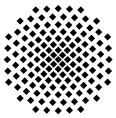

Universität Stuttgart

SimTech Cluster of Excellence

D. Karastoyanova[1] D. Dentsas[1] D. Schumm[1] M. Sonntag[1] L. Sun[1] K. Vukojevic[1]

# Service-based Integration of Human Users in Workflow-driven Scientific Experiments

Stuttgart, October 2012

**[1] Institute of Architecture of Application Systems (IAAS)**
University of Stuttgart,
Universitaetsstrasse 38
70569 Stuttgart, Germany
{karastoyanova, schumm, sonntag, vukojevic}@iaas.uni-stuttgart.de
http://www.iaas.uni-stuttgart.de

**Abstract** The use of information technology in research and practice leads to increased degree of automation of tasks and makes scientific experiments more efficient in terms of cost, speed, accuracy, and flexibility. Scientific workflows have proven useful for the automation of scientific computations. However, not all tasks of an experiment can be automated. Some decisions still need to be made by human users, for instance, how an automated system should proceed in an exceptional situation. To address the need for integration of human users in such automated systems, we propose the concept of Human Communication Flows, which specify best practices about how a scientific workflow can interact with a human user. We developed a human communication framework that implements Communication Flows in a pipes-and-filters architecture and supports both notifications and request-response interactions. Different Communication Services can be plugged into the framework to account for different communication capabilities of human users. We facilitate the use of Communication Flows within a scientific workflow by means of reusable workflow fragments implementing the interaction with the framework.

**Keywords** Scientific Workflows, Human Communication Flows, Cyber-infrastructure, Communication Services.

**Reference** Karastoyanova, D., Dentsas, D., Schumm, D., Sonntag, M., Sun, L., and Vukojevic, K. (2012) Service-based Integration of Human Users in Workflow-driven Scientific Experiments. In: Proceedings of the 8th IEEE International Conference on eScience, Chicago, USA, October 2012.

© IEEE
The original publication is available at: http://ieeexplore.ieee.org/

**Stuttgart Research Centre for Simulation Technology (SRC SimTech)**

SimTech – Cluster of Excellence
Pfaffenwaldring 7a
70569 Stuttgart
publications@simtech.uni-stuttgart.de
www.simtech.uni-stuttgart.de

# Service-based Integration of Human Users in Workflow-driven Scientific Experiments

Dimka Karastoyanova, Dimitrios Dentsas, David Schumm,
Mirko Sonntag, Lina Sun, and Karolina Vukojevic
Institute of Architecture of Application Systems (IAAS)
University of Stuttgart
Stuttgart, Germany
lastname@iaas.uni-stuttgart.de

***Abstract*—The use of information technology in research and practice leads to increased degree of automation of tasks and makes scientific experiments more efficient in terms of cost, speed, accuracy, and flexibility. Scientific workflows have proven useful for the automation of scientific computations. However, not all tasks of an experiment can be automated. Some decisions still need to be made by human users, for instance, how an automated system should proceed in an exceptional situation. To address the need for integration of human users in such automated systems, we propose the concept of Human Communication Flows, which specify best practices about how a scientific workflow can interact with a human user. We developed a human communication framework that implements Communication Flows in a pipes-and-filters architecture and supports both notifications and request-response interactions. Different Communication Services can be plugged into the framework to account for different communication capabilities of human users. We facilitate the use of Communication Flows within a scientific workflow by means of reusable workflow fragments implementing the interaction with the framework.**

***Keywords- Scientific Workflows, Human Communication Flows, Cyber-infrastructure, Communication Services.***

## I. INTRODUCTION

Scientific workflows enable the automation of scientific experiments and are thus beneficial to scientists in terms of easier modeling, cost, speed, accuracy, and flexibility. Workflows implement the logic of an experiment by composing and connecting domain-specific functions and code modules, enabling so-called programming-in-the-large, known as a concept from workflow technology [28]. Many concepts and approaches to leverage workflows in e-Science have been proposed [8], coining the term Scientific Workflow Management Systems (SWfMS). Many of the currently available SWfMS cover a broad spectrum of scientific problem domains, e.g., Kepler [21], Triana [22], Taverna [23], and Pegasus [24], however, the technologies used are not in all cases applicable for additional domains. In contrast, the workflow technology used in business environments is designed to be generic and applicable for various business processes or the integration of enterprise applications. Business workflow languages and systems focus more on business-critical aspects like interoperability, standardization, scalability, robustness, and transactionality. Since many of these aspects are also relevant for workflow-driven scientific experiments, our approach is to enhance and extend conventional business workflow technology to also meet the requirements and needs of e-Science applications [10]. The major goal is to create a flexible cyber-infrastructure that provides the benefits of business workflow systems and supports scientists in the design, execution, monitoring, and analysis of experiments. We build on standards like the Business Process Execution Language (BPEL) for workflow design and execution [14]. Domain-specific, scientific computation functions to be used in workflows are wrapped using Web service technology [15].

In scientific experiments not all tasks can be automated and therefore an interaction of the SWfMS with human users is often needed. In business workflow technology, this is addressed by so-called human task management and worklist applications, e.g. for BPEL an integration of human users is already in place [11, 12]. Making this functionality available in e-Science experiments is a first step, but not yet sufficient. Consider the following example scenario in which a human user needs to be involved in a scientific workflow: A scientist has to go on a business trip to a conference abroad for one week. Prior to his departure he starts a long-running simulation experiment. During the conference he has no direct access to the running workflow that coordinates the computations. In case the simulation enters an error state, he can only react to this failure after returning from the conference trip, assuming the SWfMS does not typically provide remote monitoring and control functionality. Therefore, valuable computing time may be wasted and the simulation results get delayed. The approach presented in this paper changes this scenario: Before leaving for the conference, the scientist extends the simulation workflow with reusable workflow fragments [4] which represent logic to activate a communication flow. In case of an error, a communication flow is activated and informs the scientist about the error and proposes different actions, e.g. retry or abort. Based on presence information of the scientist [16], a particular communication channel is chosen. For example, when not present in the lab and not online in his messenger, the scientist will receive an e-mail with the error log attached and a set of predefined options for error handling. The scientist can also forward the error message to another person who is present in the lab or who has remote access to

the workflow system in order to perform more advanced failure recovery. This scenario is just one example where involvement of human users is useful. Further examples are: status updates to be sent frequently (e.g. notifications about simulation completeness), approvals to be made (e.g. if results should be sent to a poster printer), or enquiries for parameters (e.g. for specifying data quality properties) and others. The concept we propose in this paper does not focus on one particular scenario, but on a general solution for seamless integration of the human way of working with workflow-driven scientific experiments.

As initial step, we devised an abstract architecture of a workflow-driven infrastructure for scientific computations that allows human intervention [3]. In cooperation with industry partners we then specified the flows in detail, message formats, and data structures required for human communication [1]. Based on these foundations we refined and implemented the approach [5, 6]. In this paper, we present the essential contributions and key conclusions of these efforts.

The paper is structured follows. In Section II we present an architectural overview of the approach, covering communication participants, flow of messages, and key components with their interrelations. In Section III we describe the Human Communication Manager (HCM). In Section IV we discuss different services that can be used for communication with human users over a variety of channels. In Section V we describe how the framework seamlessly integrates with workflow-driven scientific experiments. Our implementation is presented in brief in Section VI. In Section VII we compare our approach with related work. The paper concludes with a summary and outlook on future work in Section VIII.

## II. ARCHITECTURAL OVERVIEW

To establish communication between a sender (i.e. a scientific workflow) and a receiver (i.e. a human user) we use different components and protocols, as illustrated in Figure 1. The communication initiator, e.g. a scientific workflow, sends a request to the 'Human Communication Manager', which coordinates the routing of the request to the human user. Depending on the user's availability and preferences a 'Communication Service' is invoked to deliver the message to a device that is currently being used by the human user. Depending on the chosen communication channel, different protocols come into play, for example SMTP for e-mail delivery. In case the initiator expects a response, the human user sends a response (dashed grey arrows in Figure 1) to the corresponding service, which returns it to the HCM. The response is checked for syntactic correctness and delivered to the initiator either via a callback or by means of polling.

We distinguish three major types of messages exchanged between the different components (see Figure 2). The initiator prepares a 'Communication Request Message' containing the message payload (i.e. the communication message) and parameters interpreted by the HCM. These parameters include the recipients, information about if and until when a response is required, which communication channel should be used, and details about the expected response format.

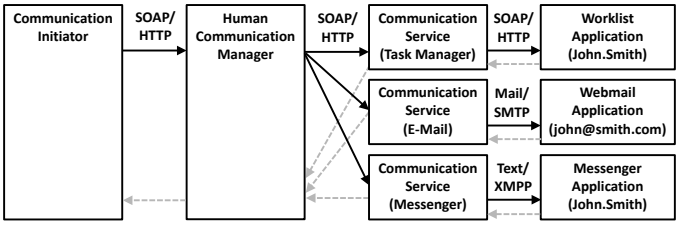


Figure 1. Components and protocols involved in communication between a communication initator and a human user's communication device.

The HCM extracts the 'Communication Message' and passes it to a Communication Service via the chosen communication channel. This service transforms the message properties into a 'Channel-specific Message', which allows proper rendering of the information sent to the human user. The service also processes the expected response message and transforms it into a service- and channel-specific response template. More details about message structures and data formats can be found in [1].

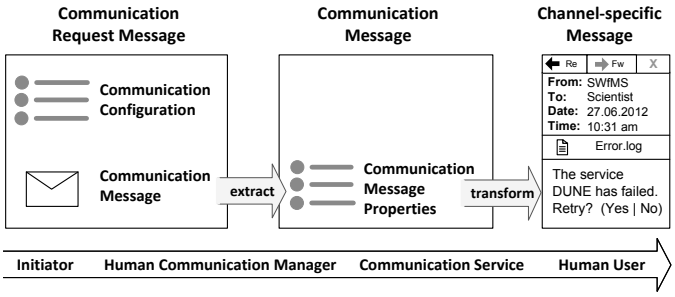


Figure 2. Message trandformations on the route to the human user.

The overall architecture is depicted in Figure 3. The Workflow Modeling and Monitoring Environment [10] is a rich client application for workflow design and execution control. In order to integrate scientific workflows with human users, predefined and reusable workflow fragments (aka process fragments or fragments) are used. The fragments represent the logic for invoking the HCM using standard workflow language constructs and can be shared in a Fragment Library [4]. A fragment is a connected sub-graph of a workflow graph typically used as a reusable artifact to enrich existing workflows with particular logic – in our case with logic related to the interaction with the HCM.

In the rich client platform we realize the human task integration through a worklist plug-in which offers a feature-rich interface for managing tasks, e.g. for claiming an open task, for suspending a running task, or for sorting by importance. Responses to a started task can be made by filling a form. The human user can then also perform semi-automated tasks, which require the usage of further Scientific Applications and access to information sources like a Provenance Store.

The Workflow Execution Environment executes the workflow, i.e. it orchestrates Scientific Services to perform an experiment. When a workflow fragment for communication with a human user is reached, the HCM is invoked, which routes the task to the human user.

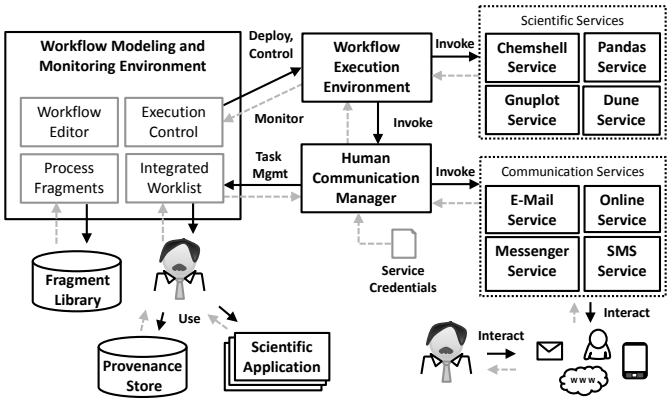


Figure 3. Architecture considering heavyweight (worklist) and lightweight (service-based) integration of human users in scientific workflows.

If the user is signed-in within the rich client the task will appear on the worklist. Otherwise, the task is routed to one of the Communication Services. Depending on the devices the human user had registered, a service capable of the required transport protocol routes the task to the user. In case of two-way communication, responses are sent back via the chosen communication channel.

## III. HUMAN COMMUNICATION MANAGER

In this section, the key perspectives of the HCM are presented from a flow perspective, component architecture perspective, and message-oriented realization perspective.

### A. Human Communication Flows

We have conducted several case studies with scientists working in the domain of simulation technology [25, 26]. In these case studies, workflows for scientific experiments and simulations have been developed to orchestrate scientific computation functions that are wrapped using Web services. The case studies show that in many cases a notification about certain events, status changes, and completeness updates of the computation would be useful, beyond the GUI of the workflow editor and monitor as this information is also useful when the scientist is out of the lab for a longer time. Further discussions showed that there are many scenarios in which also a two-way communication is very useful, like parameter selections, data checking and correction, and approvals, e.g. if simulation results should be sent to a poster printer or not.

Figure 4 shows a generalized communication flow, designed using the Business Process Model and Notation (BPMN) [20]. This flow captures the requirements from the case studies, comprising one-way and two-way communication. In the following, we describe the main reasons for the design of the flow. Typically, notifications and two-way communication – contact a user and process a response by the user – are considered trivial. However, this straightforward flow does not consider the user's 'presence', which states the availability status of a user on a given communication channel, e.g., the status in an instant messaging application [16]. Another step before a user is contacted is to select the device with the shortest response time. Also, if multiple users are able to answer a request, the user who is "best" available should be chosen. A timeout mechanism is employed to deal with lost responses and hence to prevent waiting infinitely for a message. Furthermore, the initiator of a communication can predefine a default response that should be returned in case the user does not respond. This leads to a distinction between flow branches for 'response required' and 'response optional'. Sometimes human users do not use the right response format or do not provide all required information. Thus, a cycle for invalid responses needs to be added. This is the last step in the generalized flow for communication (Figure 4), which supports the flow types notification, response required, and response optional. It considers the features presence, validation, timeouts, and predefined responses. The communication flows can be realized as a workflow or as a message-oriented system; we chose the latter. This is reflected in the HCM architecture and communication processing.

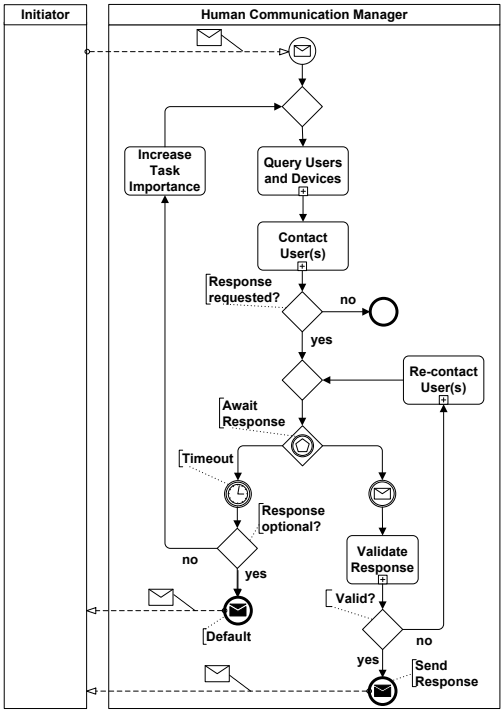


Figure 4. Generalized Human Communication Flow.

### B. Human Communication Manager Architecture

The HCM is hosted by an Application Server. It consists of several conceptually separated components, shown in Figure 5.

*Human Task Manager.* The foundations for human tasks have been laid in the Bangkok project [7, 13], an open source human task manager. Appropriate extensions have been made to Bangkok for loose coupling of Communication Devices and Communication Services. These extensions also include security mechanisms prohibiting unauthorized access to the Worklist API and presentation layer components.

*Messaging.* The messaging component manages all interactions of the components of the business logic layer in a message-oriented way. The underlying routing engine provides the necessary orchestration logic as mandated by human communication flows. It executes all the needed functionality with the help of specific message processors. Transactional

behavior and reliability concerns are being handled by an integrated message broker, which is also responsible for all interactions with the persistent Message Store.

*Frontend Manager.* The Frontend Manager consists of all functions which act as a link between the business logic layer and the presentation layer.

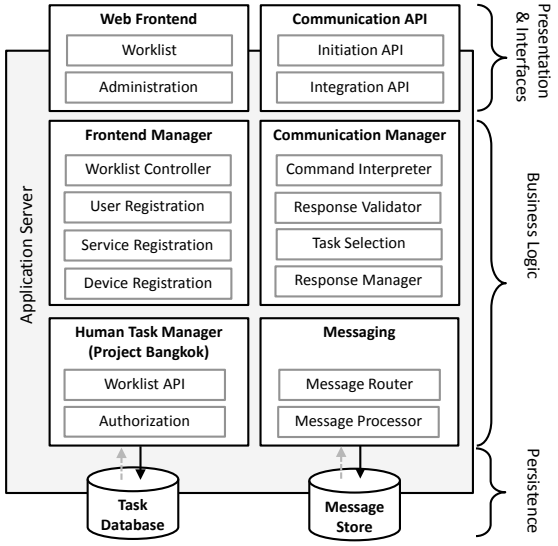


Figure 5. Architecture of the Human Communication Manager.

*Communication Manager.* Human response messages are intrinsically unstructured. This is especially true for text-based messages as they are not rigorously bounded and restricted to predefined response structures. To cope with this phenomenon it is essential to validate and interpret all incoming messages from human users in order to make them understandable for the HCM. For these purposes the Communication Manager offers specialized classes of filters like the Command Interpreter. This interpreter identifies all substrings of the response message conforming to a set of regular expressions which correspond to predefined commands utilizing the HCM's public interfaces. The command #help, for example, can be used to obtain a list of all available commands. Other commands, which are very useful in many scenarios, are '#forward' – to forward a task to another user, '#useChannel' – to move the task to another communication channel, and '#resend' – to start over with a task. If no command has been identified in a response message the HCM assumes that the message signals a completion of a task and validates it against the response schema.

*Communication API.* The HCM offers two distinct Application Programming Interfaces (API) exposed as Web services. The Integration API handles the communication with all registered Communication Services. The Initiation API on the other hand provides an asynchronous method for instantiating new tasks and a callback mechanism for receiving responses to be delivered to the communication initiator. Details about the communication services which integrate with the communication API are covered in Section IV.

*Web Frontend.* The presentation layer provides a feature-rich, Web-based frontend. It comprises a visual representation of the worklist, dynamically generated forms for processing tasks as well as administration functions for managing user accounts, services, and devices.

### C. Human Communication Processing

This part focuses on the exact orchestration rules used by the Messaging component to achieve correct routing behavior. These rules, depicted in Figure 6, are specified based on the nomenclature and definitions introduced by the Enterprise Integration Patterns (EIP) [9].

Incoming initiation messages must pass syntactic and semantic validation checks before they are made persistent by a message queue (Step 1, Figure 6). Depending on the flow type, messages are routed through intermediate steps involving task creation. Note that this detour is only required for two-way communication flows (Step 2). At this point a content-based router separates notification messages sending them to their final destinations after storing them (Step 3). The alternative route, taken when a response is requested, leads up to the central response loop (Step 4). This construct handles all incoming human user input coming either from communication devices or the Web-based frontend. Depending on the chosen input source the provided message parameters are routed to the Worklist Controller or to the Command Interpreter where text-based commands are identified. The message processor validates responses for matching the required response schema. Previously stored and unfinished tasks are also selected and routed periodically to this point. In case of a failed or finished task the response loop is exited (Step 5). If no valid response message was provided after a certain time-frame, optional response messages are enriched with a predefined answer and sent back to the initiator, i.e. to the scientific workflow. Mandatory response messages, however, are kept in the worklist with an increased importance level until they are chosen by an Earliest-Deadline-First algorithm at a later selection cycle.

### D. Advanced Aspects

*Response schema.* We propose a sufficiently rich subset of the XML Schema specification for modeling task structures and for validating their responses, i.e. for the data provided by a human user. The subset includes the standard primitive data types (e.g. `string`, `decimal`, `integer`) and restrictions (e.g. `enumeration`, `length`). Complex types include `sequence`, `choice` and `all`, with the attributes `required`, `optional`, `maxOccurs` and `minOccurs`. The instantiation of the response schema into a response template is specific to the used service and channel and is thus discussed in Section IV.

*Response time.* Certain factors can be considered in order to compute future response probabilities and to shorten the waiting interval. Studies conducted in [17] propose a probabilistic model of individual e-mail.

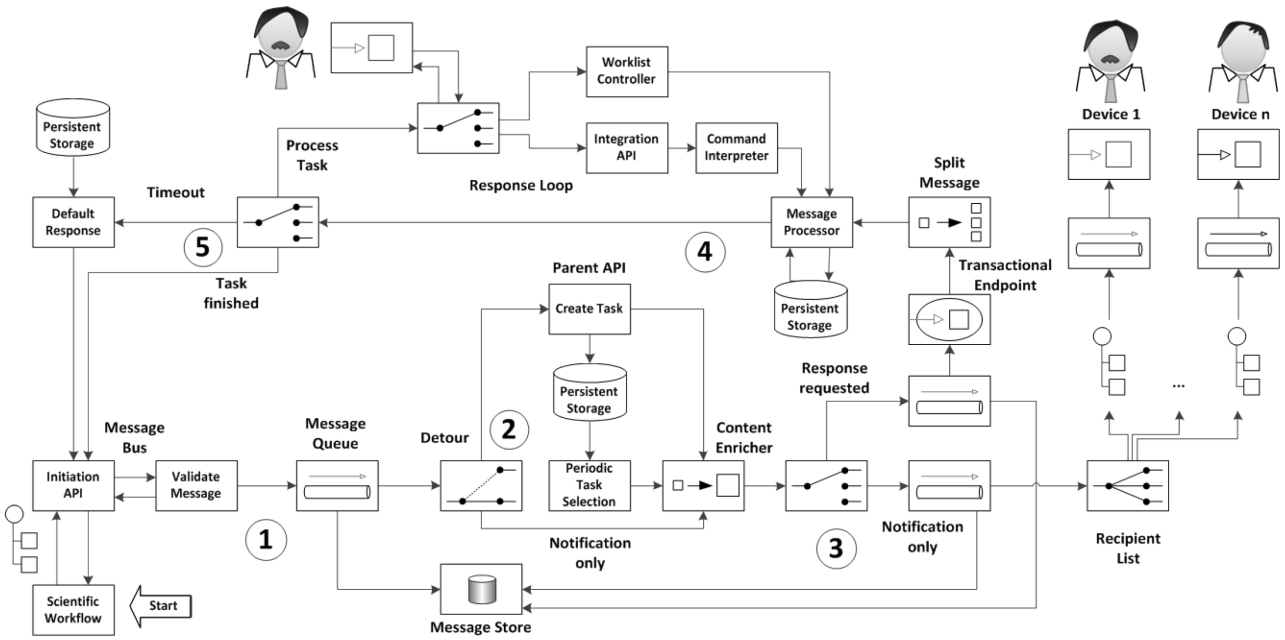


Figure 6. Message-oriented perspective on the processing steps of the communication between a scientific workflow and human users.

The approach defines a cascading non-homogeneous Poisson process as a double-chain hidden Markov model, allowing for an efficient inference algorithm with which a survival function of these Poisson-distributed random variables can be computed and used. When using such models in the HCM, the probability of receiving a response to a submitted request can be calculated at any time. In case a response is unlikely, the task can be rerouted.

*Reliability.* In order to avoid message loss, persistent message queues guarantee the atomic behavior between critical sections coupled with specific error handling protocols, like attempting multiple automated redeliveries. As one of many possible recovery strategies this approach increases the robustness and reliability of the system. Beside the transactional aspect, the HCM was specifically designed to handle concurrent access of shared resources in multi-threaded and multi-user environments and to avoid race conditions through various synchronization techniques.

## IV. HUMAN COMMUNICATION SERVICES

The aim we followed in the architecture was to consider both tight integration with a workflow modeling and monitoring environment and a lightweight and pluggable integration with a variety of channels. Human Communication Services serve the lightweight form of integration. After integration of a new service in the HCM by registering the protocol and its endpoint reference, users can register new devices that can be reached through the newly established communication channel. In this section, we dwell on only two of the Communication Services we used in order to show their features and differences. We also present a generic service template for implementing new Communication Services.

### A. *e-mail Service*

The e-mail service provides a communication channel with two significant properties: (1) the channel is asynchronous, i.e., a human user does not need to be present on his device to receive messages and (2) the messaging capabilities of the channel vary depending on the architecture and implementation. Figure 7 depicts the architecture of an e-mail service which does not implement a complete mail server but which uses a provider like Yahoo! Mail. Depending on the provider used by the service for sending messages, the messaging capabilities change. For example, Yahoo! Mail currently supports subject, body, and attachments up to 25 MB in total size. The service manages all messages in database tables, one for e-mails to be sent (Outbox), one for e-mails received (Inbox), and one for correlation and storage of requests by the HCM and responses by human users.

The e-mail service contains multiple sub-components. The Communication Request Handler is invoked by the HCM. It instantiates the XSD of the requested response and appends it as a response template to the message body. Thus, the instantiated response template is specific for the communication service and channel. As described in [6], an e-mail service may provide the response template as valid XML instance, where the requested information elements are filled with example values.

Furthermore, the e-mail service generates and adds a unique identifier to the e-mail subject for correlation of sent and received e-mails. It puts the prepared e-mail in the Outbox. The Mail Sender sub-component periodically processes the Outbox table and sends all prepared e-mails via an external Mail Server. The timing of the polling is defined in a local configuration. The credentials for the used Mail Server account are either passed to the e-mail service by the HCM or pre-defined in the local configuration.

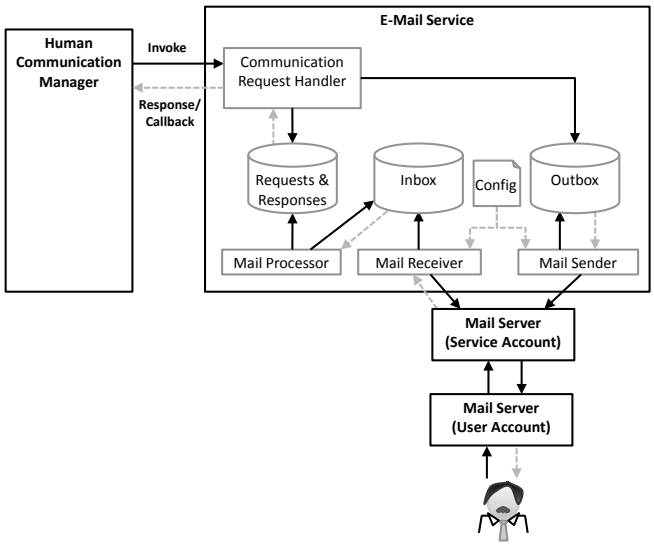


Figure 7. Architecture of the e-mail service using an external mail server.

A human user can then perform a task by simply replying to the e-mail and filling out the response template in the e-mail body. The Mail Receiver periodically polls the Mail Server for new messages, which are then stored in the Inbox table. The Mail Processor periodically processes all new mails stored in the Inbox table, deletes spam messages and stores correlated user responses in the Requests & Responses table. Periodically, the Communication Request Handler checks if tasks have been completed. In case a task is complete, the callback of HCM is invoked returning the response. The e-mail service can also be used stand-alone without the HCM. However, for two-way communication some of the logic realized in the HCM is needed in the e-mail service, like validation of responses and timeout handling. This was investigated in [6].

### B. Google Talk Service

The Google Talk (GTalk) service is different from the e-mail service. Instant messaging is a synchronous channel – typically messages are only delivered when both GTalk users are online. In contrast to the e-mail channel the presence of a human user can be queried and the instant messaging channel would be chosen only when the user is online. However, in order to access the presence of a user and for exchanging messages the human user has to add his GTalk account as contact information. Therefore, a part of the user registration happens outside of the overall framework. One deficiency in instant messaging is that only one task can be delivered to a user at a time. After one request has been started by a user, the service has to wait for a reply before the next request can be sent, i.e., the device is blocked. In the e-mail service this problem does not exist as correlation is made by an identifier added to the subject. In GTalk interactions there is no body and no subject, just text transmissions. So the GTalk service has to put all information into one transmission. For complex requests, the task is chunked and only one parameter is requested at a time. The chunking accounts for mobile phone devices with limited text editing functionality.

In the GTalk service, the user can directly send commands to the HCM. Figure 8 shows sending of the '#help' command to get a list of all available commands.

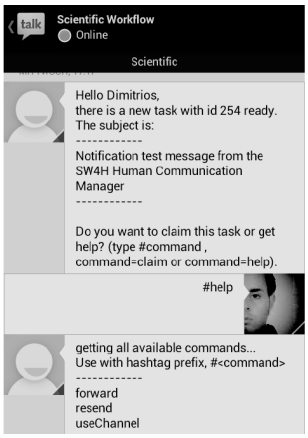


Figure 8. Sending command '#help'to the HCM through the GTalk service.

### C. Communication Service Template

All Communication Services that have to be integrated with the HCM must implement a specific Service Endpoint Interface (SEI) and expose it as a Web service. The SEI operates through a set of methods with primitive data types instead of one complex message, which has to be disassembled and analyzed after each call.

There are three types of methods: (1) getter methods used by the HCM, e.g., to query the presence status of a human user; (2) setter methods used by the HCM for setting a message subject, body, attachments, recipient, response schema, etc.; and (3) methods to trigger a two-way communication with the service by exchanging text messages asynchronously. Not every interface method has to have a proper implementation. In case a specific service does not consider any attachments, then the corresponding method is unnecessary. Therefore, it is important to state the message channel capabilities of a Communication Service, like the maximum length of subject and body, if attachments are allowed, and the maximum file size of attachments. We currently provide such information as plain text. However, this information could also be provided by using the policy language WS-Policy [27] that allows automated matchmaking of services and message requirements.

## V. INTEGRATION IN SCIENTIFIC WORKFLOWS

The integration of human users in scientific workflows can be realized with predefined reusable workflow fragments. The use of fragments reduces the integration of human communication in scientific workflows to a 'drag and drop' operation during the modeling of scientific workflows. Figure 9 shows a fragment providing asynchronous communication with the HCM using BPEL [14]. The assign activity `prepareMessage` constructs a Communication Request Message that will be sent to the HCM by the Web service invocation activity `invokeHCM`. The message contains both the actual message to be presented to the human user and additional information for the HCM. The HCM requires, e.g.,

an EPR pointing to the communication initiator (to whom the response message should be sent) and an XSD defining the structure of a valid response.

After sending the message to the HCM, the following `pick` construct `awaitResponse` waits for a response message from the HCM (callback). When a response message arrives, the `OnMessag'` branch of the `pick` construct is activated. Then, the response message is processed in the `assign` activity `assignResponse`, providing the data contained in the human user's response for further processing. The pick construct also contains a handler for timeout, `OnAlarm`. This construct addresses scenarios where the human user does not respond within a given time or when the HCM cannot be reached. The `assignTimeout` activity then provides information about the missing response message.

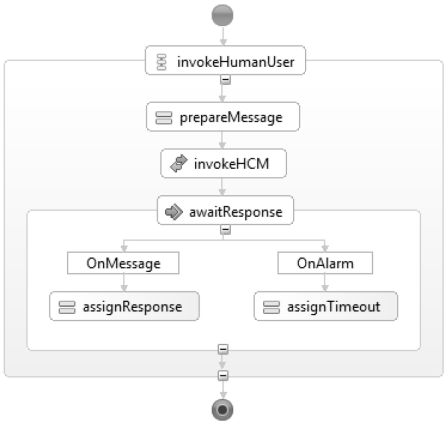


Figure 9. Example workflow fragment for integration of a human user.

We implemented fragments to communicate with human users both using the HCM and directly invoking the Communication Services without the HCM. Note that the HCM accounts for optimal response time with a large variety of communication capabilities, whereas the sole use of Communication Services accounts for easy and fast setup. The presented fragment realizes an asynchronous communication with the HCM using a callback mechanism. A different approach is a polling mechanism not shown here due to space limitations. Instead of waiting for a response message a fragment can periodically poll for available response messages; this however is a decision on the realization and is left to the discretion of the workflow designer/scientist.

## VI. Implementation

As a proof of concept we have implemented an open source HCM-prototype called SW4H (Scientific Workflows for Humans). The prototype is licensed under the Apache 2 license and can be downloaded at [2]. It comes with two prepackaged Communication Services for e-mail (SMTP) and Google Talk (XMPP) communication and a service template for custom service implementations. The solution is based on the Spring Framework managing transactional behavior, security-mechanisms as well as providing the IoC-Container (Inversion of Control). The used Message Broker and Routing Engine are realized using the Apache ActiveMQ and Apache Camel frameworks, respectively. The Human Task Manager is realized by Project Bangkok [13] prototype. The modular structure of the HCM allows easy extensions with new Communication Services and upgrades of user- and task-selection algorithms. The HCM and the Communication Services use Web service technology for interoperability.

For the presentation layer we provide a Web-based Worklist Manager together with all administration interfaces necessary for registering and modifying Communication Services, devices, and user accounts. The Web frontend can be used within Eclipse as a view or stand-alone using a common Web browser, e.g. for performing tasks using a tablet PC. Based on the Eclipse BPEL Designer project, we can now use this feature to provide scientists with a rich client platform for modeling, monitoring, and managing scientific workflows. Recently, we developed extensions to this platform to support the usage of workflow fragments for reusing and sharing the fragments [4]. This extension can be used to easily integrate the interaction logic between the workflow and the HCM.

## VII. Related Work and Comparison

Integration of human users in automated information processing is a typical requirement for business workflow systems. As a consequence, most products provide efficient support, offering a human task manager and a worklist application, like Intalio Tempo [18]. Standards for human tasks like WS-HumanTask [11] and integration in business workflows like BPEL4People [12] have been developed. Such extension specifications account for portability between different products but also require extensions of the workflow modeling and execution environment. We avoided such extensions by modeling workflow fragments providing the necessary invocation logic. Our approach provides the worklist functionality common in business scenarios and additionally considers the integration of Communication Services. The use of different services for integration of human users has been previously applied in industry. For example, the Oracle workflow suite [19] provides multiple services for one-way communication, such as for sending short messages to mobile phones and for sending e-mails. However, in contrast to most offerings, our approach considers presence of users, loose integration of Communication Services, and two-way communication over different channels.

To the best of our knowledge existing SWfMS like Kepler [21], Triana [22], Taverna [23], or Pegasus [24] do not offer an integration of human communication in workflows. Additionally, we have identified two significant differences between the business process technology used in our approach and most SWfMS. The first is the focus on Web service technology for integration and the second is the ability to perform asynchronous calls. These allow for modeling and performing request-response interactions with a human user by means of predefined callbacks; otherwise a polling mechanism needs to be implemented on the side of the communication initiator and the HCM. Notification to a user from the workflow can also be sent either by calling directly a Web service or through a synchronous call to a program, which in turn calls a Web service. The concept of workflow fragments in

our approach is common to both WfMS and SWfMS. Although our fragment approach builds on the workflow language BPEL, reusable fragments can be applied to other languages in a similar manner. Besides, some scientific workflow languages already have ways to capture workflow logic as reusable components, e.g., via groupings of task structures in Triana [22].

## VIII. Summary and Outlook

In this work, we have presented different perspectives of our solution towards integrating interaction with human users in scientific workflows, including architectural perspectives, control and message flow perspectives, services, and implementation. It is important to note that the approach is not limited to scientific workflows; it can be used by any kind of communication initiator capable of invoking Web services. The presented approach significantly facilitates the work of scientists by taking into account different communication channels. As we experienced in discussions with scientists, the beneficial features like availability of different Communication Services and the possibility to flexibly react to exceptional situations makes workflow-driven systems even more appealing and more likely to be applied in future projects.

We presented two Communication Services realizing an instant messaging and an e-mail channel. Additional channels can also be enabled, like FTP for transferring large files to and from a user's device, Twitter for continuously posting status updates, Skype/MSN/… for supporting the variety of instant messaging systems, and encrypted channels for security-sensitive applications. The work is also relevant to the developments in the field of combining social media and BPM Systems, as well as human-supported computing [29].

## Acknowledgment

The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

## References

[1] D. Schumm, C. Fehling, D. Karastoyanova, F. Leymann, and J. Rütschlin, "Processes for Human Integration in Automated Cloud Application Management," Technical Report No. 2012/02, University of Stuttgart, 2012.

[2] D. Dentsas, "SW4H – Scientific Workflows for Humans", Google Code Project, 2012. Available at http://code.google.com/p/sw4h-2012/

[3] D. Schumm and D. Karastoyanova, "Integrating Humans in Scientific Workflows: Integrate, Register & Communicate," In: The 4th SimTech Status Seminar, Poster, 2011.

[4] D. Schumm, D. Dentsas, M. Hahn, D. Karastoyanova, F. Leymann, and M. Sonntag, "Web Service Composition Reuse through Shared Process Fragment Libraries," In: Proceedings of the 12th International Conference on Web Engineering (ICWE'12), Demo, Springer, 2012.

[5] D. Dentsas, "Einbindung von Menschen in Scientific Workflows," (in German), Diploma Thesis No. 3297, University of Stuttgart, 2012.

[6] L. Sun, "Web Services for Human Interaction," Diploma Thesis No. 3275, University of Stuttgart, 2012.

[7] S. Wagner, "A Concept of Human-oriented Workflows," Diploma Thesis No. 2987, University of Stuttgart, 2010.

[8] I. J. Taylor, et al. (Eds.). Workflows for e-Science: Scientific Workflows for Grids. Springer-Verlag New York, 2006.

[9] G. Hohpe and B. Wolf, "Enterprise Integration Patterns: Designing, Building, and Deploying," Addison-Wesley, 2004.

[10] K. Görlach, M. Sonntag, D. Karastoyanova, F. Leymann, and M. Reiter, "Conventional Workflow Technology for Scientific Simulation," In: Guide to e-Science, Springer-Verlag, 2011.

[11] A. Agrawal et al., "Web Services Human Task (WS-HumanTask)," White Paper, 2007.

[12] Agrawal et al., "WS-BPEL Extension for People (BPEL4People)," White Paper, 2007.

[13] S. Wagner and T. Unger, "Project Bangkok", Google Code Project, 2010. Available at http://code.google.com/p/projectbangkok/

[14] Organization for the Advancement of Structured Information Standards (OASIS), "Business Process Execution Language 2.0 (BPEL)," 2007.

[15] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson, "Web Services Platform Architecture," Prentice Hall PTR, 2005.

[16] M. Day, J. Rosenberg, and H. Sugano, "A Model for Presence and Instant Messaging," Network Working Group RFC 2778, 2000.

[17] R. D. Malmgren, et al., "Characterizing Individual Communication Patterns," In: Proceedings (KDD'09), ACM, 2009.

[18] Intalio, "Intalio Workflow Tempo," 2009. Available at http://www.intalio.org/confluence/display/TEMPO/Home

[19] Oracle, "BPEL Process Manager Developer's Guide," Version 10g (10.1.3.1.0), B28981-03, 2007.

[20] Object Management Group (OMG), "Business Process Model and Notation (BPMN)," OMG Available Specification, Version 2.0, 2011.

[21] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, and S. Mock, "Kepler: An Extensible System for Design and Execution of Scientific Workflows," In: Proceedings of the 16th International Conference on Scientific and Statistical Database Management, 2004.

[22] D. Churches, et al., "Programming Scientific and Distributed Workflow with Triana Services," In: Concurrency and Computation: Practice and Experience. Special Issue on Workflows in Grid Systems, 18(10): 1021–1037, John Wiley & Sons, 2006.

[23] D. Hull, et al., "Taverna: A Tool for Building and Running Workflows of Services," In: Journal of Nucleic Acids Research, 34(Web Server issue): 729–732, Oxford University Press, 2006.

[24] E. Deelman, et al., "Pegasus: Mapping Large-scale Workflows to Distributed Ressources," In "Workflows for e-Science—Scientific Workflows for Grids," Springer-Verlag, 2007.

[25] M. Sonntag, S. Hotta, D. Karastoyanova, D. Molnar, and S. Schmauder, "Using Services and Service Compositions to Enable the Distributed Execution of Legacy Simulation Applications," In: Proceedings of ServiceWave, 2011.

[26] P. Reimann, M. Reiter, H. Schwarz, D. Karastoyanova, and F. Leymann, "SIMPL – A Framework for Accessing External Data in Simulation Workflows," In: Proceedings of the 14th BTW 2011, LNI, 2011.

[27] W3C, "Web Services Policy 1.5 - Framework (WS-Policy)," W3C Recommendation, 2007.

[28] F. Leymann and D. Roller, "Production Workflow – Concepts and Techniques," Prentice Hall, 2000.

[29] S. Dustdar and K. Bhattacharya, "The Social Compute Unit." In: IEEE Internet Computing 15(3):64—69, IEEE, 2011.