Grid Workflows Specification and Verification

P. KURDEL, J. SEBESTYÉNOVÁ
Institute of Informatics
Slovak Academy of Sciences
Bratislava, Dúbravská cesta 9
SLOVAKIA
{peter.kurdel,sebestyenova}@savba.sk http://www.ui.sav.sk

Abstract: - Grids are being adopted and developed in several scientific disciplines that deal with large-scale collaboration, massive distributed data, and distributed computing problems. The service orchestration is a problem of making multiple services coordinate themselves and communicate in an orderly fashion so as to accomplish a task more complex than the single tasks provided by the individual composing services. Composition is devoted to the aim of connecting services in a collaborative fashion. A Grid workflow system is a type of application-level Grid middleware that is supposed to support modelling, redesign and execution of large-scale processes. A grid workflow can be represented by a grid workflow graph, where nodes correspond to activities and edges correspond to dependencies between activities, called flows. Verification is usually based on an extension of a kind of formal method. Grid workflow verification and validation must be conducted so that we can identify any violations of the correctness in workflow specification and consequently remove them in time.

Key-Words: - Distributed computing, Grid infrastructure, Service orchestration, Workflow management system, Verification, Web portal

1 Introduction

Service-Oriented Computing [23] utilises services as fundamental elements for developing distributed applications. One goal of Service-Oriented Architectures [8] is to integrate and compose services that are deployed on heterogeneous middleware paradigms. Composition is devoted to the aim of connecting services in a collaborative fashion [15]. Web services encapsulate information, software or other resources, and make them available over the network via standard interfaces and protocols. Aggregating the functionality provided by simpler ones may create complex web services. This is referred to as service composition and the aggregated web service becomes a composite web service.

Grid systems are composed of computational elements and disk storage. Next-generation grid solutions must include support for accessing instruments and sensors. [13] Remote control of, and data collection from, instruments and sensors ware parts of the initial Grid concept.

The service orchestration problem is a problem of making multiple services coordinate themselves and communicate in an orderly fashion so as to accomplish a task more complex than the single tasks provided by the individual composing services [20]. Web services make information and software available programmatically via the Internet and may be used as building blocks for applications. A composite web service is one that is built using multiple component web services and is typically specified using an appropriate language. Once its specification has been developed, the composite service may be orchestrated either in a centralized or in a decentralized fashion. Decentralized orchestration offers performance improvements in terms of increased throughput and scalability and lower response time. However, decentralized orchestration also brings additional complexity to the system in terms of error recovery and fault handling. Further, incorrect design of a decentralized system can lead to potential deadlock or non-optimal usage of system resources.

The term workflow [3] can be defined as the orchestration of a set of activities to accomplish a larger and sophisticated goal. Significant research has been conducted in recent years to automate these activities using advanced workflow management tools. Some of the most popular and sophisticated workflow systems available in market include Websphere MQ Workflow [21], Staffware [19] etc. These products offer extensive functionality and support a variety of workflow patterns.

2 Grid Workflow Systems

In Grid architecture [6], a Grid workflow system is a type of application-level Grid middleware that is supposed to support modelling, redesign and execution of large-scale processes in a variety of complex scientific and business applications such as climate modelling, astrophysics, high-energy physics, structural biology and chemistry, medical surgery, disaster recovery,

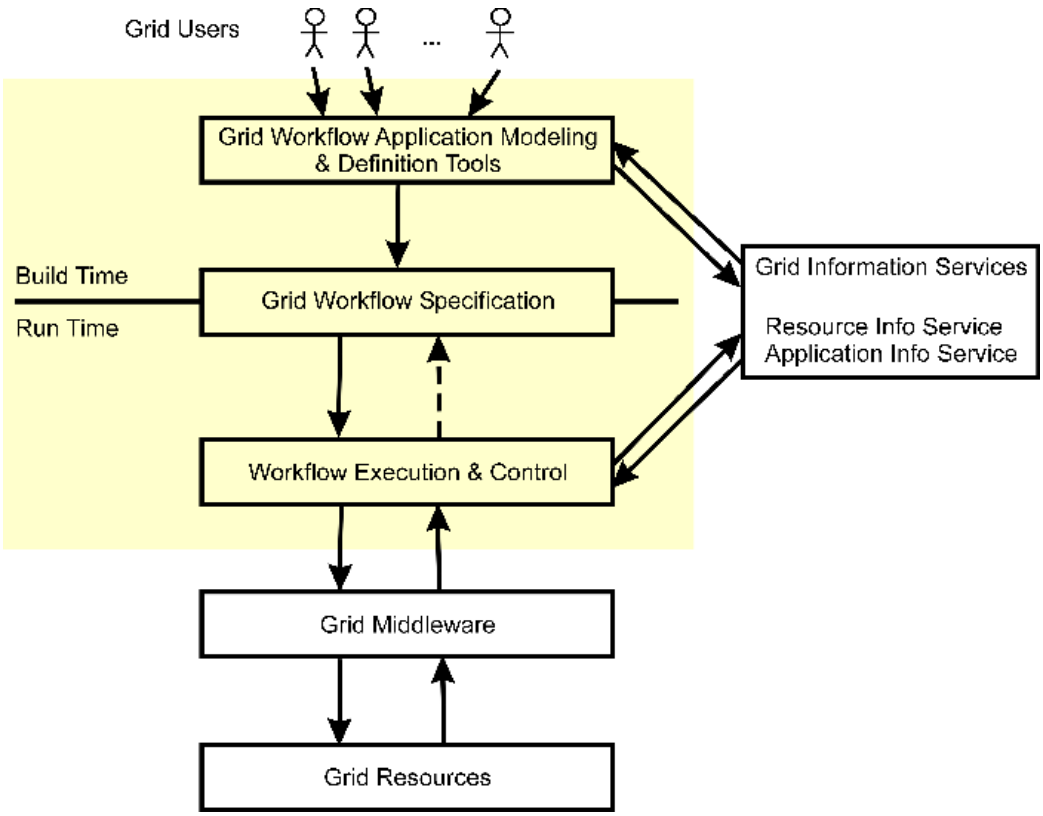


Fig. 1 Grid workflow management system

international banking, insurance, international stock market modelling and control.

Existing workflow systems for web services can technically be used for composing grid services, as grid services are a special kind of web services. There are several problems with current workflow systems for web services, which make them less suited for the grid services context. Most workflow languages for web services do not have a clearly defined semantics.

Another important topic regarding workflow languages for grid services is a support for specific requirements that are typically of importance with high-performance computing: in such an environment, it is common that large amounts of data need to be transferred from one step in a workflow to another. Special attention should be directed to how this happens: it would, for example, be unacceptable if all these data were transferred to a central workflow coordinator before being transferred to a next step. This is current practice in BPEL4WS [4] workflow systems.

The whole working process of a Grid workflow system can be divided into three stages: build-time, run-time instantiation, and run-time execution (see Fig.1). The build-time functions [24] are concerned with defining and modelling workflow tasks and their dependencies, while the run-time functions are concerned with managing workflow executions and interactions with Grid resources for processing workflow applications.

2.1 Workflow Specification

Workflow modelling [11] is similar to process modelling. In a workflow context, tasks are basic work units that collectively achieve a certain goal. The collective nature shows various types of process dependencies. Therefore, whatever language is used to specify workflows, it should be sufficiently powerful to capture those dependencies: sequential order, parallelism, iteration, choice, synchronisation etc.

Workflow specification contains information [14] formally describing various aspects of a workflow, for example, the process aspect, information aspect and organisation aspect. The process aspect describes the structure of a workflow using such entities as activities, subprocesses (a process that is enacted or called from another process or subprocess), as well as flows (data and control flows) between them. The information aspect addresses what are input and output by activities in a workflow. The organisation aspect defines entities belonging to an organisation or a virtual organisation as well as the relationship between them.

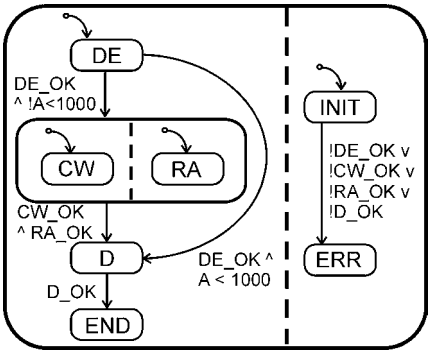
Workflow specification should be capable of expressing at least the following essential workflow concepts: Properties of tasks (pre and post conditions,

redo-ability - whether a task can be redone etc.), Information flows between tasks and information of a more persistent nature (e.g. external databases), Execution dependencies between tasks (also referred to as control flow), Capacities of tasks (might refer to storage, to throughput, or to numbers of active instances). Generally speaking, workflow specifications need not pay attention to task functionality as focus is on the coordination of tasks.

Every workflow language needs to define the basic activities that it supports (e.g. invoking grid services synchronously and/or asynchronously, and assigning and retrieving variables), and how these activities can be ordered.

To formally check [7] the consistency between complex e-science and e-business processes and corresponding grid workflow specifications, firstly, we must capture and formally represent the complex processes.

Based on the directed graph concept, a grid workflow can be represented by a grid workflow graph, where nodes correspond to activities and edges correspond to dependencies between activities, called flows. A novel approach for graphically modelling and describing Grid workflow applications based on the Unified Modelling Language (UML) is presented in [17]. The approach provides a graphic representation of Grid applications based on UML standard that is more amenable than pure textual-oriented specifications, such as XML. UML activity diagrams are special cases of UML state diagrams, which in turn are graphical representations of state machines. The state machine formalism as defined in the UML, is a variant of Harel's statecharts. State machines are transition systems whose arcs are labelled by *event-condition-action* rules. A credit-request example of workflow-style Statechart is given in Fig. 2 (nodes represent activities and edges represent data-flow).



DE … data entry
CW … credit worthiness
RA … risk assessment
D … decision
INIT … initial state
ERR … error state
A … amount

Fig. 2 Credit request Statechart

2.2 Workflow Patterns

Differences in features supported by various contemporary commercial workflow management systems point to different suitability and different levels of expressive power. There is a need [1] to systematically address workflow requirements, from basic to complex. Requirements for workflow languages are indicated through workflow patterns (see Fig. 3), which can be divided into seven categories:

- basic control patterns (sequential, parallel, selective and iterative structures are defined in the workflow reference model WfMC [22])
- advanced branching and synchronization patterns
- structural patterns (loop, iteration, cycle, implicit termination)
- patterns involving multiple instances
- state-based patterns
- cancellation patterns (cancel activity).
- data patterns (data visibility, data interaction, data transfer, data-based routing).

State-based patterns. In real workflows, where human and material resources are not always available, activities are more often in a waiting state than in a processing one [9]. In the following patterns a distinction is made between the moments when an activity is enabled and when it starts running:

Deferred choice: One among several branches is chosen based on an external event, which is not necessarily available when this point is reached, and the choice between them is delayed until an external signal is received (implicit XOR-split).

Interleaved parallel routing: A set of activities need to be executed in an arbitrary order. Each activity in the set is executed exactly once. The order between the activities is decided at run-time. In any case, no two activities can be active at the same time.

Synchronisation patterns. The discriminator is a point in a workflow that waits for one of its incoming branches to complete before activating the subsequent activity.

The *N-out-of-M join* is a point in a workflow where M parallel branches converge into one. The outgoing branch should be started once N incoming branches have completed. Completion of all remaining branches should be ignored (discriminator is a 1-out-of-M join).

Multiple instances requiring synchronization: A point in a workflow where an activity A is enabled multiple times. The number of instances of A that need to be enabled is known only when the point is reached. After completing all the enabled in-stances of A, an instance of an activity B has to be executed.

Dynamic invocations. Within an activity diagram, it is possible to specify that multiple invocations of an action or subactivity execute concurrently. The dynamic

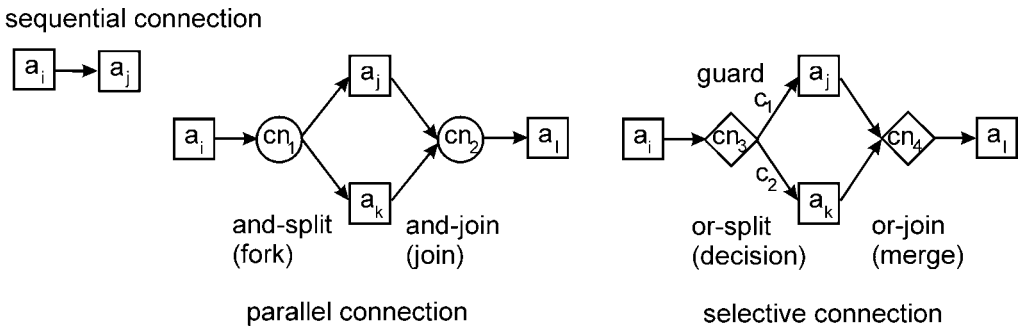


Fig. 3 Some examples of basic control patterns

multiplicity of a state, is the maximum permitted number of invocations of its action or subactivity.

Producer-consumer patterns. These patterns are variants of the producer-consumer pattern found in distributed systems design. They correspond to situations where several instances of an activity A (the producer) are executed sequentially, and the termination of each of these instances triggers the execution of an instance of another activity B (the consumer).

Producer-consumer pattern with termination activity: This pattern involves three activities A, B and C. The process starts with the execution of an instance of A. When this execution completes, an instance of B is enabled. Concurrently, a second instance of A can be started. When this second instance of A completes, a second instance of B is enabled and a third instance of A can be started. When all the instances of A are completed, the system continues executing instances of B until the number of completed executions of B is equal to that of A. Finally, a terminating activity C is executed.

Producer-consumer with bounded queue: pattern is similar to previous one, except that at any time, the difference between the number of times that activity A has been executed, and the number of times that activity B has been executed, is bounded by an integer called the size of the queue.

Whereas a consensus has been reached on defining the set of workflow patterns for *business process* modeling languages, no such patterns exists for workflows applied to *scientific computing* on the Grid. The notion of data flow used in scientific workflows is a natural representation for simple data processing pipelines. Pure data flow is not expressive enough to model either branches and merges in the execution path nor iterative behavior. This is why workflow languages typically focus on the control flow primitives rather than on the data flow aspects. [16] identifies a set of workflow patterns related to parallel and pipelined execution by looking at different kinds of parallelism. These patterns can be classified in two broad categories:

Parallel execution patterns include:

- simple parallelism, where tasks lacking control flow dependencies are executed in parallel
- data parallelism, a form of single instruction multiple data (SIMD) parallelism with three variants: static, dynamic and adaptive.

Pipelined execution patterns include:

- best effort pipelines, where intermediate results are dropped if downstream tasks are not ready to process them
- blocking pipelines, where a form of flow control is used to stop tasks that are located upstream from busy ones
- buffered pipelines, where the workflow accumulates intermediate results between tasks
- superscalar pipelines, where multiple parallel instances of slow tasks are started to process intermediate results
- streaming pipelines where intermediate results are fed into continuously running tasks.

Another classification of workflow patterns related to parallel computing identifies three different kinds of parallelism: inter-workflow, intra-workflow, intra-program. Intra-program and inter-workflow parallelism are also mentioned under the terms of fine-grained and coarse-grained parallelism. Inter-workflow parallelism refers to the simultaneous execution of multiple workflow instances (in general) and of multiple instances of the same workflow (in particular).

Intra-workflow parallelism is defined as the concurrent execution of more than one program (or task) within the same workflow. One can further distinguish tasks without dependencies (parallel execution patterns) from tasks depending on the previous results of one another (pipelined execution patterns). *Intra-program parallelism* implies the distributed execution of individual tasks of the workflow.

2.3 Grid Workflow Management Systems

A workflow management system (WfMS) 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.

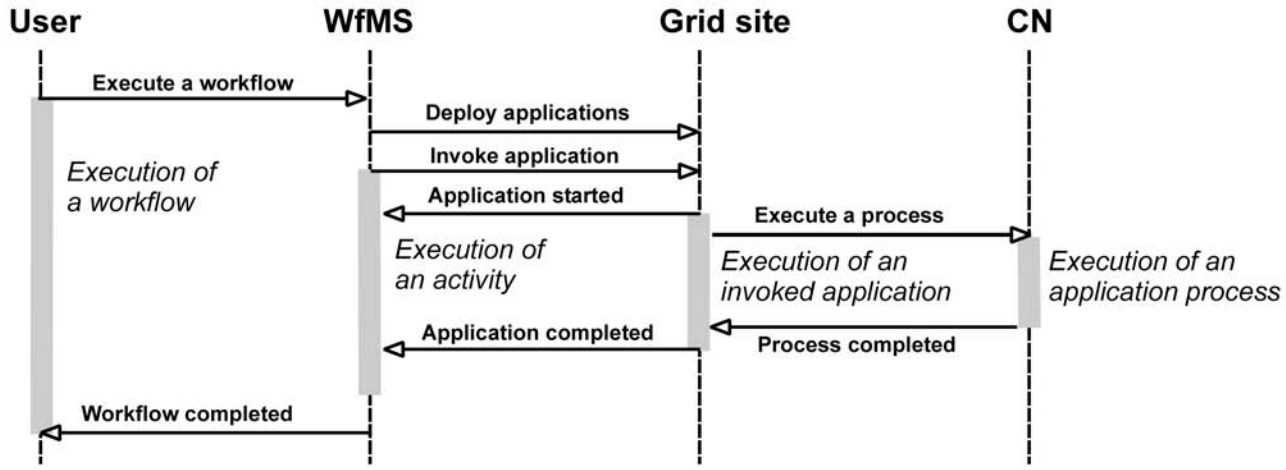


Fig. 4 Grid workflow execution

WfMS consists of two main components: a workflow modelling component and a workflow enactment component (workflow engine). The first offers a build-time environment where workflow specifications can be defined, analysed and managed and it also supports a persistent storage for workflow specifications. The second provides a run-time environment for the creation, execution and management of workflows.

Some Grid workflow management systems: Taverna, GridAnt, GridFlow, Gridbus, Askalon, Karajan, Kepler, and GridNexus.

P-GRADE is a graphical Grid application development environment that graphically supports the design, execution, monitoring, and performance visualisation of workflow grid applications. The workflow describes both the control-flow and the data-flow of the application. A job can be started when all the necessary input files are available and transferred by GridFTP to the site where the job is allocated for execution.

Semantic Grid is becoming a key enabler for next generation Grid and the need for supporting process description and enactment, by means of composition of multiple resources, emerged as one of the fundamental requirements. The aim is to provide architecture with dynamic behaviour and interoperability. The Knowledge-based Workflow System for Grid Applications (K-Wf Grid) [12] addresses the need for a better infrastructure for the Grid environment. The Grid as a vast space of partially cooperating, partially competing Grid services will be a very dynamic and complex environment. The innovation of the K-Wf Grid workflow management technology includes a novel approach of supporting knowledge-based workflow orchestration by means of an expert system.

The compute resources of a grid resource-service provider may be distributed over a wide geographical area. A scheduler should be able to handle a diverse set of jobs, with arbitrary interdependences among processes [2]. Schedule of Grid workflow execution on WfMS, grid site, and on a computational node (CN) is given in Fig.4.

3 Verification

Formal verification using mathematical methods examines the state space of the given design and verifies whether it satisfies the required properties. Specification and verification of concurrent systems are problematic because number of states can increase exponentially with number of parallel parts.

The formal approach uses the modeling language to system description, the specification language to description of the required correct system behavior, and it provides an analysis technique. The model has to describe not only the designed system but also the environment in which it will work. [18] The system can be modeled at different abstraction levels.

Computer-aided verification is a general approach with applications to hardware verification, software engineering, multi-agent control systems, etc. It is appropriate for control-intensive applications with interesting interaction among components.

Most approaches to verification and synthesis of discrete systems are based on dividing the set L of all system behaviors to L_{φ} and $L_{\neg\varphi}$ (i.e., the subset satisfying some property φ, and the subset not satisfying it). The verification verifies whether the subset $L_{\neg\varphi}$ is empty.

To ensure the correctness of grid workflow specification and execution, grid workflow verification and validation must be conducted so that we can identify

any violations and consequently take proper action to remove them in time.

Both grid workflow verification and validation are important. Validation failure constitutes a breach of contract between the complex process developer and the client. Verification failure results in the grid workflow specification and execution containing faults or flaws.

Grid workflow validation is mainly concerned with the consistency between complex processes and the grid workflow specifications. When we model or redesign a complex e-science or e-business process as a grid workflow specification based on models and constructs provided by the selected grid workflow system, we must ensure that all main complex process requirements are modelled or redesigned in the grid workflow specification. Otherwise, the grid workflow specification is incomplete and is incorrect from the perspective of user requirements and needs. For example, some grid workflow systems or architectures cannot support temporal constraint modelling. For those processes such as climate modelling processes for weather forecasting where timing is very important, the corresponding grid workflow specifications are incomplete or incorrect, as temporal information will be ignored.

Grid workflow verification is mainly concerned with the specific correctness of the grid workflow specification and execution such as no deadlock, no livelock, no temporal violation or no resource conflict. It aims at no faults in the grid workflow specification and execution under the condition where complex process requirements have been correctly supported in the grid workflow specification by the selected grid workflow system.

3.1 User & System Requirements

Quality of Service (QoS) issues have not been addressed very well in most Grid workflow management systems [24] due to their focus on the use of system centric policies in resource allocation. When workflow management systems are used in commercial or production environments, supporting QoS at both specification and execution level becomes increasingly critical.

At the specification level, workflow languages need to allow users to express their QoS requirements. At the execution level, the workflow scheduling must be able to map the workflow onto Grid resources to meet users' QoS requirements. Workflow QoS constraints are for example: time, cost, fidelity, reliability, security etc. Some basic verification problems are defined in [11]:

The initiation problem for a workflow object x in a workflow structure W is to determine whether there is a sequence of events leading to the execution of x.

The *termination problem* is to determine whether a workflow structure can reach a terminal state.

A workflow structure is safe if and only if a terminal state can be reached from every reachable state.

A *workflow object w is n-bounded* if and only if in every reachable state w does not occur more than n times.

Verification is usually based on an extension of a kind of formal method, e.g., directed graph, Statecharts, Petri net, process algebra, or temporal logic.

Composition of two workflow modules may cause deadlocks. *Usability* means that an environment exists such that the composition cannot deadlock. Formally, a module is usable iff there exists an environment, such that each execution of the module in the environment terminates correctly, i.e.: each started process finishes, no tokens (documents, messages) are left over, and each transition is relevant, i.e. may be fired.

Guaranteed behaviour and outcome of mission-critical workflows is crucial for workflows in banking, medical applications, electronic commerce, etc. Formalization of required properties can be done using temporal logic e.g. CTL – Computation tree logic, ATL – Alternate temporal logic, etc.

3.2 Grid Workflows Verification Aspects

A workflow specification is a formal description of processes in the real world. Its correctness is critical to the workflow execution.

Structure verification. Structure verification consists of syntactic structure verification and semantic structure verification [7]. In a grid workflow specification, the main syntactic structure inconsistencies include deadlock, livelock, lack of synchronisation, misuse of modelling objects and constructs, active end, dead activity etc.

Current research is more involved in the syntactic structure verification, but semantic structure verification is also very important. The result of structural incorrectness is that either a possible execution exists that does not reach the end node or there are still "uncompleted things to do" when the end node is reached.

Temporal verification. To control the temporal correctness of the workflow execution, explicit temporal constraints are set at the build-time stage and verified at the build-time, run-time instantiation and run-time execution stages. According to [9] the mutual dependency between these constraints is affecting the effectiveness and efficiency of the temporal verification.

Temporal verification aims to check the consistency of fixed-time constraints. A fixed-time constraint at an activity is an absolute time value by which the activity must be completed. Time is expressed in some basic time units - minutes, hours, or days - the granularity is selected according to specific workflow applications. At build-time and run-time instantiation stages we need not

consider where we should conduct temporal verification as each fixed-time constraint needs only be verified once statically. At run-time execution, activity completion durations vary and we may need to verify each fixed-time constraint many times at different activities. The activities at which the verification is done are called *checkpoints*. After the fixed-time constraints are set, the temporal verification is conducted at the above three stages to check whether they are consistent. At the run-time execution stage, some checkpoints are selected for conducting the temporal verification because it is inefficient to do so at all activity points. According to [5] we can specify for each activity its maximum duration, mean duration, minimum duration, run-time start time, run-time end time and run-time completion duration.

Conventionally, a temporal constraint is either consistent or inconsistent. In the grid workflow systems, due to the high uncertainty of the activity completion times, a temporal constraint may have multiple consistencies.

Performance verification. A grid workflow specification or execution is normally attached with some user defined performance parameters [7] such as average activity completion time, average capacity utilisation rate, average activity queuing time, average activity synchronisation delay, average activity set-up delay or average resource allocation. We need to verify whether these parameters can be met by the selected grid workflow system and current system run-time status.

Resource verification. In a grid workflow or between different ones, different activities may compete for the same resource [7] such as a machine or a human being. For example, two activities may compete for the same machine during the same time interval. It may or may not result in actual resource conflict, as they may still be able to get the resource at different times within the same time interval. Resources are classified as two types: shared resources or private resources.

The *shared resources* can be classified as two types: One type of resources can be accessed simultaneously by activities no matter what operations are executed on them. Another type of resources is not allowed to do so. The latter type of resources can be divided into the following two subclasses:

- Resources can be accessed simultaneously by activities depending on their access modes. The actions can be allowed if these modes are compatible.
- The resources are assumed to be occupied exclusively by an activity during its execution and cannot be accessed by another one until its completion. In this case, there is a *resource constraint* between these activities.

Authorisation verification. A grid workflow system often needs to deploy heterogeneous and distributed hardware and software systems to execute a given grid workflow. This gives rise to decentralised security policies and mechanisms that need to be managed. Some authorisation constraints are set for different participants and roles, which show which kinds of resources can be accessed by them and how to access the authorised resources.

Cost verification. We must verify the cost aspect of the grid workflow specification and execution to check whether the grid workflow specification and execution can meet the corresponding budget requirements.

4 EGEE Project

The general principle of grid computing consists in the availability of a network that connects geographically spread computing and storage resources while giving many user groups access to this network. Each user can gain access to the totality of the resources (computing capacity, memory, software, storage, etc.) that have been added to the network by other members of the network. Grid computing means in fact a globalisation and virtualisation of computer infrastructures.

Grid Computing is not yet a standard product on the ICT market but is gaining in popularity. Many projects in the research world are showing the power of computing/data grid infrastructures. Worldwide production grid infrastructures (like EGEE, Enabling Grids for EsciencE http://www.eu-egee.org/) do exist and are looking for cooperation. Computer manufacturers and software companies have started cooperation with those large projects.

The EGEE project [10] brings together experts from over 30 countries with the common aim of building on recent advances in grid technology and developing a service grid infrastructure, which is available to scientists 24 hours a day. The project aims to provide researchers in academia and industry with access to major computing resources, independent of their geographical location. The project primarily concentrates on three core areas:

1. The first area is to build a consistent, robust and secure grid network that will attract additional computing resources.
2. The second area is to continuously improve and maintain the middleware in order to deliver a reliable service to users.
3. The third area is to attract new users from industry as well as science and ensure they receive the high standard of training and support they need.

The *Regional Operations Centres* (ROC) are the heart of the operational support for the grid infrastructure. They have a key role as sources of expert

Fig. 5 EGEE SK web portal

advice and technical support in the process of building and operating the infrastructure.

The Slovak EGEE information portal (see Fig. 5) created in the frame of the EGEE project in Slovakia provides grid related (especially EGEE grid related) information to users from all communities (science, education, industry, and developers).

P-GRADE portal is available as service for the different grid systems and also in Central European Virtual Organization (VOCE) of the EGEE grid infrastructure.

GENIUS is grid portal system giving - alongside with standard command line environment - the most simple and used usage mode of EGEE grid infrastructure services by gLite grid software.

5 Grid computing in Slovakia

Many grid infrastructures were built with different purpose, amount of resources and security levels. Early grid infrastructures were often called testbeds as the grid technologies were under development and grids were used mainly for testing. As the number of users using grids for their work and number of sites involved had increased, testbeds had to transform to production grids.

National grid infrastructures in central Europe aim to join national and regional grid infrastructures and create robust and secure grid available to scientist. 180 participating sites from 41 countries with more than 18.000 CPUs are organized to 12 federations. Slovakia with Poland, Czech Republic, Austria, Hungary, Slovenia and Croatia form the Central European federation. Most of the countries in our federation have national grid initiatives either as projects funded on

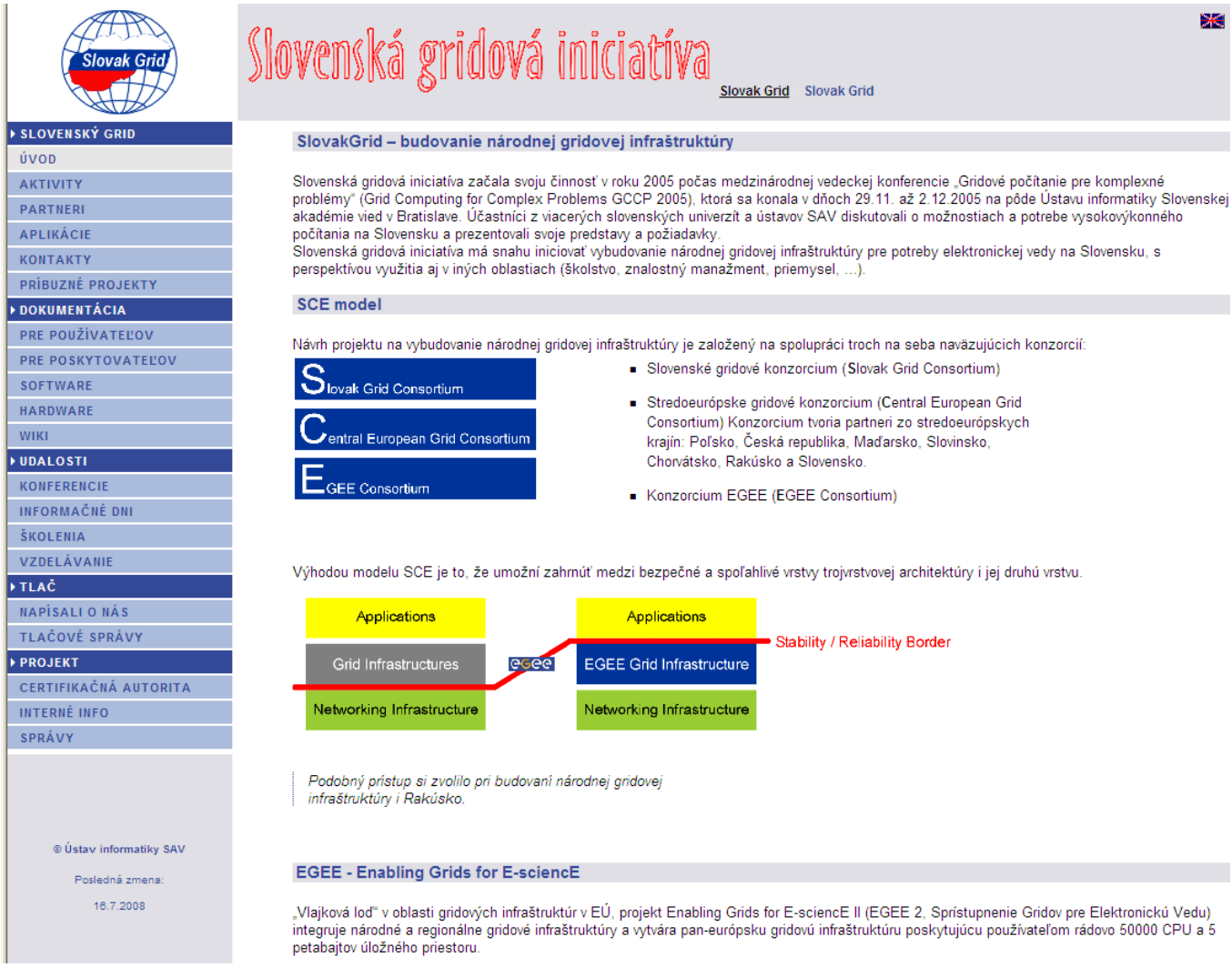


Fig. 6 Slovak grid infrastructure *SlovakGrid* web portal

national level or as centres coordinating national grid activities.

Unlike other countries in central Europe, Slovakia has no national funding program for supporting the development of grid technologies and building national grid infrastructure. Currently there are five grid sites included in EU grid infrastructures.

Institute of Informatics SAS provides 54 CPUs in multiple grid infrastructures. II SAS also provides site and user support including operation of Slovak GridCertification Authority issuing digital grid certificates for users and hosts from Slovakia.

The European Grid Initiative (EGI) Design Study represents an effort to establish a sustainable grid infrastructure in Europe. Driven by the needs and requirements of the research community, it is expected to enable the next leap in research infrastructures, thereby supporting collaborative scientific discoveries in the European Research Area (ERA).

A key component of the EGI vision is the provision of a large-scale, production Grid infrastructure – built on National Grids (NGIs) that interoperate seamlessly at many levels. It is essential that the base functions of currently funded EU grid projects (such as EGEE) will seamlessly transit to EGI. Production hardware will be owned and operated by the NGIs and not by EGI. Regional Operations Centres (ROCs) will be operated by the NGIs – clusters of NGIs may establish a common ROC. The resource allocation will be left to the NGIs.

The main foundations of EGI are the National Grid Initiatives (NGI), which operate the grid infrastructures in each country. EGI will link existing NGIs and will actively support the setup and initiation of new NGIs.

Screenshot of the Slovak grid infrastructure *SlovakGrid* web portal http://www.slovakgrid.sk is given in Fig. 6.

6 Conclusion

EGEE grid project aims to provide researchers in academia and industry with access to major computing resources, independent of their geographical location. The European Grid Initiative (EGI) will link existing National Grid Initiatives (NGI), which operate the grid infrastructures in each country. A grid workflow can be represented by a grid workflow graph, where nodes correspond to activities and edges correspond to dependencies between activities, called flows. To ensure the correctness of grid workflow specification and execution, grid workflow verification and validation must be conducted so that we can identify any violations and consequently take proper action to remove them in time.

Acknowledgements

This work is supported by projects EGEE-III FP7-222667, APVV Project LPP-0231-06, APVV project RPEU-0024-06, VEGA 2/7101/27, and VEGA 2/6103/6.

References:

[1] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, A.P. Barros. “Workflow Patterns”, Journal Distributed and Parallel Databases, Publisher Springer Netherlands, Issue Vol. 14, No 1 / July 2003, pp. 5-51.

[2] A. Aggarwal, M. Aggarwal. “An Adaptive Genetic Algorithm based Scheduler for Grid Applications”, WSEAS Transactions on Information Science and Applications, Issue 12, Vol. 2, December 2005, ISSN: 1790-0832, pp. 2123-2130.

[3] K. Amin, G. von Laszewski, M. Hategan, N. J. Zaluzec, S. Hampton, A. Rossi. “GridAnt: A Client-Controllable Grid Workflow System”, Proceedings of the 37th Hawaii International Conference on System Sciences – 2004.

[4] BPEL4WS http://www-128.ibm.com/developerworks/library/

[5] J. Chen, Y. Yang. “Selecting Necessary and Sufficient Checkpoints for Dynamic Verification of Fixed-time Constraints in Grid Workflow Systems”, Business Process Management, LNCS, Vol. 4102, pp. 445-450, Springer-Verlag, 2006.

[6] J. Chen, Y. Yang. “Multiple states based temporal consistency for dynamic verification of fixed-time constraints in Grid workflow systems”, Concurrency and Computation: Practice and Experience, 2006, www.interscience.wiley.com

[7] J. Chen, Y. Yang. “Key Research Issues in Grid Workflow Verification and Validation”, 4th Australian Workshop on Grid Computing and e-Research, Tasmania, Australian. CRPIT, Vol. 54. Rajkumar Buyya and Tianchi Ma, Ed., pp.97-104, 2006.

[8] D. Dahlem, D. McKitterick, L. Nickel, J. Dowling, B. Biskupski, R. Meier. “Binding- and Port-Agnostic Service Composition using a P2P SOA”, Int. Workshop on Dynamic Web Processes DWP 2005, ICSOC 2005, Amsterdam, Netherlands, 2005, pp. 61-72.

[9] M. Dumas, A. H. M. ter Hofstede. “UML Activity Diagrams as a Workflow Specification Language”, In: M. Gogolla and C. Kobryn, editors, Proceedings of the UML'2001.

[10] EGEE http://www.eu-egee.org/

[11] A.H.M. ter Hofstede, M. E. Orlowska, J. Rajapakse. “Verification problems in conceptual workflow specifications”, In: B. Thalheim, editor, Proc. of the 15th Int. Conf. on Concep-tual Modeling ER'96, Vol. 1157 of LNCS, pp. 73-88, Cottbus, Germany, October 1996. Springer-Verlag.

[12] K-Wf Grid http://www.kwfgrid.net/

[13] F. Lelli, G. Maron, S. Orlando, S. Pinter. “Bringing instruments into a Grid: an Empiric Approach”, WSEAS Transactions on Computers, Issue 1, Vol. 6, January 2007, ISSN: 1109-2750, pp. 153-159.

[14] H. Li, Y. Yang, T.Y. Chen. “Resource constraints analysis of workflow specifications”, The Journal of Systems and Software 73 (2004) pp. 271–285.

[15] S. Pastore “Investigating frameworks to address issues involved in sharing service-based software resources within grid environments”, WSEAS Transactions on Computers, Issue 1, Vol. 6, January 2007, ISSN: 1109-2750, pp. 72-79.

[16] C. Pautasso, G. Alonso. “Parallel Computing Patterns for Grid Workflows”, In: Proc. of the HPDC2006 Workshop on Workflows in Support of Large-Scale Science WORKS06, Paris, France, June 2006, http://www.iks.ethz.ch/publications/jop_grid_workflow_patterns

[17] S. Pllana, T. Fahringer, J. Testori, S. Benkner, I. Brandic. “Towards an UML Based Graphical Representation of Grid Workflow Applications”, The 2nd European Across Grids Conference, Nicosia, Cyprus, 2004. LNCS, Springer Verlag.

[18] Sebestyénová J. (2003). Hierarchical Verification of Reactive Systems with Timing Constraints, In: “WSEAS Transactions on Computers”, Ed. Mastorakis N., Issue 4, Vol.2, October 2003, ISSN: 1109-2750, pp. 1174-1179.

[19] Staffware http://www.staffware.ch/

[20] A. Terracina, S. Beco, T. Kirkham, J. Gallop, I. Johnson, D. Mac Randal, Brian Ritchie. “Orchestration and Workflow in a mobile Grid environment”, 5th Int. Conf. on Grid and Cooperative Computing Workshops, 2006, pp. 251-258.

[21] Websphere MQ Workflow http://www-306.ibm.com/software/integration/wmqwf/

[22] WfMC http://www.wfmc.org/standards/model.htm

[23] J. Yan, Y. Yang, R. Kowalczyk, X. T. Nguyen. “A Service Workflow Management Framework Based on Peer-to-Peer and Agent Technologies”, www.it.swin.edu.au/personal/yyang/

[24] J. Yu, R. Buyya. “A Taxonomy of Workflow Management Systems for Grid Computing”, www.gridbus.org/reports/