available in a workflow execution engine's CAC is mapped to a domain task concept in the CDoT through a subclass relation.

Figure 4 illustrates this, showing the CACs for two different workflow execution engines. There are different application codes for the J48 and ID3 modelers and classifiers available in each workflow execution environment.

The invocation signatures of domain tasks need to be mapped to the signatures of application codes. In our work to date we constrain this problem by making a simplifying assumption. We

assume that each domain task has the same signature as the application codes that implement it. We prepare and encapsulate application codes so that their parameters are always aligned with the domain task that they accomplish.

In prior work, we developed workflow reasoning algorithms that automatically specialize high-level workflows so that each step is mapped to executable components to create a workflow specification at a logical layer. We extended this work so that there is a clear separation between the CDoT and the CAC for the algorithm. The user creates workflows using domain tasks from the CDoT. To run a workflow, the user can then select one of the CACs available for that CDoT. Each CAC is supported by a workflow execution engine that can run those application codes in its execution environment. The workflow reasoning algorithms then generate executable workflows by mapping domain tasks in the CDoT to the application codes available in the selected CAC.

Figure 5(a) shows a semantic workflow of domain tasks from a CDoT for a simple machine learning domain. This workflow takes training data and generates a model, then classifies any new test data according to that model. Figure 5(b) shows a workflow was created using one of the CACs in Figure 4, which represents what is available in the execution environment. Note that the domain tasks that appeared in the workflow of Figure 5(a) have been automatically mapped to application codes in Figure 5(b) and (c) that are available in the execution environment and that appeared in the CAC.

References

More details about this work can be found in the following articles:

- Flexibly mapping task-centered network models to different execution environments and resources: [Gil-ICSC-13]
- Flexible assignment of resources to workflow: [Gil-et-al-WORKS-13]

[Gil-ICSC-13] Mapping Semantic Workflows to Alternative Workflow Execution Engines Gil, Y. In *Seventh IEEE International Conference on Semantic Computing (ICSC)*, Irvine, CA, 2013.

[Gil-et-al-WORKS-13] Time-Bound Analytic Tasks on Large Datasets through Dynamic Configuration of Workflows Gil, Y.; Ratnakar, V.; Verma, R.; Hart, A.; Ramirez, P.; Mattmann, C.; Sumarlidason, A.; and Park, S. L. In *Proceedings of the Eighth Workshop on Workflows in Support of Large-Scale Science (WORKS), held in conjunction with SC 2013*, Denver, CO, 2013.

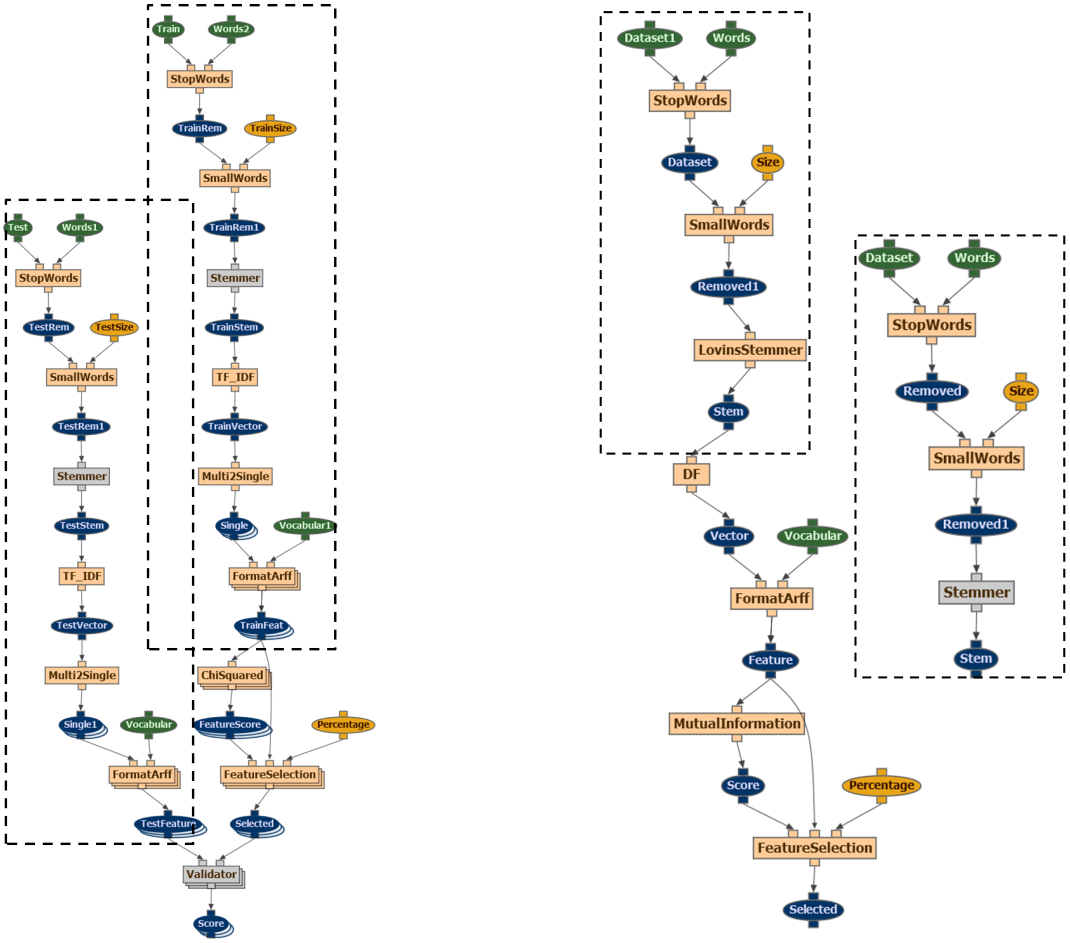
2.3 Graph-Based Clustering Algorithms to Generate Workflow Abstractions

Given a set of workflow templates and/or provenance execution traces we aim to detect the following two types of patterns, illustrated in Figure 6:

1. Internal Macros: This kind of abstraction refers to groups of steps in a workflow that correspond to repetitive patterns of combined tasks. An example can be seen in Figure 6, showing a workflow for document classification. Part of the branches of the workflow

per- form the same tasks (StopWords, SmallWords, Stem-mer, TF IDF, Multi2Single, FormatArff), so we can consider them as an internal macro. Although this pattern might be easy to spot manually in simpler tem- plates, it gets increasingly hard to detect when work- flows grow in size.

2. Composite workflows: Abstraction referring to a workflow composed by one or more sub-workflows. In this work we have expanded the original definition, considering under this category those workflows with overlapping fragments. Figure 6 shows an example of a composite workflow, where a template for stemming a document appears in another template for feature selection. Note that the Stemmer step is specialized in the larger workflow, so it is not an exact match.



(a) **(b)**

Figure 6: Internal macros and composite workflows: (a) Example of an internal macro within a workflow for feature selection, two branches have the same sequence of steps. (b) Example of a composite workflow, where an abstract workflow template used for stemming (right) is part of another that performs a feature selection of the input data (left).

Given a dataset that contains both workflow templates and workflow execution provenance traces, our goal is to detect Internal Macros and Composite Workflows. We face two major challenges in our work:

1. Detection of common workflow fragments in the work- flow dataset. It is important to note that we are not interested in retrieving just the largest common subworkflows, but also the most frequent smaller workflows.

2. Generalization of workflow fragments to derive abstract workflow templates. For instance, two different workflow templates may be in fact specializations of the same abstract workflow template. By noting explicitly their common abstract workflow fragments, we make them comparable and easier to understand (both are alternative implementations of the same method).

If we consider a workflow as a dataflow graph, we can use graph algorithms to detect common fragments as common subgraphs. We represent the workflow templates and the workflow execution provenance traces as labeled graphs.

Nodes are labeled with the types of the workflow components of the taxonomy and the types of the data being used and produced in the workflow; while the edges are labeled with the dependencies of the workflow (usage when a computational process consumes an input and generation when a computational step produces an output). Our goal is finding the maximum overlapping subgraphs between N different graphs, where N corresponds to the number of workflows or execution provenance traces available in the dataset. Since this problem can be reduced to the subgraph isomorphism problem between two graphs, its complexity is NP-Complete.

Our approach is based on the application of graph mining techniques to a workflow corpus. We represent workflows as labeled directed acyclic graphs (LDAGs). This section introduces the algorithms that have used**Error! Reference source not found.**. We reuse mining techniques for Frequent Graph Matching detection (FGM). We have used three different algorithms that use two types of graph mining techniques:

1. *Inexact FGM Techniques:* techniques that use approximate measures to calculate the similarity between two graphs, and detect the commonalities among them. In general, these techniques apply heuristics to detect the most common workflow fragments efficiently. However, the solution is not always complete. Therefore, these techniques do not identify all the possible common fragments in the corpus. We use the SUBDUE algorithm, which applies graph clustering on the input corpus, and provides two different heuristics for detecting common fragments:
    - *Minimum description length (MDL):* at each iteration, the best fragment is chosen according to the bytes necessary to encode the input graph collection.
    - *Size*: at each iteration, the best fragment is chosen according to how the overall size of graph collection is reduced.
2. *Exact FGM Techniques:* techniques that deliver all the possible frequent sub-graphs included in a dataset. FragFlow integrates two different algorithms:

- *The gSpan algorithm*, which is one of the most popular exact FGM techniques by using a deep first search strategy and canonical representation for each graph.
- *FSG*, which uses a breadth first strategy and a canonical labeling method for graph comparison.

Since FSG works only on undirected labeled graphs, some of the fragments it finds on our LDAGs are incorrect. We extended the algorithm to filter FSG results.

Both exact and inexact FGM techniques have advantages and disadvantages. On the one hand, inexact FGM techniques tend to find smaller but highly frequent patterns, although they might not identify potential useful workflow fragments since they are incomplete.
On the other hand, exact FGM techniques identify all fragments possible in the input corpus. However, when the frequent sub-graph size in the input dataset is high, exact FGM techniques might return too many fragment candidates, while consuming significant computational resources. Adjusting the parameters of the search (e.g., frequency of the desired workflow fragments) is important in order to obtain the best results.

Another important aspect is how each technique considers the graphs of the input corpus. Some approaches are *frequency based*, i.e., they consider each candidate structure based on the number of times they appear (frequency), while others are *transaction based*, i.e., they only consider a candidate structure once per input graph (support). For example, if a three step sequence appears 200 times in a workflow, a frequency based approach would consider the sequence of steps as a candidate fragment that appeared 200 times, while a transaction based approach would count it as one. Both techniques produce valuable workflow fragments, and are worth considering for detecting workflow common fragments.
In our work, all the currently integrated inexact FGM techniques are frequency based, while exact FGM techniques are transaction based.

We tested our approach on three different workflow corpora

- User Corpus 1 (WC1): A set of 790 workflows (475 workflows after applying our filtering) designed mostly by a single user. Some of the workflows are product of collaborations with other users, which produced different versions of workflows originally produced by this user. The domain of the workflows is in general medical imaging. Other workflows were designed for a specific purpose and are not reused anymore.
- User Corpus 2 (WC2): A set of 113 well-documented workflows (96 after filtering) created and validated by one user sometimes in collaboration with others. Most of the workflows have been made public for others to reuse, and range from neuroimaging to genomics. Some of the workflows were developed as early as 2007, and many of them are being used in different institutions.
- Multi-user Corpus 3 (WC3): A set of 5859 workflows (357 after filtering), submitted to LONI pipeline for execution by 62 different users over the time lapse of a month (Jan 2014). The number of filtered workflows descends drastically from the input corpus as many of the executions are on the same workflow or are one component workflows designed for testing.

This evaluation led to several important findings. Our first main finding is that when the fragment frequency is set to 10% of the size of the corpus, 30% to 75% of the total fragments found correspond directly to user defined groupings in the single user corpora. In the multi user corpus, the best results are 50% to 56% with minimum frequency. If we consider the overlap of 80% of the steps, the precision is 40% to 80%. However, there is no common configuration to obtain the best fragment results, as the fragments found depend on how users define the groupings on each corpus.

Our second main finding is that users find the fragments proposed by our algorithms as useful candidates for groupings, and therefore useful for reuse in their workflows. For one user 66% of the proposed fragments were useful, for another 100% were useful. Even though this preliminary user based evaluation cannot be considered definitive (an evaluation with more participants is needed), it indicates that our approach can be useful to users. In this regard, in our case it is better to be accurate with the fragments suggested to the user rather than trying to find all the possible desired groupings, as that could overwhelm the user with suggestions.

Lastly, we studied the distribution and size of the groupings in the corpora, which gives an insight into how workflows and groupings are reused. We showed how likely they are to be reused, and found that there is a minimum of 4 groupings per workflow in those workflows using groupings. We also showed how much are they reused, with 209 groupings reused 1463 times in WC1.

In summary, these algorithms can be used with different settings, varying the minimum or maximum frequency of the fragments to find, their minimum and maximum size and the type of the graph mining algorithm to be applied. By combining different configurations, we believe we will be able to improve the outcome of our system, according to user-defined preferences.

References

More details about this work can be found in the following articles:

- Automatic detection of workflow fragments through provenance models: [Garijo-et-al-eScience-12; Garijo-et-al-FGCS-14]
- Detection of common workflows in multi-media processing tasks [Sethi-et-al-MM-13] [Sethi-et-al-PRL-13]
- Graph-based clustering algorithms of workflow libraries to detect normal behaviors: [Garijo-et-al-KCAP-13] [Garijo-et-al-eScience-14a] [Garijo-et-al-eScience-14b]

[Garijo-et-al-eScience-12] Common Motifs in Scientific Workflows: An Empirical Analysis Garijo, D.; Alper, P.; Belhajjame, K.; Corcho, O.; Gil, Y.; and Goble, C. In *Proceedings of the IEEE Conference on e-Science*, Chicago, Illinois, 2012.

[Garijo-et-al-KCAP-13] Detecting Common Scientific Workflow Fragments Using Templates and Execution Provenance Garijo, D.; Corcho, O.; and Gil, Y. In *Seventh ACM International Conference on Knowledge Capture (K-CAP)*, Banff, Canada, 2013.

[Sethi-et-al-MM-13] Large-Scale Multimedia Content Analysis Using Scientific Workflows Sethi, R. J.; Gil, Y.; Jo, H.; and Philpot, A. In *Twenty-First ACM International Conference on Multimedia*, Barcelona, Spain, 2013.

[Sethi-et-al-PRL-13] Structured Analysis of the ISI Atomic Pair Actions Dataset Using Workflows Sethi, R. J.; Jo, H.; and Gil, Y. *Pattern Recognition Letters*, 34(15). 2013.

[Garijo-et-al-FGCS-14] Common Motifs in Scientific Workflows: An Empirical Analysis Garijo, D.; Alper, P.; Belhajjame, K.; Corcho, O.; Gil, Y.; and Goble, C. *Future Generation Computer Systems*, 36, 2014.

[Garijo-et-al-eScience-14a] FragFlow: Automated Fragment Detection in Scientific Workflows Garijo, D.; Corcho, O.; Gil, Y.; Gutman, B. A.; Dinov, I. D.; Thompson, P.; and Toga, A. W. In Proceedings of the IEEE Conference on e-Science, Guarujua, Brazil, 2014.

[Garijo-et-al-eScience-14b] Workflow Reuse in Practice: A Study of Neuroimaging Pipeline Users Garijo, D.; Corcho, O.; Gil, Y.; Braskie, M. N.; Hibar, D.; Hua, X.; Thompson, N. J.P.; and Toga, A. W. In Proceedings of the IEEE Conference on e-Science, Guarujua, Brazil, 2014.

2.4 Algorithms for Resource Criticality, Vulnerability, and Reallocation

We developed an algorithm for interleaving workflow generation and execution that take advantage of semantic workflow representations of domain tasks. This algorithm enables a workflow system to take into account metadata that is dynamically generated by the worfklow and adjust the workflow accordingly. It also enables users to change the resource availability even during execution, and the workflow is adjusted accordingly. The algorithm supports the incremental submission of partial workflows for execution until completion. As new metadata is generated dynamically during execution for all new data products, the algorithm can incorporate that dynamically generated metadata in the workflow planning process. The algorithm also supports replanning in case a resource is no longer available and in case of failure, not just by reassigning resources but also by redesigning the plan by replacing components that may fail to execute.

By interleaving workflow generation and execution, the system is able to allocate or deallocate resources dynamically. When a resource is compromised, or a resource suddenly has become available, the system can react and reallocate resources for the workflow dynamically. This can be done during workflow generation or during workflow execution. It causes the system to reassign resources for execution. Many workflow execution systems have the ability to handle these changes in resource availability during execution. This algorithm enables a unique capability in being able to do this also during workflow generation.

The algorithm was implemented in the WINGS workflow system, and is being used for a medical application for clinical omics.

The algorithms are also complex because they handle the parallel processing of data and component collections described in compact form in the input workflow, as described in [Gil et al 2009].

Workflows generated with WINGS can be executed in Apache OODT, Pegasus, or by WINGS itself in a distributed setting accessing several local machines in a lab [Gil 2013]. The algorithms here are shown for the WINGS execution since that is the setting that we have used to demonstrate the algorithms, but could be used with the other execution engines.

Table 1. Top-level algorithm for interleaving workflow generation and workflow execution.

```
Algorithm: INTERLEAVED-WF-GENERATION+EXECUTION

Input: seeded-workflow (workflow-template + input-data + input-parameters)
Output: execution-outputs

expanded-workflows ← WORKFLOW-GENERATION
                         (seeded-workflow)
resourced-workflows ← {}
for each expanded-workflow ∈ expanded-workflows
    workflow ← SELECT-RESOURCES(expanded-workflows)
    resourced-workflows ← resourced-workflows ∪ workflow
resourced-workflow ← WF-SELECTION(resourced-workflows)

/* The following also creates .met files for the execution outputs */
execution-outputs ← WORKFLOW-EXECUTION-WINGS
                        (resourced-workflow)

while resourced-workflow.is-incomplete do
    expanded-workflows ← WORKFLOW-GENERATION
                             (seeded-workflow)
    new-resourced-workflows ← {}
    for each expanded-workflow ∈ expanded-workflows
        workflow ← SELECT-RESOURCES(expanded-workflows)
        new-resourced-workflows ← new-resourced-workflows ∪ workflow
    new-resourced-workflow ← resourced-workflow
    /* Select the first workflow that has more steps than current workflow */
    while num-steps(new-resourced-workflow) =
              num-steps(resourced-workflow)
        new-resourced-workflow ← dequeue(new-resourced-workflows)
    if new-resourced-workflow = resourced-workflow then
        mark-error(resourced-workflow)
        break
    end if
    resourced-workflow = new-resourced-workflow
    execution-outputs ← WORKFLOW-EXECUTION-DISTRIBUTED
                             (resourced-workflow)
end while

return execution-outputs
```

Table 1 shows a high-level description of the algorithm for interleaving workflow generation, resource allocation, and workflow execution. The algorithm starts with a seeded workflow consisting of a workflow template which may include abstract steps (e.g. a component class whose instances are executable components), data bindings for some or all of the input variables, and value bindings for some or all of the input parameters. A workflow generation step generates a set of expanded workflows that have executable components for all the steps, bindings for all the input data variables, values for all the input parameters, and individual nodes to process each item in any collections in the workflow (data collections and component collections). The workflows generated at this stage may be truncated workflows.

The next step is to select resources for each of the expanded workflows. This is done based on the software and hardware requirements of the components and taking into account the execution resources available. There may be many possibilities, and one may be selected by the user or by the system (for example based on minimizing execution time).

Next, the selected workflow is executed. If it is a truncated workflow, it will only proceed until the breakpoints.

Table 2. The main steps of the workflow generation algorithm.

Algorithm: **WORKFLOW-GENERATION**

Input: *seeded-workflow (workflow-template + input-data + input-parameters)*
Output: *expanded-workflows*

/* Add inferred constraints to the initial seeded workflow */
/* See Table 2 */
inferred-workflow ← **GET-INFERRED-WORKFLOW***(seeded-workflow*)

/* Component selection/validation & incorporation of input data requirements */
/* See Table 3 */
specialized-workflows
← **SELECT-OR-VALIDATE-COMPONENTS***(inferred-workflow*)

/* Input data selection/validation */
/* See Table 4 */
bound-workflows ← {}
for each *specialized-workflow* ∈ *specialized-workflows* **do**
valid-workflows ← **SELECT-OR-VALIDATE-INPUT-DATA**
(specialized-workflow)
bound-workflows ← *bound-workflows* ∪ *valid-workflows*

/* Parameter selection/validation and handling of data ad component collections */
/* See Table 5 */
configured-workflows ← {}
for each *bound-workflow* ∈ *bound-workflows* **do**
valid-workflows ← **CONFIGURE-WORKFLOWS***(bound-workflow)*
configured-workflows ← *configured-workflows* ∪ *valid-workflows*

/* Expand workflows */
/* See Table 6 */
expanded-workflows ← {}
for each *configured-workflow* ∈ *configured-workflows* **do**
workflow ← **EXPAND-WORKFLOW***(configured-workflow)*
expanded-workflows ← *expanded-workflows* ∪ *workflow*

return *expanded-workflows*

Table 3. Algorithm to select candidate resources for nodes

Algorithm: **SELECT-RESOURCES**

Input: *expanded-workflow*
Output: *resourced-workflow*

resourced-workflow ← create-copy(*expanded-workflow*)
node-list ← get-nodes(*resourced-workflow)*
for each *node* ∈ *node-list*
node-constraint ← **component-catalog:**find-component-requirements*(node*)
component-requirements ← get-component-requirements(*node-constraint*)
candidate-machine-ids ←
resource-catalog:get-matching-machine-ids(*component-requirements*)
*node.*set-machine-ids(*candidate-machine-ids*)

return *resourced-workflow*

Table 4. Algorithm to run workflow in distributed machines

Algorithm:**WORKFLOW-EXECUTION-WINGS**

```
Input: resourced-workflow
Output: execution-outputs

execution-outputs ← {}
execution-plan ← CREATE-EXECUTION-PLAN(resourced-workflow)

/* Get steps that haven't run yet, and whose predecessors have successfully run */
executable-steps ← get-steps-ready-to-execute(execution-plan)
while executable-steps ≠ {} do
    for each executable-step ∈ executable-steps do
        machine-ids ← executable-step.machine-ids
        healthy-machine-ids ← filter-healthy-machines(machine-ids)
        machine-id ← random(healthy-machine-ids)

        code ← executable-step.code
        upload-list ← executable-step.input-files
        upload-list ← upload-list ∪ files-in-directory(code.directory)
        upload-list-with-md5 ← create-md5-hashes(upload-list)
        modified-files ← check-remote-cache(machine-id, upload-list-with-md5)
        upload-files(machine-id, modified-files)

        environment ← resource-catalog:get-environment(machine-id)
        arguments ← get-invocation-arguments(executable-step)

        job-detail ← run-component-remotely(machine-id, arguments, environment)

        /* Set execution step provenance: status and log */
        executable-step.set-log(job-detail.log)
        executable-step.set-status(job detail.status)

        output-files ← executable-step.output-files
        download-list ← output-files
        /* Also download .met files for the produced outputs */
        download-list ← download-list ∪ get-met-files(output-files)
        download-files(machine-id, download-list)
        execution-outputs ← execution-outputs ∪ download-list

    executable-steps ← get-steps-ready-to-execute(execution-plan)
end while
return execution-outputs
```

The algorithm then iterates the workflow generation, resource selection, and workflow execution until there is an execution failure, there are no possible resources to assign to some workflow task, or there are no more nodes in the workflow to execute.

Table 2 describes the five major steps of the high-level workflow generation algorithm, which are elaborated further in the publications below. Its input is a seeded workflow whose workflow template nodes can contain abstract components (this means that any executable component in that abstract component class is valid). The output is a set of expanded workflows, each with executable components for all the nodes, each with bindings for all the input data and parameters, and each containing all the constraints associated with its components and the data. All these expanded workflows have a consistent set of constraints, since any workflow generated by the algorithm that has inconsistent constraints will be eliminated.

Tables 3 and 4 show the algorithms for the resource allocation and distributed workflow execution steps of the high-level algorithm shown in Table 1. Table 3 shows the resource selection algorithm. All workflow systems have algorithms similar to this. For each node in the workflow, the execution requirements of the component are matched against the capabilities of the available execution resources. In our case, all the possible assignments are returned, and either the user does the selection or an automated algorithm does. Table 4 shows the distributed workflow execution algorithm. It runs the workflow on a distributed set of machines within a lab, where each workflow component is run on a machine randomly selected after filtering out available machines from the list of machines where the component can run. The execution engine connects to the machine via SSH, and uploads the component software and input data if not already uploaded (modifications to the component software are checked via md5 hashes). The Resource Catalog is consulted in order to get all the environment variables that point to the software resources needed. Once all components and data are uploaded, a remote execution job is started on the machine, and the execution log is sent back to the client. This includes any met files generated during execution. For each step, the log and the status are set to record the provenance.

References

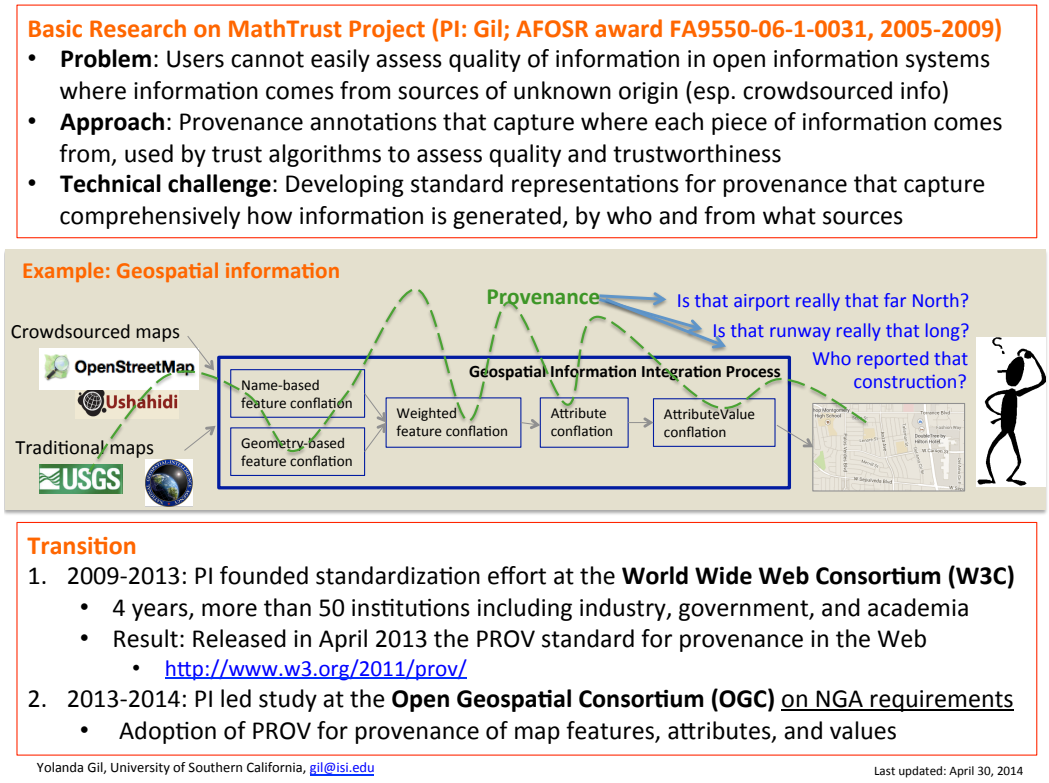
More details about this work can be found in the following articles:

- Algorithms for mapping workflows to execution resources, suspending workflow execution, and reassigning resources.

[Gil-Ratnakar-WORKS-15] "Dynamically Generating Metadata and Replanning by Interleaving Workflow Generation and Execution." Gil, Y. and Ratnakar, V. Submitted to the IEEE Tenth International Workshop on Workflows in Support of Large-Scale Science (WORKS 2015), in conjunction with the IEEE International Conference on Supercomputing, Austin, TX 2015.

3. Tech Transition: Provenance Representations

We had a significant transition mostly based on a prior AFOSR award but related to and enabled by the award reported here.



- World Wide Web Consortium (W3C) standard for provenance on the Web: [Moreau-et-al-W3C-13] [Gil-et-al-W3C-13]
- Open Geospatial Consortium (OGC) report for provenance of geospatial conflation processes [Maso-et-al-OGC-14]
- Provenance of geospatial representations: [Harth-Gil-LGD-14] [Garijo-et-al-PA-14] [Garijo-et-al-IPAW-14]

[Moreau-et-al-W3C-13] PROV-DM: The PROV Data Model Moreau, L.; Missier, P.; Belhajjame, K.; B'Far, R.; Cheney, J.; Coppens, S.; Cresswell, S.; Gil, Y.; Groth, P.; Klyne, G.; Lebo, T.; McCusker, J.; Miles, S.; Myers, J.; Sahoo, S.; and Tilmes, C. World Wide Web Consortium (W3C), Technical Report, April 2013.

[Gil-et-al-W3C-13] A Primer for the PROV Provenance Model Gil, Y.; Miles, S.; Belhajjame, K.; Deus, H.; Garijo, D.; Klyne, G.; Missier, P.; Soiland-Reyes, S.; and Zednik, S. World Wide Web Consortium (W3C) Technical Report, April 2013.

[Harth-Gil-LGD-14] Geospatial Data Integration with Linked Data and Provenance Tracking Harth, A.; and Gil, Y. W3C/OGC Workshop on Linking Geospatial Data (LGD), London, UK, 2014.

[Garijo-et-al-PA-14] User Requirements for Geospatial Provenance Garijo, D.; Gil, Y.; and Harth, A. 2014. Provenance Analytics, co-located with the Fifth International Provenance and Annotation Workshop (IPAW), Cologne, Germany, 2014.
[Garijo-et-al-IPAW-14] Challenges in Modeling Geospatial Provenance Garijo, D.; Gil, Y.; and Harth, A. In Proceedings of the Fifth International Provenance and Annotation Workshop (IPAW), Cologne, Germany, 2014.
[Maso-et-al-OGC-14] OGC® Testbed 10 Provenance Engineering Report. Maso, J., Closa, G., Gil, Y, and Proß, B. Open Geospatial Consortium (OGC) report, document 14-001, July 14, 2014.

4. Archival Publications

- "Towards Task-Centered Network Models through Semantic Workflows." Gil, Y. In Proceedings of the IEEE Conference on Intelligence and Security Informatics (ISI), Seattle, WA, 2013.
- "Mapping Semantic Workflows to Alternative Workflow Execution Engines." Gil, Y. In Seventh IEEE International Conference on Semantic Computing (ICSC), Irvine, CA, 2013.
- "Time-Bound Analytic Tasks on Large Datasets through Dynamic Configuration of Workflows." Gil, Y.; Ratnakar, V.; Verma, R.; Hart, A.; Ramirez, P.; Mattmann, C.; Sumarlidason, A.; and Park, S. L. In Proceedings of the Eighth Workshop on Workflows in Support of Large-Scale Science (WORKS), held in conjunction with SC 2013, Denver, CO, 2013.
- "Towards Workflow Ecosystems Through Semantic and Standard Representations." Garijo, D.; Gil, Y.; and Corcho, O. In Proceedings of the Ninth Workshop on Workflows in Support of Large-Scale Science (WORKS), held in conjunction with the IEEE ACM International Conference on High-Performance Computing (SC), New Orleans, LA, 2014.
- "Detecting Common Scientific Workflow Fragments Using Templates and Execution Provenance." Garijo, D.; Corcho, O.; and Gil, Y. In Seventh ACM International Conference on Knowledge Capture (K-CAP), Banff, Canada, 2013.
- "Large-Scale Multimedia Content Analysis Using Scientific Workflows." Sethi, R. J.; Gil, Y.; Jo, H.; and Philpot, A. In Twenty-First ACM International Conference on Multimedia, Barcelona, Spain, 2013.
- "Structured Analysis of the ISI Atomic Pair Actions Dataset Using Workflows." Sethi, R. J.; Jo, H.; and Gil, Y. Pattern Recognition Letters, 34(15). 2013.
- "A Primer for the PROV Provenance Model." Gil, Y.; Miles, S.; Belhajjame, K.; Deus, H.; Garijo, D.; Klyne, G.; Missier, P.; Soiland-Reyes, S.; and Zednik, S. World Wide Web Consortium (W3C) Technical Report, April 2013.
- "PROV-DM: The PROV Data Model." Moreau, L.; Missier, P.; Belhajjame, K.; B'Far, R.; Cheney, J.; Coppens, S.; Cresswell, S.; Gil, Y.; Groth, P.; Klyne, G.; Lebo, T.; McCusker, J.; Miles, S.; Myers, J.; Sahoo, S.; and Tilmes, C. World Wide Web Consortium (W3C), Technical Report, April 2013.
- "Geospatial Data Integration with Linked Data and Provenance Tracking." Harth, A.;

and Gil, Y. W3C/OGC Workshop on Linking Geospatial Data (LGD), London, UK, 2014.

- "User Requirements for Geospatial Provenance." Garijo, D.; Gil, Y.; and Harth, A. 2014. Provenance Analytics, co-located with the Fifth International Provenance and Annotation Workshop (IPAW), Cologne, Germany, 2014.
- "Challenges in Modeling Geospatial Provenance." Garijo, D.; Gil, Y.; and Harth, A. In Proceedings of the Fifth International Provenance and Annotation Workshop (IPAW), Cologne, Germany, 2014.
- "OGC Testbed 10 Provenance Engineering Report." Maso, J., Closa, G., Gil, Y, and Proß, B. Open Geospatial Consortium (OGC) report, document 14-001, July 14, 2014.
- "Common Motifs in Scientific Workflows: An Empirical Analysis." Garijo, D.; Alper, P.; Belhajjame, K.; Corcho, O.; Gil, Y.; and Goble, C. Future Generation Computer Systems, 36, 2014.
- "FragFlow: Automated Fragment Detection in Scientific Workflows." Garijo, D.; Corcho, O.; Gil, Y.; Gutman, B. A.; Dinov, I. D.; Thompson, P.; and Toga, A. W. In Proceedings of the IEEE Conference on e-Science, Guarujua, Brazil, 2014.
- "Workflow Reuse in Practice: A Study of Neuroimaging Pipeline Users." Garijo, D.; Corcho, O.; Gil, Y.; Braskie, M. N.; Hibar, D.; Hua, X.; Thompson, N. J.P.; and Toga, A. W. In Proceedings of the IEEE Conference on e-Science, Guarujua, Brazil, 2014.

5. Changes in Research Objectives

None.

6. Change in AFOSR Program Manager

Dr. Robert L. Herklotz was the AFOSR Program Manager responsible for the Information Operations and Security Area until his retirement in May 2014. Dr. Tristan Nguyen is the new Program Manager for this area.

7. Extensions Granted or Milestones Slipped

None.

1.

1. Report Type

Final Report

Primary Contact E-mail
Contact email if there is a problem with the report.

gil@isi.edu

Primary Contact Phone Number
Contact phone number if there is a problem with the report

310-822-1511

Organization / Institution name

University of Southern California

Grant/Contract Title
The full title of the funded effort.

Cybersecurity through Nimble Task Allocation: Workflow Reasoning for Mission-Centered Network Models

Grant/Contract Number
AFOSR assigned control number. It must begin with "FA9550" or "F49620" or "FA2386".

FA9550-11-1-0104

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

Yolanda Gil

Program Manager
The AFOSR Program Manager currently assigned to the award

Tristan Nguyen

Reporting Period Start Date

06/01/2011

Reporting Period End Date

03/31/2015

Abstract

Traditional cybersecurity has focused on techniques to analyze and eliminate vulnerabilities in a network, often in response to actual security breaches of previously unknown weaknesses. Recognizing that in practice network operations can never be fully secure, a major focus of recent research is on intrusions that are assumed to be ongoing in the network by one or more malicious parties. In this new view on cybersecurity, a key desired capability is to be able to accomplish a mission even while the network is compromised and subject to deception. However, traditional network models lack a representation of the mission and of how network resources are utilized to accomplish various aspects of the mission. This project investigated a new approach to develop a general framework for representing models of mission goals and tasks, and to exploit those models to make a mission more robust to deception operations co-occurring in the network. These mission-centered network models (MCNMs) build on and extend current two-layered (logical/physical) network models by integrating a new layer of task-level representations of the mission into those models. In this new task-oriented layer, a mission can be characterized as a set of goals, each accomplished by a set of interdependent tasks that place requirements on the network resources. The system can then dynamically control the mappings of those tasks onto network resources using a variety of algorithms that take into account which resources are currently compromised. As a result, a mission can be protected from ongoing intrusion and deception activities by dynamically reallocating

resources as they become compromised and by examining provenance records of task outcomes to determine their reliance on compromised resources. MCNMs can be used to determine which resources are critical for any given mission, to prioritize the use of uncompromised resources, to accomplish and estimate the trust on mission tasks when resources are compromised, and to determine the practical impact on the mission of deception activities. MCNMs enable a new approach to cybersecurity in network-based operations.

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.

FinalReport-SF 298_WORKFLOW-NET 2 (#FA9550-11-1-0104) copy 2.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.

FinalReport-2015-SUBMITTED.pdf

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

Archival Publications (published) during reporting period:

"Towards Task-Centered Network Models through Semantic Workflows." Gil, Y. In Proceedings of the IEEE Conference on Intelligence and Security Informatics (ISI), Seattle, WA, 2013.

"Mapping Semantic Workflows to Alternative Workflow Execution Engines." Gil, Y. In Seventh IEEE International Conference on Semantic Computing (ICSC), Irvine, CA, 2013.

"Time-Bound Analytic Tasks on Large Datasets through Dynamic Configuration of Workflows." Gil, Y.; Ratnakar, V.; Verma, R.; Hart, A.; Ramirez, P.; Mattmann, C.; Sumarlidason, A.; and Park, S. L. In Proceedings of the Eighth Workshop on Workflows in Support of Large-Scale Science (WORKS), held in conjunction with SC 2013, Denver, CO, 2013.

"Towards Workflow Ecosystems Through Semantic and Standard Representations." Garijo, D.; Gil, Y.; and Corcho, O. In Proceedings of the Ninth Workshop on Workflows in Support of Large-Scale Science (WORKS), held in conjunction with the IEEE ACM International Conference on High-Performance Computing (SC), New Orleans, LA, 2014.

"Detecting Common Scientific Workflow Fragments Using Templates and Execution Provenance." Garijo, D.; Corcho, O.; and Gil, Y. In Seventh ACM International Conference on Knowledge Capture (K-CAP), Banff, Canada, 2013.

"Large-Scale Multimedia Content Analysis Using Scientific Workflows." Sethi, R. J.; Gil, Y.; Jo, H.; and Philpot, A. In Twenty-First ACM International Conference on Multimedia, Barcelona, Spain, 2013.

"Structured Analysis of the ISI Atomic Pair Actions Dataset Using Workflows." Sethi, R. J.; Jo, H.; and Gil, Y. Pattern Recognition Letters, 34(15). 2013.

"A Primer for the PROV Provenance Model." Gil, Y.; Miles, S.; Belhajjame, K.; Deus, H.; Garijo, D.; Klyne, G.; Missier, P.; Soiland-Reyes, S.; and Zednik, S. World Wide Web Consortium (W3C) Technical Report, April 2013.

"PROV-DM: The PROV Data Model." Moreau, L.; Missier, P.; Belhajjame, K.; B'Far, R.; Cheney, J.; Coppens, S.; Cresswell, S.; Gil, Y.; Groth, P.; Klyne, G.; Lebo, T.; McCusker, J.; Miles, S.; Myers, J.; Sahoo, S.; and Tilmes, C. World Wide Web Consortium (W3C), Technical Report, April 2013.

"Geospatial Data Integration with Linked Data and Provenance Tracking." Harth, A.; and Gil, Y. W3C/OGC Workshop on Linking Geospatial Data (LGD), London, UK, 2014.

"User Requirements for Geospatial Provenance." Garijo, D.; Gil, Y.; and Harth, A. 2014. Provenance Analytics, co-located with the Fifth International Provenance and Annotation Workshop (IPAW), Cologne, Germany, 2014.

"Challenges in Modeling Geospatial Provenance." Garijo, D.; Gil, Y.; and Harth, A. In Proceedings of the Fifth International Provenance and Annotation Workshop (IPAW), Cologne, Germany, 2014.

"OGC Testbed 10 Provenance Engineering Report." Maso, J., Closa, G., Gil, Y, and Proß, B. Open Geospatial Consortium (OGC) report, document 14-001, July 14, 2014.

"Common Motifs in Scientific Workflows: An Empirical Analysis." Garijo, D.; Alper, P.; Belhajjame, K.; Corcho, O.; Gil, Y.; and Goble, C. Future Generation Computer Systems, 36, 2014.

"FragFlow: Automated Fragment Detection in Scientific Workflows." Garijo, D.; Corcho, O.; Gil, Y.; Gutman, B. A.; Dinov, I. D.; Thompson, P.; and Toga, A. W. In Proceedings of the IEEE Conference on e-Science, Guarujua, Brazil, 2014.

"Workflow Reuse in Practice: A Study of Neuroimaging Pipeline Users." Garijo, D.; Corcho, O.; Gil, Y.; Braskie, M. N.; Hibar, D.; Hua, X.; Thompson, N. J.P.; and Toga, A. W. In Proceedings of the IEEE Conference on e-Science, Guarujua, Brazil, 2014.

Changes in research objectives (if any):

None.

Change in AFOSR Program Manager, if any:

Dr. Robert L. Herklotz was the AFOSR Program Manager responsible for the Information Operations and Security Area until his retirement in May 2014. Dr. Tristan Nguyen is the new Program Manager for this area.

Extensions granted or milestones slipped, if any:

None.

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

Sep 04, 2015 21:59:17 Success: Email Sent to: gil@isi.edu