

Workflow Enactment Based on a Chemical Metaphor

Zsolt Németh, Christian Pérez, Thierry Priol

▶ **To cite this version:**

Zsolt Németh, Christian Pérez, Thierry Priol. Workflow Enactment Based on a Chemical Metaphor. [Research Report] PI 1734, 2005, pp.20. inria-00000160

HAL Id: inria-00000160
https://inria.hal.science/inria-00000160

Submitted on 12 Jul 2005

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION
INTERNE
N° 1734

WORKFLOW ENACTMENT BASED ON A CHEMICAL METAPHOR

ZSOLT NÉMETH AND CHRISTIAN PÉREZ AND THIERRY PRIOL

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Workflow Enactment Based on a Chemical Metaphor

Zsolt Németh * and Christian Pérez ** and Thierry Priol ***

Systèmes numériques
Projet Paris

Publication interne n ° 1734 — Juillet 2005 — 20 pages

Abstract: Executing workflows on large-scale heterogeneous distributed computing systems is a challenging task. Albeit there have been many well elaborated approaches, they are often motivated by a certain class of applications and focus on some implementation specific problems. The work presented in this paper aimed at establishing a highly abstract coordination model for distributed workflow enactment where decentralized control, autonomy, adaptation to high dynamics and partial lack of information are of primary concerns. The model is based on a nature metaphor and envisioned as a chemical reaction where molecules react autonomously according to local and actual conditions. The execution model of such chemical workflow enactment is formalized using the γ-calculus. In the γ-calculus control, scheduling, dependencies, errors and the state of the computation are all represented in a single uniform declarative formalism that has a mathematically founded clear semantics. The paper shows that the abstract coordination model expressed in γ-calculus is able to grasp all aspects of such a chemical enactment, provides a more complex and adaptive framework than most current approaches, and actual realizations may be founded on it.

Key-words: Workflow, Chemical Metaphor, Grid Computing

(Résumé : tsvp)

* Zsolt.Nemeth@sztaki.hu
** Christian.Perez@irisa.fr
*** Thierry.Priol@irisa.fr

Centre National de la Recherche Scientifique
(UMR 6074) Université de Rennes 1 – Insa de Rennes

Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Coordonner les calculs en utilisant une métaphore chimique

Résumé : Coordonner et exécuter des calculs au sein de systèmes distribués hétérogènes et à grande échelle est un réel défi. Bien que ce problème est fait l'objet de nombreuses recherches, les solutions proposées sont souvent dédiées à une certaine classe d'applications et focalisées sur des mises en oeuvre spécifique. Le travail proposé dans ce rapport a pour but d'établir un modèle abstrait de coordination capable d'exécuter des calculs de façon décentralisée, autonome et entièrement dynamique dans un environnement incertain. Le modèle proposé est fondé sur la métaphore de la réaction chimique dans laquelle des molécules réagissent de manière autonome en fonction de conditions locales. Le modèle d'exécution, associé à la coordination et l'exécution de calculs suivant le principe de la réaction chimique, est formalisé en utilisant le γ-calcul. Avec ce modèle de calcul, l'ordonnancement, les dépendances, les erreurs et l'état du calcul sont représentés dans un formalisme déclaratif uniforme et simple, associé à une sémantique claire et mathématiquement fondée. Le papier montre que le modèle abstrait de coordination exprimé en γ-calcul peut saisir tous les aspects de la coordination de calcul. Il fournit un cadre plus large que les approches courantes et peut être implémenté de manière réaliste.

Mots clés : Workflow, Métaphore Chimique, Grilles de calcul

Contents

1 Introduction **4**

2 Problem analysis **5**

3 Related works **6**

4 Workflow enactment based on a chemical metaphor **6**
4.1 Gamma and the γ-calculus 7
4.2 The concept of resource quantum 9

5 Workflow enactment based on the γ-formalism **9**
5.1 Modeling resources using γ-calculus 10
5.2 Representation of a single activity 10
5.3 Resource dependencies 11
5.4 Representation of dependent activities 11
5.5 Combining resource and data/control dependencies 15

6 Concept for realization of the chemical enactment model **16**
6.1 The workflow engines 17

7 Conclusion **18**

8 Acknowledgments **19**

1 Introduction

A workflow is a collection of loosely coupled activities (computational tasks) that are processed in some order and where both data-flow and control-flow relationships may be present [10]. Accordingly, a workflow management system is ”a system that completely defines, manages and executes ”workflows” through the execution of software whose order of execution is driven by a computer representation of the workflow logic” [14].

The Workflow Reference Model of the Workflow Management Coalition [14] provides the most general description of a workflow management system and defines the following activities and entities related to workflow management. At build-time, in the Process Design & Definition phase the individual activities of the workflow are separated and their relationships are established yielding a workflow specification, called Process Definition. At run-time the Process Definition is handled to the Workflow Enactment Service that instantiates and controls the activities according to the logic expressed in the specification. During the enactment, the enactment engine may react to human intervention, may use applications and tools and may cause changes in the Process Definition.

We restrict our scope to the management of scientific workflows executed in large-scale distributed heterogeneous computing environment, such as Grids [12]. Although there are many approaches, they seldom address the entire problem of workflow management, and typically, enactment is rarely in the focus. The enactment of scientific workflows require a complex coordination between activities of the workflow and entities of the distributed system. Such a coordination comprises resource, service, data discovery and selection, handling control and data dependencies of activities, preparing an activity for execution, possibly fault detection and recovery. All these elements of the coordination are carried out in a large, highly dynamic, fault prone environment.

The investigation presented in this paper aims at finding a high level declarative model that is able to provide a coordination framework for advanced workflow enactment where (i) a higher level of autonomy is provided than that of current approaches, (ii) activities in the workflow are able to react the changes of the environment, (iii) enactment itself is distributed and does not assume any centralized support or components, (iv) workflow enactment is able to make decisions on partial information since a consistent view of all activities and resources cannot be assumed, (v) a workflow may contain advanced control structures.

As in many cases, where the problem is complex and difficult to formalize, a nature metaphor may inspire a heuristic solution. To model workflow enactment we have considered a chemical analogy. Activities of a workflow should be scheduled on resources autonomously, without any centralized control, yet maintaining certain properties like molecules react according to laws of nature. We used γ-calculus to formalize the chemical metaphor and establish an inherently distributed declarative coordination framework. The paper presents the basic principles and elementary constructs of such a framework that can be further elaborated and refined realizing sophisticated coordination strategies. Both the chemical metaphor and the application of γ-calculus for enacting distributed scientific workflows are novel approaches.

2 Problem analysis

In the scope of scientific workflows, Process Definition [14] often has two aspects, as discussed in, e.g., [8, 9, 10]. In the definition phase an application to be executed is decomposed into a number of independent activities that are related to each other by data and control dependency. The problem is transformed into the so called *abstract workflow* that expresses the logic of the problem to be solved but it does not contain any specific means how to be executed. Subsequently, *abstract workflow* is transformed into a *concrete workflow* where each logical entity in the abstract workflow is assigned (in other terms: mapped, scheduled) to resource entities (i.e., resources, services, processes, etc.) that can enable the execution. Workflow execution means executing a concrete workflow.

Due to the lack of a strict definition or standard and the diverse concepts in the literature it is necessary to put it in a clear way our interpretation of the reference model in [14]. There is no consensus if transforming an abstract workflow into a concrete one belongs to the Process Design & Definition or the Workflow Enactment. In our view, in order to provide the potential of full dynamicity and autonomy, turning an abstract workflow into a concrete workflow must be carried out at runtime. Therefore, we use the term 'enactment' in a broader sense than usual assuming, that Workflow Enactment comprises mapping an abstract workflow onto a concrete one and control of execution.

In the following, groups of unsolved or partially solved problems, or questionable concepts are listed. At elaborating the workflow enactment model based on a chemical metaphor, the focus was set on these:

1. *Dynamicity.* Any schedule that is prior to runtime has limited or no means at all to adapt to a dynamically changing environment. Resources can appear or disappear any time, abrupt changes may occur in the execution that an a priori schedule cannot cope with.

2. *Validity of information.* Information repositories contain collected and stored data that are often inconsistent with the actual state of resources. A consistent and timely view of all the resources cannot be assumed. Thus, any schedule based on stored information represents a proper schedule for the time when the information was collected. For the same reason, a priori simulated results are not necessary related to the current state of the system.

3. *Cross-effects.* If there are many applications working simultaneously – which is the case for Grids –, identical or similar mapping algorithms may lead to the same results based on the same available information. It means that many independent applications may discover and select the same set of resources leading to crucial imbalance.

4. *Optimal schedule.* Some works describe an attempt for achieving optimal resource scheduling. The optimal scheduling is evolving in time in a dynamic environment. Resources can be added or withdrawn and their characteristics and performance can change continuously. The term optimal or near optimal may be true for a certain point in time but not necessarily for a period of time. In our view the quest for optimal schedule is questionable in large-scale heterogeneous computing.

5. *Centralized schedule.* Many workflow management tools realize a single, centralized workflow engine that can form a bottleneck, hinders scalability and introduces a single point of failure.

6. *Human interaction.* In some cases certain parts of resource scheduling involve human interaction. It necessarily leads to less autonomous, more rigid solutions. Workflow enactment should be independent from any human interaction.

3 Related works

Many aspects of scientific workflow management are well studied and there exist elaborated solutions. There are sophisticated graphical tools for process definition [16], workflow portals [15], advanced languages [10] and process planners [9], just to mention a few examples. On the other hand, workflow enactment itself got less attention. Here we present publications that are aimed at (partly) dynamic enactment and are closer to our approach in some sense.

Deelman et al. [9] propose applying AI techniques for planning an abstract workflow for applications where many large files are transferred between components. The mapping of abstract workflow onto concrete workflow is prior to execution (i.e., it belongs to the Process Definition and not Workflow Enactment) and is based on static information. In a later paper [8] this policy is relaxed and a semi-dynamic mapping is introduced where the workflow is divided into smaller workflows and such partial workflows are mapped at one time. Still, partitioning may pose some constraints on the execution.

The work presented in [13] introduces a Petri-net based model. It has dynamic features: Petri-nets can be refined and on-the-fly resource mapping is also present. It supports more workflow constructs than usual and provides some degree of fault tolerance but the workflow engine is centralized. Petri-nets also make a step towards a formal enactment model

Condor DAGMan [6] provides an on-the-fly dynamic mapping with fault tolerant features. Its functionality is similar to ours, but the approach is however, different. Condor is aimed at providing a high throughput computing environment by maximizing processor efficiency. Its structure and the scheduling policy is fixed, the supported workflow constructs are limited.

The Workflow Enactment Engine (WFEE) [18] is probably the closest to our approach. It is a distributed event-driven workflow enactment engine that realizes fully dynamic, just-in-time resource mapping. It utilizes a tuple space to maintain the state of the computation where tuples are related to events. WFEE makes its scheduling decisions based on stored information.

4 Workflow enactment based on a chemical metaphor

Chemical reactions take place without any centralized control. Atoms or molecules react according to their chemical properties, actual local conditions and universal laws of nature. Each molecule reacts autonomously and each reaction between two molecules is unpredictable, yet the overall properties of the matter are predictable based on chemical laws.

Workflow enactment should behave similarly like a chemical reaction. Resource scheduling and control of execution should be realized in a fully distributed and autonomous way using little or no a priori information but sensitively to changes in the infrastructure.

We propose a model where activities and resources are modeled as molecules. A virtual pool of resources is a chemical solution. The exact number and exact properties of all the molecules in the solution is assumed to be unknown. Activities are modeled as reactive molecules added to the solution. Properties of molecules define the possible reactions. A workflow is composed as compounds of activities or reactions between activities.

We envision workflow enactment as a procedure where these molecules react in an autonomous way, entirely based on actual and local conditions and where nothing is predefined in a static way. The state of the computation is represented by the chemical solution itself – it is a distributed information system by nature. Reactions in the solution realize a decentralized workflow enactment.

The vision of the chemical enactment is turned into a coordination model by formalizing it using γ-calculus. The other cornerstone of the concept to be introduced in a following subsection is the notion of resource quantums that makes possible to represent resources in the chemical model.

4.1 Gamma and the γ-calculus

Due to the domination of imperative languages, most algorithms are expressed sequentially even if they describe inherently parallel activities. Gamma (General Abstract Model for Multiset Manipulation) [5] aimed at relaxing the artificial sequentializing of algorithms. The basic data structure of Gamma is the multiset, i.e., a set that may contain multiple occurrences of the same element and that does not have any structure or hierarchy. Multisets are affected by so called reactions. The effect of a reaction (R_i, A_i), where R_i and A_i are closed functions, on multiset M is to replace in M a subset of elements $\{x_1, ...x_n\}$ such that $R_i(x_1, ...x_n)$ is true, by elements of $A_i(x_1, ...x_n)$ [5]. This model yields a multiset rewriting system where the program is represented by a set of declarative rules. Rules are atomic, they can fire independently from each other (and potentially simultaneously), according to local conditions. A Gamma program is inherently parallel. It has been shown in [5] that some fundamental problems of computer science (sorting, prime testing, string processing, graph algorithms, etc.) can be expressed in Gamma in a more natural way than applying any other sequential semantics. Gamma is a pioneer work among other languages realizing the so called chemical model – chemical in a sense that there is no concept of any centralized control, ordering, serialization rather, the computation is carried out in an indeterministic way according to local conditions.

γ-calculus is a formal definition of the chemical paradigm from which all these chemical models can be derived. In this work we applied γ-calculus therefore, a very short introduction to γ-calculus is presented here based on [2] and [3]. The background and evolution of Gamma and the related models are introduced in [4, 5].

The fundamental data structure of the γ-calculus is the multiset M. γ-terms (molecules) are : variables x, γ-abstractions $\gamma\langle x\rangle.M$, multisets (M_1, M_2) and solutions $\langle M\rangle$ [2]. Note, that molecule is a synonym for all γ-terms. In some cases, if there is no ambiguity, solutions can be referred as molecules.

Juxtaposition of γ-terms is commutative ($M_1, M_2 \equiv M_2, M_1$) and associative ($M_1, (M_2, M_3) \equiv (M_1, M_2), M_3$). Commutativity and associativity are the properties that realize the 'Brownian-motion', i.e., the free distribution and unspecified reaction order among molecules that is a basic principle in the chemical paradigm [3].

γ-abstractions are the reactive molecules that can take other molecules or solutions and replace them by other ones by reduction. Due to the commutative and associative rules, the order of parameters is indifferent; molecules, solutions participating in the reaction are extracted by pattern matching – *any* of the matching ones may react. The semantics of a γ-reduction is

$$(\gamma\langle x\rangle.M), \langle N\rangle \rightarrow_\gamma M[x := N]$$

i.e., the two reacting terms on the left hand side are replaced by the body of the γ-abstraction where each free occurrence of variable x is replaced by parameter N if N is inert, or x is hidden in M, i.e., it only occurs as a solution $\langle x\rangle$ [3].

Besides the associativity and commutativity, reactions occur according to laws of

- locality (also called as chemical law) [3], i.e., if a reaction can occur, it will occur in the same way irrespectively to the environment:

$$\frac{M_1 \rightarrow_\gamma M_2}{M, M_1 \rightarrow_\gamma M, M_2}$$

- solution (also called as membrane law) [3], i.e., reactions can occur in nested solutions or in other words, solutions may contain sub-solutions.

$$\frac{M_1 \rightarrow_\gamma M_2}{\langle M_1\rangle \rightarrow_\gamma \langle M_2\rangle}$$

A *conditional reaction* is a γ-abstraction of form

$$\gamma\langle x\rangle\lfloor C\rfloor.M$$

that can be reduced only if C evaluates to true before the reaction [2].

Atomic capture adds the possibility of reacting with more than one molecule at a time. It is represented by an n-ary γ-abstraction term:

$$\gamma\langle x_1\rangle, \langle x_2\rangle, ...\langle x_n\rangle.M$$

that can be reduced in a single atomic step if it can be matched by n appropriate terms [2].

Note, that γ-calculus is a *higher order* model, where abstractions – just like any other molecules –, can be passed as parameters or yielded as a result of a reduction. The higher order property makes it possible to express usual programming types (e.g. boolean, integer, etc.) within the γ-calculus.

The γ-calculus shows some similarities with the λ-calculus. Like the λ-calculus establishes the theoretical foundation for functional languages, the γ-calculus plays the same role for languages realizing the chemical paradigm. The notion of variables and abstractions, the concept of reduction (rewriting) and the syntax are very close to each other, yet the semantics differs significantly. While the λ-calculus is a sequential and deterministic model, the γ-calculus is inherently parallel and nondeterministic.

4.2 The concept of resource quantum

The chemical enactment model can solve some of the problems listed in Section 2, but the problem of validity of information and cross-effects stem from the way how information about resources is collected and queried. In current practices characteristic parameters of resources are stored in information systems (databases). While some parameters of a resource are fixed, some others may change in time. Therefore, stored information cannot be entirely descriptive and precise. Furthermore, even if the information retrieved from the database is precise and timely, there is no guarantee that a certain job will get the described service, quality and performance. Many independent applications can query the same information system thus, they may discover and select the same resource based on the same retrieved information, causing the cross-effects.

To solve these problems we introduce the notion of resource quantums to the chemical coordination model. Each quantum represents a certain, guaranteed service on behalf of the resource. Activities may apply for such a quantum and if granted, they have the exclusive right to use the given quantum. Quantums represent not just information about a certain resource capacity but also they guarantee an access to it. They can be considered as a ticket for accessing a resource. Activities compete for the tickets and cannot share them.

Resource owners may offer all or parts of their capacities or services in quantums manifested as tickets. The strategy how quantums are established and how tickets are emitted is at the resource owners' discretion. A ticket entitles the activity that bears it to use the quantum represented by the ticket – after the necessary authentication and authorization procedures. For instance, clusters are common resources in grid computing. A cluster of 16 computing nodes may be transformed into tickets in any of the following forms according to a chosen strategy (details like network or storage are omitted here but can be added easily): 16 tickets as 1 processor resource each, 1 ticket as 16 processor resource, 3 tickets as 8,4,4 processor resource each, and so on. To achieve optimal resource utilization, it is also possible, that more tickets, or tickets representing more capacity are emitted. The strategy is similar to that of airline companies that actually sell more tickets for a certain flight than the number of seats it has in order to achieve better profiting.

Since a ticket can belong to at most one activity – just like a molecule may be part of one reaction at a time –, the notion of resource quantums perfectly supports the chemical coordination model.

5 Workflow enactment based on the γ-formalism

In this section we show how the informal idea of chemical workflow enactment can be formalized in the γ-calculus. The entire chemical coordination takes place in a single solution. Within this solution, both workflow structure (activities) and resources (quantums) are represented as γ-expressions that at the same time define their possible interactions, i.e., the enactment procedure as well. For the sake of simpler presentation we do not define a strict syntax here, some ideas are represented symbolically.

5.1 Modeling resources using γ-calculus

In our approach all resources, represented as quantums, are modeled as a *chemical solution*. Resource solutions are sub-solutions within the coordination solution. Attributes of the resource form molecules within the solution. The general form of a solution describing a resource quantum is:

$\langle a_1 : v_1, a_2 : v_2, ...a_n : v_n \rangle$

where each $a_i : v_i$ forms an attribute-value pair. The exact number of attributes and their meaning depends on the resource; there is no need for fixed format. Yet, there are some attributes that every resource should possess like identification, contact address, optionally further information about contact protocols, validity, security and so on. Their order is absolutely irrelevant according to the logic of the chemical model and the commutativity rule of γ-expressions. For example:

- $\langle id : R1,\ type : comp,\ proc : 16,\ OS : Linux,\ mem : 32,\ installed : equsolver, ... \rangle$ describes a computational resource identified as R1 that has 16 processors, Linux operating system, 32MB of memory and a certain equation solver software installed. (It is assumed that the default capacity unit is Gbytes for disk and Mbytes for memory and thus these capacity units do not appear in the γ-expression.)
- $\langle id : R4,\ OS : Linux, ... \rangle$ – a certain resource identified as $R4$ that has Linux operating system. The description does not tell the number of processors, yet it is fully valid and correct. Activities sensitive to the number of processors will specify the $proc$ tag in their parameter pattern and would not match this resource – others may accept.

5.2 Representation of a single activity

In our model a single activity is treated as an atomic unit that is executed and some results are returned. The exact working behavior or the structure of the activity is irrelevant at the coordination level. It is assumed that activity A ready for execution is represented symbolically as:

$$\textbf{execute}\ A\ \textbf{on}\ resource(s)\ \textbf{using}\ parameter(s)$$

where **execute** can be considered as a primitive procedure, i.e., it is not defined further in terms of γ-calculus. The entire chemical metaphor is aimed at *coordinating* the enactment of the workflow but not at *executing* its activities. Once a γ-expressions has been reduced to this primitive, it is taken out of the chemical solution and passed to some external processes that realize the physical execution. The technical realization of the execution may vary since it depends on the physical environment (if it is a cluster, a Grid, a service oriented architecture, components, etc.) After the activity has been executed externally, results from the execution and the redeemed resources must be put back to the chemical solution.

The transfer between the chemical coordination model and the physical execution environment can be expressed in the γ-calculus by the following assumption: **execute** is reduced externally to a chemical solution holding the result of the execution, and some other solutions representing the released resources, i.e.,

$\textbf{execute}\ A\ \textbf{on}\ resource\ \textbf{using}\ parameters$

$\rightarrow \langle A : result, ... \rangle, \langle id : resource, ... \rangle, ...$

Note that $result$ is a symbolic representation here. It may be anything: a number, a string, a complex structure, a pointer to a file, a pointer to any other data items or oppositely, holding no information at all just signaling the termination. Just like resources do not have a fixed form of representation, apart from the obligatory $activity : result$ pair they may contain other information as well related to the execution, e.g., error codes, number of iterations, resources that were used and so on.

5.3 Resource dependencies

Obviously, an activity may and does require resources to perform its task. We define the general form of resource dependency expressed in the γ-calculus as:

$$\gamma \langle a_1 : v_1, a_2 : v_2, ... a_n : v_n \rangle . \mathbf{execute}\ A\ \mathbf{on}\ v_i$$

where a_i is the mandatory identifier tag. The attributes specify a resource profile that will be found in the solution by pattern matching. This active molecule may capture *any* resource that has the specified profile, i.e., those solutions where $attribute : values$ pairs can be matched. This general form may involve variables and the universal matching symbol, ω, as it is introduced in the following simple examples.

Activity A requires a computing node with 4 processors and 128M memory is represented as

$\gamma\ \langle id : r, procno : 4, memory : 128, \omega \rangle .$
$\mathbf{execute}\ A\ \mathbf{on}\ r$

The same activity requires a computing node with Linux OS and disk space of at least 30G

$\gamma\ \langle id : r, OS : Linux, disk : x, \omega \rangle \lfloor x > 30 \rfloor .$
$\mathbf{execute}\ A\ \mathbf{on}\ r$

Resource dependencies may involve multiple resources and more complex constraints, e.g., activity A requires a computing node with disk space of at least 30G and a computing node with at least 128M memory so that the OS of the two nodes are the same:

$\gamma\ \langle id : r_1, OS : x, disk : y, \omega_1 \rangle,$
$\langle id : r_2, OS : x, memory : z, \omega_2 \rangle \lfloor y > 30, z > 128 \rfloor .$
$\mathbf{execute}\ A\ \mathbf{on}\ r_1, r_2$

5.4 Representation of dependent activities

Let us assume the problem to be solved is decomposed at the Process Design & Definition [14] phase into $k \geq 1$ stand-alone activities represented as set $A = \{A_1, A_2, ...A_k\}$. An activity may depend on the result of other activities (data dependency) or simply triggered by the termination of another activity (control dependency). Dependencies are represented by a binary relation $\rightarrow$ on $A \times A$. In

our model they are expressed as a parameter in the head of the γ-abstraction. (In the following A_i is used for the activity and a_i for the corresponding γ-term.)

If activity A_i depends on another activity A_j, i.e., they are executed in *sequence* : $A_j \rightarrow A_i$, the corresponding γ-expression is:

$$\gamma\langle A_j : x, \omega\rangle.\mathbf{execute}\, A_i \mathbf{using}\, x$$

i.e., execution of A_i requires a solution containing the result from A_j.

If activity A_l depends on n other activities, i.e., realizes a *synchronization*: if $\exists n \geq 1$ so that $A_{s_1} \rightarrow A_l, A_{s_2} \rightarrow A_l, ...A_{s_n} \rightarrow A_l$,where $s_i \in \{1, 2, ...k\}, s_i \neq l, s_i \neq s_j$ for $i \neq j$, it is represented as:

$\gamma\ \langle A_{s_1} : x_1, \omega_1\rangle, \langle A_{s_2} : x_2, \omega_2\rangle, ...\langle A_{s_n} : x_n, \omega_n\rangle.$
$\mathbf{execute}\, A_i \mathbf{using}\, x_1, x_2, ...x_n$

In case of a *parallel-split*, i.e., if $\exists m > 1 : A_l \rightarrow A_{s_1}, A_l \rightarrow A_{s_2}, ...A_l \rightarrow A_{s_m}$ where $s_i \in \{1, 2, ...k\}, s_i \neq l, s_i \neq s_j$ for $i \neq j$, then a γ-abstraction that multiplies the result m times is added to the body ofA_l:

$(\gamma\ \langle\langle A_l : x, \omega\rangle\rangle.\langle A_l : x, \omega\rangle, \langle A_l : x, \omega\rangle, ...\langle A_l : x, \omega\rangle),$
$\langle \mathbf{execute}\, A_l\rangle$

Note, that the execution of A_l takes place in a solution ($\langle \mathbf{execute}\, A_l\rangle$) therefore, its result will be a sub-solution within the solution ($\langle\langle A_l : r_{A_l}\rangle\rangle$) and reducing further:

$\rightarrow (\gamma\ \langle\langle A_l : x, \omega\rangle\rangle.\langle A_l : x, \omega\rangle, \langle A_l : x, \omega\rangle, ...\langle A_l : x, \omega\rangle),$
$\langle\langle A_l : r_{A_l}\rangle\rangle$

$\rightarrow \langle A_l : r_{A_l}\rangle, \langle A_l : r_{A_l}\rangle, ...\langle A_l : r_{A_l}\rangle$

the result solutions are multiplied. This construct is necessary otherwise, the result molecule ($\langle A_l : r_{A_l}\rangle$) could react with any of the depending activities *before* the multiplication takes place. The double nested solutions "hide" the result.

These three γ-expressions realize sequence, synchronized merge and parallel split constructs – many workflow management concepts do not go beyond these ones. In the followings, using the same formalism, advanced constructs are introduced.

Conditional. Conditional workflow constructs can be evidently described by conditional γ-expressions.

$\exists A_l \rightarrow A_i, A_l \rightarrow A_j$. Depending on condition C either A_i or A_j is executed. In this case the corresponding molecules:

$a_i = \gamma\langle A_l : x, \omega\rangle\lfloor C\rfloor.A_i$
$a_j = \gamma\langle A_l : x, \omega\rangle\lfloor \neg C\rfloor.A_j$

In the same way multiple choices can be realized provided that C_1, C_2 and C_3 are mutually disjoint.

$a_i = \gamma\langle A_l : x, \omega\rangle\lfloor C_1\rfloor.A_i$
$a_j = \gamma\langle A_l : x, \omega\rangle\lfloor C_2\rfloor.A_j$
$a_k = \gamma\langle A_l : x, \omega\rangle\lfloor C_3\rfloor.A_k$

Split. There is number of potential follow-up activities for A_l, $\exists n \geq 1$, $A_l \rightarrow A_{s_1}$, $A_l \rightarrow A_{s_2}$,... $A_l \rightarrow A_{s_n}$, $s_i \in \{1, 2, ...k\}, s_i \neq l, s_i \neq s_j$ for $i \neq j$. from all A_{s_i} **exactly** p, $n \geq p \geq 1$, may be activated after termination of A_l. This case can be solved similarly to *parallel-split* but result molecules are multiplied p times instead of n.

Exclusive choice. Exclusive choice, as a special case for split: from all A_{s_i} **one and only one** may be activated after termination of A_l. By combining it with conditional, *explicit choice* can be realized. There is no need for any additional construct, execution semantics of the γ-calculus coincides with this case, i.e., if A_l produces just one result molecule, it can react with one and only one of the dependent activities only.

Merge. There is a number of antecedent activities that may produce input for A_l: $\exists n \geq 1$, $A_{s_1} \rightarrow A_l$, $A_{s_2} \rightarrow A_l$,... $A_{s_n} \rightarrow A_l$, $s_i \in \{1, 2, ...k\}, s_i \neq l, s_i \neq s_j$ for $i \neq j$. From all A_{s_i}, the termination of **exactly** p, $n \geq p \geq 1$, may activate A_l. $p = 1$ means that the termination of **any** A_{s_i} may activate A_l.

In this case results from activities A_{s_i} are put into a named solution $pool_{a_l}$, from which activity A_l can choose randomly. Without the extra sub-solution random choice and explicit naming of the possible input activities would be in conflict. Activity A_l is represented as

$\gamma\ (pool_{a_l} = \langle\langle x_1 : y_1, \omega_1\rangle, \langle x_2 : y_2, \omega_2\rangle, ...\langle x_p : y_p, \omega_p\rangle, \omega\rangle).$
$\mathbf{execute}\, A_l\, \mathbf{using}\, y_1, y_2, ...y_p$

where $x_i, y_i, 1 \leq i \leq p$ are variables.

Furthermore, there is an initially empty solution $pool_{a_l} = \langle\rangle$. For each A_{s_i} the following rule that puts results into the named solution is added to the system:

$\gamma\ \langle A_{s_i} : result, \omega_{s_i}\rangle, (pool_{a_l} = \langle\omega\rangle).$
$pool_{a_l} = \langle\omega, \langle A_{s_i} : result, \omega_{s_i}\rangle\rangle$

If it is known that one and only one of A_{s_i} may be active but it is unknown which one then A_l can be represented simply as a set of molecules:

$\gamma\langle A_{s_1} : x, \omega\rangle.\mathbf{execute}\, A_l\, \mathbf{using}\, x$
$\gamma\langle A_{s_2} : x, \omega\rangle.\mathbf{execute}\, A_l\, \mathbf{using}\, x$
...
$\gamma\langle A_{s_n} : x, \omega\rangle.\mathbf{execute}\, A_l\, \mathbf{using}\, x$

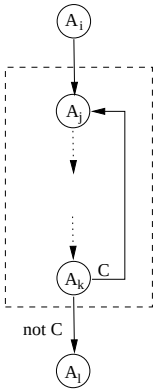


Figure 1: Loop structure

Structured loop. The loop can be constructed by a merge and a conditional structure furthermore, the core of the loop is added to the solution dynamically in each iteration. Let us assume the core of the loop begins and ends with a sequential activity (if not, this condition can be ensured by introducing pseudo nodes). Thus, let us assume, the boundaries of the core is between $A_i \rightarrow A_j$ and $A_k \rightarrow A_l$ (Figure 1.)

There are three key elements in the loop representation: the first activity in the loop, in this case A_j, the core (the activities to be repeated), i.e., $A_j ... A_k$ and the activities following the last one of the core: A_l and A_j.

The first activity of the core must realize a merge where only one of the preceding activities may be active (i.e., initial entry into the loop body or re-invocation at iteration). Additionally, the re-entry branch is conditional. It is represented by two molecules:

$$initial_entry = \gamma\langle A_i : x, \omega\rangle.\mathbf{execute}\, A_j\, \mathbf{using}\, x$$
$$re_entry = \gamma\, \langle A_k : x, \omega\rangle \lfloor C(x) \rfloor.$$
$$core,\, \mathbf{execute}\, A_j\, \mathbf{using}\, x$$

where $core$ is all activity to be iterated.

The core. The core of the loop is composed of all activities to be iterated $A_{j+1} ... A_k$ and the re_entry branch of A_j.

$$core = re_entry, a_{j+1}, ... a_k$$

The core is put into the solution each time the re_entry branch is executed – a notable example for dynamic behaviour in the γ-model.

Conditional activities. After the execution of the loop core, either the next activity outside the core or the first activity of the core would be executed. They are discriminated by a conditional construct.

The first half of the conditional has already been introduced as re_entry whereas the second half is

$exit = \gamma\langle A_k : x, \omega\rangle\lfloor\neg C(x)\rfloor.\mathbf{execute}\ A_l\ \mathbf{using}\ x$

The exact meaning if condition $C(x)$ is not specified here: it is assumed that the captured result solution $\langle A_k : x, \omega\rangle$ contains the necessary information. The most general case was shown here that can be refined for various cases, provided that the result solution contains the required molecules, e.g.:

$exit = \gamma\langle A_k : x, iteration : i, \omega\rangle\lfloor\neg(i < 12)\rfloor.$
$\quad\mathbf{execute}\ A_l\ \mathbf{using}\ x$

$exit = \gamma\langle A_k : x, errorcode : e, \omega\rangle\lfloor\neg(e == 0)\rfloor.$
$\quad\mathbf{execute}\ A_l\ \mathbf{using}\ x$

5.5 Combining resource and data/control dependencies

So far resource and data/control dependencies were handled separately albeit, as it can be seen, they do not differ at all in terms of the γ-calculus. In fact they both represent conditions that all must be met before the activity can proceed, and an active molecule representing an activity can have any number of these dependencies simultaneously. Yet, they can be written in three different forms that yield three different evaluation strategies. (For the sake of simplicity an oversimplified symbolic notation is used for resources applying R for a resource profile.)

- Data/control and resource dependencies must be satisfied simultaneously (atomic capture)

  $\gamma\ (\langle A_j : x, \omega_a\rangle, \langle id : r, R, \omega_r\rangle).$
  $\mathbf{execute}\ A_i\ \mathbf{on}\ r\ \mathbf{using}\ x$

- First activity A_i must be enabled according to data/control flow rules then it may search for an appropriate resource. Useful in cases where it is not known of a particular activity will be ever enabled.

  $\gamma\ \langle A_j : x, \omega_a\rangle.$
  $(\gamma\langle id : r, R, \omega_r\rangle.\mathbf{execute}\ A_i\ \mathbf{on}\ r\ \mathbf{using}\ x)$

- An appropriate resource must be found before data/control flow may enable the activity. It may gain efficiency since the search for a resource may overlap with computations that will enable A_i.

  $\gamma\ \langle id : r, R, \omega_r\rangle.$
  $(\gamma\langle A_j : x, \omega_a\rangle.\mathbf{execute}\ A_i\ \mathbf{on}\ r\ \mathbf{using}\ x)$

Apart from realizing different strategies, breaking a single line of multiple conditions into several nested conditions reduces the search complexity. [17] contains further complex examples for various dependencies.

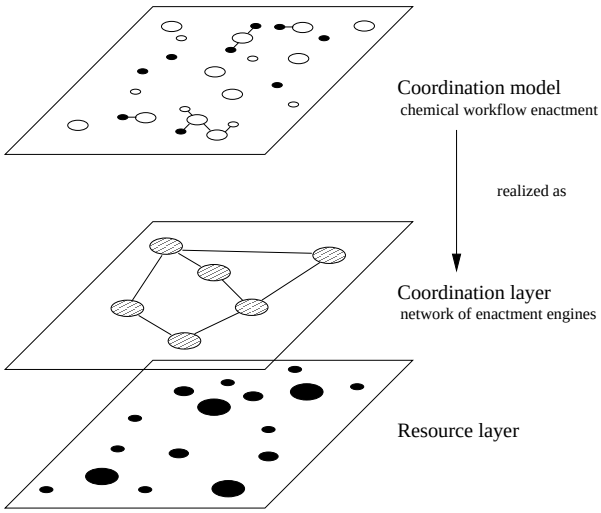


Figure 2: Concept of the workflow architecture

6 Concept for realization of the chemical enactment model

The highest level of abstraction in Figure 2 is the world of the chemical metaphor. It does not exist physically, but defines the principles of the chemical enactment model. The chemical solution contains all molecules and sub-solutions related to the structure of the workflow, work to be executed, the work that has been done, the available resources and rules for controlling the execution mechanism. Properties of the molecules define the possible reactions between them. The entities and rules of the chemical metaphor are formally modeled by the γ-calculus as it was introduced so far.

The chemical metaphor can be realized as a two layered structure: a coordination layer and the coordinated resource layer. The coordination layer, organized as a network of workflow engines is the one that brings into effect the chemical enactment model. To put in other words, the chemical metaphor is the specification of the execution model whereas the coordination layer is its implementation. Each engine has the same functionality and structure and can work independently from each other. It is important to notice that the coordination layer *does not execute* any of the activities, it just controls the progress of workflows.

Activities of the workflow are executed at the resource layer. The resource layer contains all the physically existing entities that are (potentially) utilized by the applications. The model is general and does not restrict the possible set of entities: they may be (virtualized) hardware/software resources, services, data elements, results of other applications, components, etc. The resource layer represents the lowest level of abstraction in this model but entities are not necessarily low level physical resources. From our point of view a Grid information service [7], for instance, belongs to the resource layer, yet it can be itself a complex and abstract entity, but further details of abstraction are irrelevant here.

6.1 The workflow engines

The chemical solution containing all the activity, resource and control molecules and sub-solutions is divided into disjunct volumes. A workflow engine is assigned for a certain volume of the solution (Figure 3). A molecule or sub-solution can belong to at most one volume at a time nevertheless, they can move between volumes freely. In this sense volumes are not static.

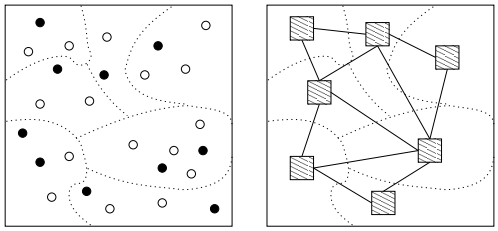


Figure 3: Network of workflow engines

The task of a workflow engine is interpreting γ-terms in that particular volume. In such a way, the most straightforward realization of a workflow engine is an abstract machine for the γ-calculus. Nevertheless, this interpretation can be realized in multiple ways. For example, an active molecule

$$\gamma P\lfloor C\rfloor.M$$

may be interpreted Gamma-style [2, 3, 5] as

$$\mathbf{replace}\ P\ \mathbf{by}\ M\ \mathbf{if}\ C$$

that brings it close to $if\text{-}then$ rules, for which advanced pattern matching algorithms of production systems, like RETE [11], an "efficient method for comparing a large collection of patterns to a large collection of objects" may serve as a foundation for implementation.

Engines keep reacting the molecules, i.e., interpreting the rules until some of them is reduced to an **execute** primitive that can be considered as a built-in procedure realizing an exit point from the engine level to the resource one. In this case the **execute** primitive is removed from the solution and passed to a process that realizes the reservation of the physical resources, controls file transfers, submits the jobs, etc., corresponding to the activity and the parameters. The same process receives results from physical entities, turns them into molecules and puts them back to the engine. Similarly, physical entities can submit resource tickets or error reports to the engines via appropriate interfaces. In such a way the engine is independent from any actual technical realization – interface processes that realize the transfer between physical entities and their γ-representation provide the necessary separation between the workflow engines and various implementations of a heterogeneous distributed system forming the resource layer.

An engine realizes the chemical behaviour for a particular volume, i.e., can control the execution over a subset of molecules which, obviously improves locality and decreases complexity. If molecules can diffuse between engines, potentially each molecule may react with any other over

the time. Since the reaction between the molecules is of random order, this way of distributing molecules maintains correctness just perturbs its temporal behavior. Information exchange between engines resembles the Brownian-motion, i.e., some molecules are transferred between them according to some strategy, and apart from this loose coupling they are completely independent from each other realizing a fully decentralized coordination layer.

7 Conclusion

Enabling the execution of a workflow in a large-scale heterogeneous environment, such as Grids, is a complex task. Principles of a chemical reaction show considerable similarities with requirements of workflow enactment therefore, we envisioned an enactment model based on a chemical metaphor. The research presented in this paper was aimed at establishing a declarative coordinating framework, and has proven that such an informal vision can be turned into an enactment model using the γ-calculus. The paper presented the elementary concepts and principles of the framework that can be further elaborated towards an implementation.

The fact that all entities in the model, activities, resources and control, are represented using the *same* γ-formalism and, the formalism defines the execution semantics as well is unique among all workflow management systems. It was demonstrated that all basic and most advanced workflow patterns [1] and the notion of resource quantums can be expressed in a declarative way using the γ-calculus. Some further details on control can be found in [17]. This model is superior to any other approaches today in the following points: (i) the uniform declarative formalism provides a possibility for describing arbitrarily complex workflows and advanced coordination strategies at the same time, (ii) the model is not a rigid script or graph rather, a set of rules and facts where logical ordering exist only, any components can be added/withdrawn/modified at run-time and, (iii) the execution semantics is given without further definitions.

In this paper we presented the coordination model for chemical enactment independently from any actual technical details. The realization and test of the model is beyond the scope of the current research work. It was outlined that the abstract chemical enactment model can be realized by a network of engines. Based on the clear semantics of the execution model, the most straightforward realization – disregarding performance issues but maintaining correctness – is simply interpreting γ-terms. On the other hand the formal framework provides a basis for various realizations that can be derived informally or formally by refinement.

Although the γ-calculus is a higher-order formalism, the model presented here used first-order features only. Thus, there is an unexplored potential in exploitation of higher-order constructs that may enable fully dynamic workflows (i.e., any activity can be captured and replaced at run-time) and sophisticated fault-tolerance.

8 Acknowledgments

The research presented in this paper was carried out while Zs. Németh, from SZTAKI, was hosted by IRISA-INRIA, Rennes, supported by the ERCIM Fellowship programme. Authors express their gratitude to Yann Radenac and Jean-Pierre Bânatre for their help with the γ-calculus.

References

[1] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, A.P. Barros: Workflow Patterns. BETA Working Paper Series, WP 47, Eindhoven University of Technology, Eindhoven, 2000.

[2] J.-P. Banâtre, P. Fradet, Y. Radenac: Principles of Chemical Programming. Fifth International Workshop on Rule-Based Programming (RULE'04), Electronic Notes in Theoretical Computer Science.

[3] J.-P. Banâtre, Y. Radenac, P. Fradet: Chemical Specification of Autonomic Systems. Proc. of the 13th Int. Conf. on Intelligent and Adaptive Systems and Software Engineering (IASSE'04)

[4] J.-P. Banâtre, P. Fradet, D. Le Métayer: Gamma and the Chemical Reaction Model: Fifteen Years After. Multiset Processing, LNCS 2235, Springer 2001, pp. 17-44.

[5] J.-P. Banâtre, D. Le Métayer: Programming by Multiset Transformation. Communications of the ACM, Vol. 36, No. 1, January 1993, pp. 98-111.

[6] Condor DAGman
`http://www.cs.wisc.edu/condor/dagman/`

[7] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman: Grid Information Services for Distributed Resource Sharing. Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.

[8] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. Su, K. Vahi, M. Livny: Pegasus: Mapping Scientific Workflows onto the Grid. Across Grids Conference 2004, Nicosia, Cyprus, 2004

[9] E. Deelman, J.Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, S. Koranda: Mapping Abstract Complex Workflows onto Grid Environments. Journal of Grid Computing, Vol. 1, No. 1, pp. 25-39, 2003

[10] T. Fahringer, J. Qin, S. Hainzer: Specification of Grid Workflow Applications with AGWL: An Abstract Grid Workflow Language. IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005), Cardiff, May 9-12, 2005.

[11] C. L. Forgy:Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence Vol. 19, No. 1, Sept. 1982, pp. 17-37.

[12] I. Foster, C. Kesselmann (eds.): The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, 2004

[13] A. Hoheisel, U. Der: Dynamic Workflows for Grid Applications. Proceedings of the 3rd Cracow Grid Workshop, 2003.

[14] D. Hollingsworth: The Workflow Reference Model 19.Jan.95. Workflow Management Coalition, Doc. No. TC00-1003, 1995.

[15] R. Lovas, G. Dózsa, P. Kacsuk, N. Podhorszki, D. Drótos: Workflow Support for Complex Grid Applications: Integrated and Portal Solutions. Second European AcrossGrids Conference, AxGrids 2004, Nicosia, Cyprus, Springer LNCS 3165, pp. 129-138

[16] S. Majithia, M. S. Shields, I. J. Taylor, I. Wang: Triana: A Graphical Web Service Composition and Execution Toolkit. Proceedings of the IEEE International Conference on Web Services (ICWS'04), pp. 514-524. IEEE Computer Society, 2004.

[17] Zs. Németh, C. Pérez, T. Priol: Scientific Workflow Enactment Based on a Chemical Metaphor. INRIA Report, 2005.

[18] J. Yu and R. Buyya: A Novel Architecture for Realizing Grid Workflow using Tuple Spaces, Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID 2004), Pittsburgh, USA, IEEE Computer Society Press