# Integrated Support for Medical Image Analysis Methods: From Development to Clinical Application

Sílvia D. Olabarriaga, Jeroen G. Snel, Charl P. Botha, *Member, IEEE*, and Robert G. Belleman

***Abstract*—Computer-aided image analysis is becoming increasingly important to efficiently and safely handle large amounts of high-resolution images generated by advanced medical imaging devices. The development of medical image analysis (MIA) software with the required properties for clinical application, however, is difficult and labor-intensive. Such development should be supported by systems providing scalable computational capacity and storage space, as well as information management facilities. This paper describes the properties of distributed systems to support and facilitate the development, evaluation, and clinical application of MIA methods. First, the main characteristics of existing systems are presented. Then, the phases in a method's lifecycle are analyzed (development, parameter optimization, evaluation, clinical routine), identifying the types of users, tasks, and related computational issues. A scenario is described where all tasks are performed with the aid of computational tools integrated into an ideal supporting environment. The requirements for this environment are described, proposing a grid-oriented paradigm that emphasizes virtual collaboration among users, pieces of software, and devices distributed among geographically dispersed healthcare, research, and development enterprises. Finally, the characteristics of the existing systems are analyzed according to these requirements. The proposed requirements offer a useful framework to evaluate, compare, and improve the existing systems that support MIA development.**

***Index Terms*—Distributed computing, grid computing, medical image analysis, problem-solving environment.**

## I. Introduction

MEDICAL images are the basis for a large number of clinical tasks in the daily routine of healthcare. Continued developments in acquisition technology enable capturing the increasing amounts of high-resolution images that reveal different aspects of the human body's structure and function with unprecedented detail. In the clinical routine, such large amounts of data raise not only storage issues but also challenges for image analysis. In summary, how can/should such large amounts of data be interpreted efficiently and safely? Adopting computational aid for medical image analysis (MIA) is no longer an option, but a necessity.

Manuscript received September 30, 2005; revised February 22, 2006. This work was supported by a BSIK Grant from the Dutch Ministry of Education, Culture and Science (OC&W) and is part of the ICT innovation program of the Ministry of Economic Affairs (EZ). This work was carried out in the context of the Virtual Laboratory for e-Science project (www.vl-e.nl).

S. D. Olabarriaga and R. G. Belleman are with the Informatics Institute, University of Amsterdam, 1098 SJ Amsterdam, The Netherlands (e-mail: silvia@science.uva.nl).

C. Botha is with the Faculty of Electrical Engineering, Mathmatics and Computer Science, Delft University of Technology, 2600 GA Delft, The Netherlands.

J. G. Snel is with the Academic Medical Center, University of Amsterdam, 1100 DD Amsterdam, The Netherlands.

Digital Object Identifier 10.1109/TITB.2006.874929

MIA is an active area of research that aims at the development of (highly autonomous) computational methods for image enhancement, segmentation, registration, measurement, and interactive visualization. Methods are designed to (automatically) enhance, detect, select, and display features of interest in the image, with goals such as shortening or eliminating examination times, reducing subjectivity, and facilitating measurement. Methods can originate from new algorithms and from innovative combinations of existing ones. In practice, MIA methods are often implemented as a composition of processing steps that progressively analyze the image to generate the required result. The images can belong to sets of different scans and the results can be of varied data types, such as images, measured values, or geometrical descriptions. In this paper, we refer to single or composed methods that perform any type of MIA operation simply as *MIA methods*.

Although many MIA methods are proposed in the literature, only a limited number prove themselves in producing results with sufficient reliability to be acceptable for application in clinical tasks. This is due to three main reasons. First, it is difficult to mimic, by software, the performance of trained human operators in the task of image interpretation. Algorithm design, as well as finding the best combination of image analysis steps, are often challenging tasks. Moreover, even when the MIA method is well established, it might need adaptation when the imaging protocol changes. In many cases, MIA methods simply do not meet the required quality standards when confronted with broader settings and are left unused. Second, the validation of a new method, before it can be used in practice, demands a huge effort. This is partly due to the high requirements on reliability inherent to clinical tasks, and also to the large biological variability and the lack of ground truth for the objective performance evaluation. The quality of the produced results is usually estimated in a statistical manner, based on a large number of images and often involving human intervention. Such evaluation studies consume large computational resources for data storage and processing, which turns out to be a major burden in the validation of software for MIA. And finally, integrating a new method into the clinical environment requires its implementation in workstations that are integrated into the existing image analysis workflow. This integration might be impossible to achieve in turnkey systems without vendor participation, or it could create an uncomfortable situation for the operator, who needs to move physically among dispersed devices.

Some of the problems discussed above can be reduced when an adequate (computational) infrastructure is adopted to efficiently handle the logistics of MIA development and deployment. Problem-solving environments (PSEs) [1], [2], which are

typically adopted to support large and complex (scientific) computations, could provide such infrastructure. These systems, or environments, consist of interactive platforms for designing, executing, and monitoring complex sequences of operations and experiments. Offered facilities typically include a graphical editor, assistance for composing sequences of operations with off-the-shelf software components, transparent access to distributed high-performance computing (HPC) resources, aids for data storage and management, recording of experiments, etc. Although different architectures and technologies can be adopted in the implementation of these systems, they share the goal of facilitating the management of "workflow" in scientific environments, which is why PSEs can also be called workflow management systems.

The goal of this paper is to identify the properties of computational tools (or PSEs) to support and facilitate the development, evaluation, and clinical deployment of MIA methods. The contribution of this paper is twofold: First, we present a detailed analysis of the lifecycle of MIA solutions as they are successfully integrated into clinical environments. The lifecycle analysis serves as a framework for projects that aim at developing MIA solutions intended for use in clinical tasks. Using this framework, researchers can understand and guide their research efforts onto clinically usable MIA methods. Secondly, we document the requirements for an environment that can support and facilitate this development process. We also summarize the characteristics of existing systems according to the requirements. The requirements can be used as a framework to evaluate, compare, and improve the existing software systems to support MIA development.

The paper is organized as follows. In Section II, we present a summary of related work on PSEs and other tools to support complex scientific computation. Next, the phases of MIA software lifecycle and the associated computational issues are identified (Section III). The requirements for integrated computational support at all phases are delineated in Section IV, using an exemplar (ideal) scenario as illustration. The characteristics of selected systems are analyzed in Section V, with the discussion of a real MIA application. Finally, Section VI closes the paper with a discussion and comments about future work.

## II. Related Work

A great number of PSEs are available that are relevant to our work, many of them being inserted in a grid computing paradigm. In this section, we briefly summarize a selection of these systems that are directly relevant to aspects of the MIA method lifecycle and also to the requirements documented in Section IV. In Section V, many of these systems will be mentioned again in the context of how they relate to the requirements that we set out.

Application visualisation system (AVS) [3] is one of the first component-based application builders based on a visual programming (VP) paradigm in which blocks, representing algorithms, can be interactively connected with lines, representing connections through which data flow. The current generation of AVS, called AVS/Express, is used for medical imaging and visualisation.

Khoros [4] is a component-based system for general image processing problems. This is also an early example of a VP-based system.

SciRun [2] is a modern software package that focuses on providing functionality for computational steering. This PSE also adopts a VP approach. The design goal of this system was to provide scientists with a tool for creating new simulations, developing new algorithms, and coupling the existing algorithms with visualization tools. Successful biomedical applications have been implemented in SciRun, e.g., BioTensor, for diffusion tensor imaging, and BioImage, for processing and visualizing medical image volumes.

The DElft visualization and image processing development environment (DeVIDE) [5], is a modular framework for the fast prototyping and deployment of medical visualization and image processing algorithms. Its primary interface adopts the VP paradigm. DeVIDE distinguishes itself from other similar packages in that it facilitates what its authors have dubbed "pervasive interaction," referring to the fact that a user can interact at all levels and with all aspects of the system. For example, algorithmic source code can be modified at run-time and the effects are immediately shown.

LONI Pipeline [6] aims at providing a tool for image analysis and visualization for neuroimaging applications, also adopting a VP approach. It is able to automatically parallelize data-independent components. The latest planned release includes integration with grid resources.

Kepler [7], VLAM-G [8], Triana [9], Taverna [10], and GridNexus [11] are other workflow management systems that aim at scientific applications in a grid-enabled environment. However, none of these focus on biomedical applications.

IXI [12], [13] is an architecture that makes medical image processing algorithms available as grid services. It is possible to combine these services into more complex image processing workflows that execute over distributed computing resources. Concretely, IXI consists of a set of grid services and a workflow definition language.

MIAKT [14] represents an interesting approach in allowing different clinicians to collaborate, via the MIAKT software, on a diagnosis and treatment plan for patients with breast disease. The system is integrated with the existing hospital environment and gives access to patient records and previous examinations, including image data. It uses grid and web services to make available relevant medical image processing functionality.

Distributed medical data manager ($DM^2$) [15], [16] addresses some of the challenges of working with real medical data (e.g., DICOM) on the grid. By acting as an interface between the normal grid mechanisms for data handling and existing clinical DICOM servers, privacy-sensitive patient data can be anonymized and encrypted before being sent to the grid. Only trusted grid sites are then able to further process the data. Image metadata is also handled.

Liu *et al.* [17] have created a database-oriented workflow management system. Their system focuses on the processing of large collections of datasets that are used in cross-sectional

studies and multicenter trials. It integrates with third-party applications for visualization.

Storage resource broker (SRB) [18] is a generic framework for storing and making accessible datasets and their metadata on the grid.

Chimera [19] is a grid-based data processing framework that implements a database to store provenance data. Provenance data refers to metadata that describes all operations that have been performed to yield a certain result. In essence, the complete history of a given dataset can be stored and queried. Pegasus [20] is able to generate workflows based on data dependencies that can be extracted, e.g., from stored provenance data.

Grid architecture for medical applications (GAMA) [21], is a framework that allows hospitals to make use of grid-based infrastructure by supplying a grid access point component that brokers grid services to hospital workstations. A primary focus of the GAMA framework is being minimally invasive at the user side. In other words, to the user, the application runs identically, independently of whether the processing is being performed on the grid or not.

## III. Phases of MIA Methods Lifecycle

During its lifetime, MIA methods evolve through four different phases. The first phase is experimental, in which new methods are designed, implemented, and tested; this is the *development* phase. The second phase is the *parameter optimization* phase, in which the optimal configuration of parameters is determined for a given class of situations. In the third phase, the MIA method is applied to a large number of images in *evaluation* studies that determine their performance at the technical and clinical levels. Finally, successful MIA methods reach a phase in which they are inserted in the *clinical routine* and their results are used to support clinical tasks.

Two main factors distinguish this model of MIA software construction from the classic sequential waterfall model [22]. First, feedback loops to earlier phases of the lifecycle are common, e.g., for further development or parameter optimization when the image acquisition protocol changes due to the employment of a new scanning device. Second, each phase involves different types of activities that are performed by users assuming different roles, in a context where people need to collaborate with the aid of computational tools over long periods of time.

In Section III-A–D the phases are characterized in terms of the aspects that influence the required computational support: type of user, typical tasks, amount of handled data, need for automated versus interactive processing, required processing speed, handling of failure, stability of implementation (software, parameters), and interoperability with other systems. The types of users we define are modeled after those presented by Parsons *et al*. [23]. A summary of the lifecycle phases characteristics is presented in Section III-E.

### A. Development

New methods for specific image analysis tasks are designed, implemented, and tested in the development phase. Here, we assume a component-based paradigm in which image analysis functions are implemented by *components* that can be combined into processing *networks*. This paradigm was introduced by the AVS system, being widely adopted in MIA and other application areas. According to this paradigm, the developed method could be an autonomous component (e.g., a new image segmentation algorithm) or a network of existing components (e.g., combination of statistical analysis and image registration for a functional MRI study).

Two user roles are identified: component developer (CD) and network developer (ND). Whereas the CD is typically a scientific programmer, the ND could be a clinician with training in image processing. Note that the same user can assume the roles of CD and ND, e.g., when a network is used for testing of a new component.

The CD and ND perform different activities. The CD builds components, performing typical programming tasks such as design, implementation, debugging, and testing. The component must be shipped with an the appropriate interface that enables its adoption into networks at a later stage. The ND selects the appropriate components, combines them into networks using control flow constructions, debugs, and tests the networks.

The development of components and networks share the following characteristics. The method is typically tested for a limited number of images, and the results are visually inspected. Interaction is essential to modify the implementation and parameters, as well as to inspect results in real time. The method is executed in a stand-alone, informal, and tolerant environment that intrinsically includes faults. The implementation is stable only at the end of the development phase, whereas the optimal parameter settings are to be found at a later stage.

### B. Parameter Optimization

Parameter optimization consists of finding the best parameter settings for a method, given an application domain (e.g., the intensity threshold for vessel segmentation from CT angiography of the head). One user role is identified in this phase: parameter optimizer (PO). The PO has a technical background but does not necessarily know how the method works; it could be the CD, ND, or a clinical co-worker.

In this phase, the method is executed repetitively by the PO with varying parameter settings, and the quality of obtained results is analyzed qualitatively by visual inspection, or quantitatively, e.g., with quality metrics. If necessary, initially, the parameter space is investigated to limit the search range, e.g., with parameter sweeping techniques (e.g., [24]). Subsequently, interactive parameter tuning is performed for a limited range of parameter values. A controlled environment is adopted to facilitate the assessment of result quality, often using artificial test data (e.g., added noise, contrast, and resolution manipulation), phantoms, and reference results constructed manually from real data.

The method is typically applied to a reduced number of images in a stand-alone setting. Interaction and short response time are essential for real-time steering in the parameter-tuning phase. For non-interactive optimization, support for autonomous and repetitive method execution is required. The execution

context is tolerant to errors, although the implementation is considered stable. Operational faults are not expected at this phase, and functional faults should motivate further development by the CD/ND.

### C. Evaluation

In this phase, the method is applied to a large number of images for (statistical) assessment of its clinical value. At the technical level, evaluation studies typically measure the method's accuracy and reproducibility, robustness of results obtained under varying conditions, and response time. At the clinical level other parameters such as the amount of human intervention, costs, hospitalization time, and patient comfort are usually assessed. The parameters that provide information about how a new method compares with the current practice differ according to the clinical task. In all cases, during evaluation, the method is inserted into a broader processing scope to test its combination with existing techniques and infrastructure, simulating a realistic environment. The activities involved are similar for a technical and a clinical evaluation, so a single user role is identified: evaluation user (EU). For technical evaluation, the EU has technical background in MIA, whereas he/she is a clinician in clinical trials and epidemiological studies.

The activities consist of applying the method repetitively to large amounts of artificial or real data and assessing the properties of the obtained results. Whenever available, quantitative assessment of quality is preferred by comparing results with a reference generated manually or with a procedure adopted routinely. Double blind statistical analysis is often employed in clinical studies. In such studies, a significant amount of management effort is spent for the evaluation, collection, and analysis of results, in particular when multiple centres are involved. And, to obtain official certification for a new method (e.g., FDA approval), all steps of the evaluation process must be carefully documented and retrieved over large periods of time.

Due to the large amount of processed data, interactive analysis is typically limited to verifying selected results (e.g., cases of malfunction). When the final goal of the MIA method is visualization (e.g., white matter fiber tracking from DTI MR scans for pre-operative planning of neurosurgery), interaction and real-time processing are essential in this phase. To facilitate access to real data (patients, volunteers) in clinical trials or epidemiological studies, the MIA method should be linked to the image database (PACS[1]) in a secure manner, preserving the hospital's policy for patient data privacy. In this phase, the implementation and parameter configuration are stable and the execution context is less tolerant to faults. Functional faults should motivate further parameter optimization or development by the PO, CD, or ND.

### D. Clinical Routine

After being approved by thorough evaluation, the method is eligible to process patient data and generate results to support daily clinical practice. The MIA method is inserted as a step in the clinical routine, such that it seamlessly participates in the generation of results used to support the clinical user (CU) in tasks such as diagnosis, prognosis, treatment planning, and follow-up. Remote access to distributed resources such as scanning devices and (image) databases are commonplace when telehealth services are available.

In this phase, the patient data are processed by the appropriate MIA method as a part of the routine workflow. The method and associated parameters correspond to the "best practices" determined during the extensive evaluation studies performed earlier in the lifecycle. The method is now fully integrated into the IT infrastructure of the radiology department (PACS), to the hospital information system (HIS), and possibly to remote information systems. Requirements on data privacy are very high in this phase, in particular when resources are distributed among different health centres. For efficiency in healthcare, the MIA methods applied in the clinic are typically automated or require minimal user intervention. When the final goal is visualization, interaction and real-time response are essential in the routine application of a MIA method. Otherwise, the type of clinical task determines the demands on response time, being very critical for first aid support. Finally, robustness of the method is a strong requirement in this phase, since the system must operate safely with guaranteed QoS[2] (for all patients, 7 days/week, 24h/day). Changes in the environment (e.g., different scanner, imaging protocol, system architecture, operating system) trigger tests to verify whether the method remains valid. Further parameter optimization (PO) or development (CD/ND) may be necessary in the case of malfunction.

### E. Characteristics of MIA Lifecycle Phases

The characteristics presented earlier directly influence the kind of support required by the users assuming different roles along the method's lifecycle (see summary in Table I).

*1) Typical User:* Along the lifecycle, the user background shifts from purely technical to purely clinical. At one end of the spectrum of users, the CD and ND have a technical background (image analysis, programming) and, at the other end, he/she is a clinician. In the center, the PO and EU have mixed competences (e.g., a radiologist with training in digital image analysis, a programmer with training in anatomy). Such a large spectrum of users represents a challenge to the design of systems and user interfaces. Moreover, it implies the need of collaboration among people with different background and interests. In particular when faults occur while the method is executed in a clinical setting, the CU (or EU) typically does not have the necessary technical background to solve the problem. In these cases, the intervention of other participants (CD, ND) should be requested. Support for such collaboration and communication is required for efficient and secure system operation.

*2) Tasks:* The focus of the task shifts from method to result along the lifecycle. In the development phase, the focus is on the internal aspects of the method's implementation, be it a component (programming tasks) or a network (composing tasks). In

[1]Picture Archiving and Communication System.

[2]Quality of Service.

TABLE I
CHARACTERISTICS OF PHASES IN THE LIFECYCLE OF MIA METHODS

| | Development | Parameter Optimization | Evaluation | Routine |
|---|---|---|---|---|
| User role | CD, ND | PO | EU | CU |
| User Background | technical | mixed | mixed | clinical |
| Tasks | design, implement, test | run, evaluate | run, evaluate, statistics | run, adopt |
| Amount of Data | small | medium | large | very large |
| Processing Type* | interactive | interactive, automated | automated, repetitive | automated |
| Processing Speed* | desired | needed | unnecessary | depends |
| Faults | acceptable | tolerable | undesired | unacceptabl |
| Stability | unstable | changing parameters | stable | fixed |
| Interoperability | not needed | not needed | desired | required |

Notes : CD: component developer; ND: network developer; PO: parameters optimiser; EU: evaluation user CU: clinical user

*Applicable to noninteractive methods.

the parameter optimization phase, the focus is on the method's parameters and the generated results. Tasks here involve running the method with chosen parameters and evaluating the results. In the evaluation phase, the focus lies on the results, which are used to assess the method's properties. The parameter settings are seen as a part of the method, which is executed as a black box. Tasks include running the method, evaluating the quality of results, collecting outputs (results, quality), and running statistical analysis on the outputs. Finally, in the clinical routine, the focus is purely on the generated results. Tasks include running the method with well-defined parameters. The design and implementation of an integrated system to support such a broad range of tasks represents a software engineering challenge.

*3) Amount of Data:* The amount of data processed in the experiments increases along the lifecycle. A small number of data instances are handled during development and parameter tuning, simply because interactive analysis is seldom admissible otherwise. More data are admissible when automated parameter-sweeping tools are adopted, requiring the assessment of quality in a quantitative manner. Evaluation studies involve larger amounts of data, typically up to 100 instances for technical evaluation and several hundreds for clinical trials and epidemiological studies. In the clinical routine, the number of images is unlimited. Note that "data" in this context refer not only to images (volumes, two-dimensional (2-D) slices) but also to other types of results generated by the method (e.g., vector data) and the associated metadata (e.g., the score produced during interpretation by a radiologist). For small to medium amounts of instances, data management can still be done manually using local processing and file systems. For larger datasets, structured support for data and information management is necessary. In particular for multicenter or long-term studies, large and distributed storage and processing resources may be needed. Besides the typical challenges associated with the design and implementation of distributed databases, medical applications introduce an additional challenge with respect to the need of security due to strict regulations about patient data privacy.

*4) Processing Type:* The type of processing shifts from highly interactive to highly automated along the lifecycle. Interaction is needed for user input (image or other input data) and parameter adjustment, being essential during development and parameter tuning. For practical reasons, interaction is not desirable when large amounts of data are processed repetitively. Moreover, interaction is usually avoided in the clinical practice because it introduces subjectivity in the analysis process. When the goal of the MIA method is visualization, however, interactive analysis is an essential part of the image interpretation process. In all cases, pre- or post-processing MIA tasks should be executed as autonomously as possible, such that the user is prompted only when intervention is needed. The challenge here is to design a platform in which the alternation and synchronization between interactive and autonomous execution can be done in a seamless fashion, without modifications in the method's implementation.

*5) Processing Speed:* High processing speed is required for real-time interactive execution in the development and parameter optimization phases, as well as in all phases when the MIA method fulfils an interactive visualization goal. In the daily routine, the CU expects results to be produced rapidly but not necessarily in real time. Higher demands could be put on the response time for critical tasks, e.g., when the generated result supports first aid or it is used in real time for image-guided surgery. Although speed is less relevant when automated processing is adopted, high throughput is required when large numbers of data instances need to be processed, e.g., in large population studies. High-performance computing (HPC) resources can be used to guarantee the desired response time in all cases. The challenge here is to adopt HPC resources transparently, such that the user does not have to be aware of the involved logistics.

*6) Faults:* The acceptability of faults in a MIA method decreases along its lifecycle. Method malfunction can occur when the input data do not match the conditions imposed by the implementation, generating an exception or a wrong result. Whereas during development faults are admitted and even expected as part of the debugging process, they become less acceptable at later phases of the lifecycle. In particular when many data instances are involved, faults are inconvenient because they disrupt the regular flow of processing (e.g., the instances have to be removed from the study). And, in the clinical practice, faults are not admitted at all, since there is a large responsibility associated with the image interpretation task that depends on the generated result. Adequate fault handling is necessary to prevent situations in which errors may go unnoticed or unsolved and a wrong result is delivered to the user. This can occur when the amount of generated data is too large, or when the image analysis network is too complex and the user does not have the

technical capacity for fully understanding it. At any rate, fault conditions should never cause a program to terminate execution abruptly or to generate a wrong result. Instead, an informative notification should be presented to the CU, which depends upon that information, and to the CD/ND, which should investigate how the problem could be avoided in the future. The challenge here goes beyond the development of fault-tolerant software in a general sense. It requires a MIA method that can detect conditions of functional fault and react accordingly, which is a nontrivial issue in image analysis.

*7) Stability of Implementation and Configuration:* The stability of a method's implementation and configuration increases along its lifecycle. The implementation evolves mainly during development, but the parameter settings change also in the parameter optimization phase, when the method is adapted to a given application. During the evaluation studies, when the method is applied to a larger number of images, the parameter configuration is tested, possibly further improved, and fixed at the end. When the method is applied in the daily routine, the parameter values and implementation are fixed, reflecting a stable status that corresponds to "best practices" found during extensive evaluation studies. The challenge here is to keep track of the history of modifications in the method's implementation and in the parameter settings for different applications, as well as the configuration adopted to produce a particular piece of (meta)data.

*8) Interoperability With Other Systems:* The need for interoperability with other (enterprise) systems increases along the MIA method lifetime. During development and parameter optimization, the method is executed in a stand-alone fashion, in an experimental research environment. When real data are adopted during evaluation studies, it is necessary to connect the method into the IT infrastructure of the radiology department, even if only to read the data. Finally, for usage in the daily routine, the method must be fully integrated into the clinical enterprise information systems, for reading scanned data and for archiving the image analysis results. The challenge here is to design and implement a software architecture that facilitates the migration among different execution environments, as well as the integration with heterogeneous multi-vendor enterprise systems.

## IV. Computational Support for MIA

For efficiency in the development of new MIA methods, computational tools are adopted to support the activities in the various lifecycle phases. The tasks to be performed are different for each phase, posing different requirements on the supporting tools. In practice, different tools, when available, are adopted for each phase, and the working environment adopted in one phase has to be manually adjusted to the next phase. For example, the interactive program used by the CD/ND has to be modified for execution from an automated environment (e.g., a scripting language) by the PO. In the simplest case, a command-line interface and compatible error handling must be provided. In another example, it might be necessary to resume the development phase when a problem has been detected during a clinical trial, but the input image and the parameter settings that caused the malfunction have to be recreated manually by the CD/ND. Such practices are not only tiresome but also prone to transcription/porting errors. An integrated supporting environment would be therefore valuable to facilitate navigation along the method's timeline. Moreover, such an environment could also capture knowledge and methodology, enabling reuse in similar situations in the same or in another phase, for the same or for another method. Section IV-A presents a scenario in which an ideal MIA supporting system (MIASS) is adopted during the lifecycle of a MIA method, and the requirements of the ideal system are defined in Section IV-B.

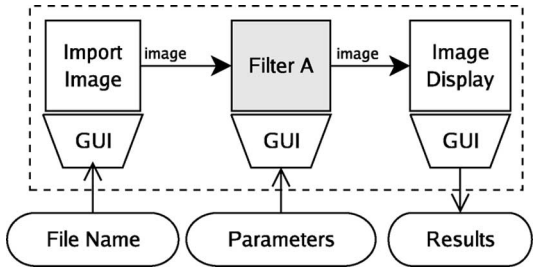


Fig. 1. Network for interactive execution. Rectangles are components, trapeziums are automatically generated GUIs, and rounded shapes represent input/output. The arrows correspond to connections between the components' ports.

### A. Example: Ideal MIASS

The scenario presented later describes the lifecycle of a hypothetical MIA method assuming the support of an ideal MIASS. The method is a filter (Filter A) that enhances an input image, producing another image as output.

In the *development phase*, the CD implements the functionality of FilterA in a given programming language, using a chosen integrated development environment (IDE). The filter is encapsulated into a component to enable its inclusion into processing networks. The MIASS component interface defines *input* and *output ports*, as well as *input parameters* and *execution monitors*. When networks are executed, ports are used to create data channels linking components, and GUIs are generated automatically to enable interactive input of parameter values and visualization of progress indicators in real time. The code for implementing the component interface is generated automatically by an interactive wizard from a high-level description provided by the CD.

For debugging, the new component is plugged into a simple processing network (see Fig. 1), and the user assumes the role of a ND. Access is provided to a large repository of general and specific components, e.g., to import, compare, and display images. A digital assistant facilitates the network construction by suggesting appropriate components and network configurations based on data type associations, templates, and other networks adopted in similar situations.

During network execution, the ND can change parameters interactively and observe progress via monitoring elements (e.g., status bar) and control functions (e.g., stop, pause, run). Only the information explicitly exposed by the encapsulation interface is

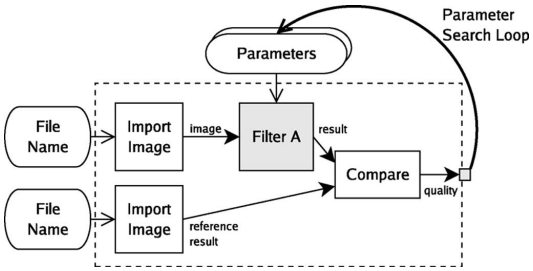


Fig. 2. Network for parameter sweeping executed repetitively by an external control loop.

visible to the ND; to debug the internal implementation of FilterA, the user must resume the role of CD. The MIASS exports the input data (obtained from input ports) and the parameter values (obtained interactively) and activates the IDE, recreating the execution context in the external environment.

The module implementation is kept up-to-date at all times. Source version control, revision management, and additional logging facilities enable tracking of the implementation history.

The *parameter optimization phase* starts after the method's implementation is considered satisfactory. Initially, the parameter space of Filter A is investigated by the PO using parameter-sweeping facilities provided by the MIASS. With these facilities, the method is applied repetitively to the same images for given ranges of parameter values, adopting the network illustrated in Fig. 2. The generated image is compared to a given reference, possibly generated manually, to obtain a quantification of quality that guides the search in the parameter space. Details of the parameter settings are now hidden from the user (no GUI) and replaced by a mechanism that automatically provides input to the method based on the search strategy. The computation is automatically distributed among HPC resources to accelerate the search process.

After the plausible ranges of parameter values have been determined, the network in Fig. 1. is reused by the PO for interactive tuning. The method is applied to test images, and the results are inspected visually in real time as the parameters are changed interactively. The history of parameter-tuning (input data, parameters, resulting images and quality) is logged automatically and can be retrieved for analysis and documentation purposes. The optimal parameter settings are saved for future use.

When an error is detected, the CD or the ND is notified and the situation that caused it is reproduced by the MIASS in the development environment. All improvements on the method's implementation are automatically available to all the dependent networks.

The *evaluation phase* starts once the optimal parameter settings have been determined. Using the MIASS image-sweeping facilities, the method is applied repetitively to a large number of images. The adopted network (see Fig. 3) is the same that is used for parameter sweeping (Fig. 2), however with a different external control mechanism. Here the parameter values are fixed, the network is run for a collection of images and corresponding reference results, and the results and measured quality are gathered for posterior analysis with statistical tools.

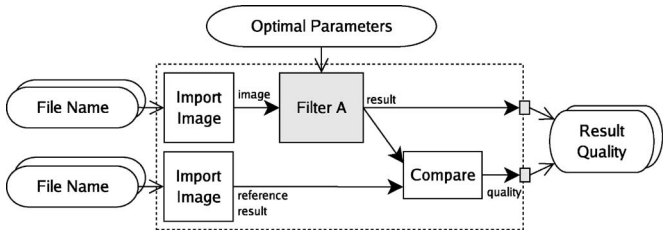


Fig. 3. Network for image sweeping executed repetitively by an external control loop.

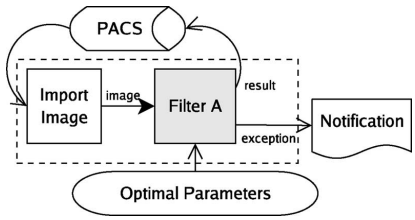


Fig. 4. Filter A inserted into the clinical environment.

The provenance/history of results is also saved for future analysis, including input data and details about the executed method such as version, parameters, etc. To support the large amount of data and processing required for such studies, the ideal MIASS furthermore provides transparent access to HPC resources. The computation is distributed among processing nodes, for e.g., by running the method in parallel for different images. Remote and large-capacity data servers are used to store the images, results, and provenance. Data are transported transparently between the data and processing servers by the MIASS without intervention of the PO.

New failure situations can occur during the evaluation, and the PO/ND/CD should be notified with the proper contextual information. When a failure corresponds to a permanent functional limitation of the method, the CD or ND introduces fault-handling mechanisms to detect the situation and activate the proper action (e.g., user notification). At the end of the evaluation phase, the EU consults the records of history to establish the "best practices" for applying the method in clinical environments. These records also facilitate the generation of documentation to obtain approval for adopting the method in the clinical routine.

Finally, the validated method is introduced in the *clinical routine*, being linked to the IT infrastructure of the radiology department (PACS). Only the core network, without interactive or quality measurement elements, is used in this phase (Fig. 4). The method is deployed in a computation node and registered into a service-oriented system. This service enables the association of methods/networks and parameter settings to metadata such as imaging modality, acquisition protocol, or annotations introduced by the operator of the scanning device.

After a scan is stored in the PACS, the service-oriented mechanism determines the necessity to include FilterA in the radiologist's diagnostic workflow. The image is then transported to the computational node where the method was installed, and the method is executed with the corresponding parameter values.

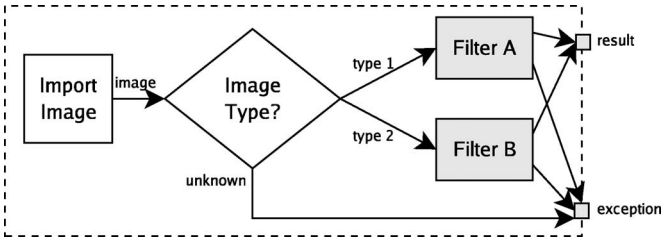


Fig. 5. Network implementing a method where either Filter A or Filter B is executed depending on a condition (diamond). Note: the parameters are omitted for simplicity.

After completion, the resulting image is stored into the PACS, and optionally the CU is notified that the result is available for interpretation via a messenger system (e.g., web-based message board or beeper). Exceptions are clearly identified, and the CU has the choice of activating the interactive processing network (Fig. 1) for further investigation. When technical background is needed, or when a problem is detected, the ND or CD are called in for action.

The clinical consolidation has now been completed, and the lifecycle can be restarted. The historical records stored in the MIASS archives are valuable to facilitate software and methodology reuse. For example, the parameter optimization and evaluation phases must be repeated when the image acquisition protocol has been modified. The strategy adopted earlier can be easily reused by retrieving the processing networks. Another example is the introduction of an alternative to FilterA, say FilterB, which provides improved image enhancement results. The networks adopted for FilterA in all phases can be adapted by replacing one component, and the evaluation data (images, references) can be reused. The results obtained with both methods can be compared easily to determine the best performing one according to the adopted quality metrics. Improvements in the MIA method (e.g., new implementation or parameter settings) can become directly available in the clinical environment by modifying the MIASS service registry.

After some time, there will be several versions of Filter A and Filter B, as well as different parameter settings for their application in particular cases. The MIASS archives are again useful to keep track of the results generated with different implementations and parameter settings. Possibly, Filter A performs better than Filter B for one type of images, but the inverse occurs for another one. Finding out the best option in each case can be facilitated by retrieving provenance records stored in the MIASS archives. In an improved implementation of a MIA method for image enhancement, the appropriate filter would be chosen dynamically depending on the image type, as illustrated in Fig. 5.

### *B. Requirements for a MIASS*

An integrated MIASS to support all phases of MIA methods lifecycle in the manner suggested by the example in Section IV-A should fulfil the following requirements:

1) Facilitate the *construction of components* by offering the following functionality:
   a) assistance for constructing the component interface with the MIASS (e.g., templates, wizards, code wrappers);
   b) automatic generation of code to obtain input parameters, to display status, progress information, and errors, and to output results; the generated code should be compatible with the execution environment, e.g., a graphical user interface (GUI) for interactive execution and command-line arguments when the component is run from a script;
   c) activation of a chosen external development environment to debug the component's internal implementation.
2) Facilitate the *construction of networks* of components by offering the following functionality:
   a) intuitive GUI to interactively combine components into networks;
   b) constructions for execution control, such as loops, conditional execution, and functional aggregation of components;
   c) a broad repertoire of components to input, manipulate, and display different types of data; and
   d) assistance for selecting and combining components into networks based on data types and records of previous system usage.
3) Support the *execution of MIA methods in varied environments* in a seamless fashion by offering the following functionality:
   a) interactive execution of methods, providing GUI to input parameters and visualize results, as well as debugging tools to run, stop, pause, and inspect the status of networks and components;
   b) activation of methods by external entities in an automated setting the input parameters for the method are obtained from an external control unit (e.g., a scripting language), and the output is stored in a designated location; if the method is integrated into the enterprise IT infrastructure, data should be transported from/to the remote information systems in a transparent manner;
   c) facilities to synchronize automated and interactive execution in situations when user intervention is needed during semi-autonomous processing; the user should be notified about the required intervention (e.g., to provide input data) and provided with the execution trace to reproduce the context accordingly.
4) Support storage and retrieval of *provenance* of the results generated with the system. Data recorded automatically should include the executed method (e.g., implementation, version), input data (e.g., images), parameter values, and associated metadata (e.g., radiological interpretation).
5) Use *HPC* resources transparently for computation and data archival as follows:
   a) distribute the processing workload to remote nodes to guarantee the desired response time during interactive processing or heavy computations;

   b) transparently adopt (remote and distributed) data resources to store large amounts of data in a secure manner for long periods of time; and
   c) transport data and transfer the execution control to the processing nodes and back without user intervention nor awareness.
6) Support *user collaboration* by offering tools to do the following:
   a) facilitate the communication between users participating in different phases of the method's lifecycle; the MIASS should keep track of the participants assuming different user roles in the lifecycle and automatically contact the appropriate user for intervention; all information needed to recreate the circumstances for intervention (input data, parameters) should be provided to the user;
   b) provide collaborative, dispersed, and secure access to data, methods, and tools by users at different locations; it could be CUs of different departments in the same hospital, the CD and ND of academic and industrial partners, or CUs of multiple health centres participating in a clinical trial; patient data privacy should be guaranteed in all cases.

Note that the list mentioned earlier intentionally omits the requirements related to software engineering in a broader sense such as version and revision control. Instead, the presented requirements focus on aspects that are relevant for the construction of MIA software to be integrated and maintained in clinical environments.

## V. Examples of Existing Systems

The requirements proposed in Section IV-B are addressed by capabilities present in the existing systems. Section V-A discusses how the systems presented in Section II fulfil the requirements, and Section V-B presents an example of a real application.

### A. Systems Revisited

The component-based paradigm is acknowledged as an effective problem-solving approach in several fields, being adopted by most systems (AVS, Khoros, DeVIDE, LONI, Kepler, Triana, GridNexus, VLAM-G, etc.). Most of them provide graphical editors to build networks (Req. 2.a), which can be stored, edited, and executed interactively (Req. 3.a). Some also provide mechanisms for control flow (Req. 2.b), such as the aggregation of components into reusable units (Triana, GridNexus) and conditional execution (Kepler). The repertoire of components (Req. 2.c) typically focuses on some application domain. LONI includes popular analysis packages such as FSL [25]; DeVIDE aims at MIA and visualization methods in general, encapsulating VTK [26] and ITK [27]; and Kepler and Triana are distributed with components for several scientific applications in areas such as astronomy and bioinformatics. Assistance for component selection (Req. 2.d) is offered in the form of data typing and polymorphism (Kepler, GridNexus) and file extensions associations (DeVIDE, LONI). IXI and Pegasus can generate networks automatically from provenance data (Req. 2.e).

Facilities for building components (Req. 1) are typically limited in current systems. Assistance to program the component interface (Req. 1.a) is found in the form of templates (DeVIDE, Kepler) and code wrappers (LONI, Triana [28]). Automatic GUI generation for input parameters (Req. 1.b) is offered by many systems (Kepler, VLAM-G), and DeVIDE additionally provides display of progress information. The activation of external development environments (Req. 1.c) is not supported by the studied systems.

Most systems provide a single execution mode, either interactive (Req. 3.a) or initiated by an external control mechanism (Req. 3.b) usually provided via a custom scripting language based in XML.[3] In some cases, networks can be executed either interactively or in batch (Kepler, GridNexus, Triana). Debugging tools for interactive execution are limited to running and stopping the network (Kepler), whereas DeVIDE additionally enables component introspection. Integration into external IT infrastructures (Req. 3.b) has been obtained by running networks as web services (Triana, Taverna, MIAKT), grid services (IXI, $DM^2$), and web-based form applications activated from databases [17]. Synchronization between interactive and automated execution (Req. 3.c) is seldom found in the studied systems. A simple mechanism is implemented in Triana, where the user can start a network interactively, disconnect, and reconnect to check the status.

Recording and retrieving data provenance (Req. 4) are increasingly popular facilities offered for knowledge management in scientific experiments (Chimera, Taverna, IXI, MIAKT). Such data provenance records have been used, for example, for drug discovery [29] and automatic generation of reports in breast cancer screening with mammography [14].

Automatic and transparent distribution of processing (Req. 5.a) is available for computer clusters (VLAM-G, LONI, GAMA) as well as heterogeneous computation nodes (IXI, MIAKT, Triana, Nimrod). Access to remote and distributed data resources (Req. 5.b) is available in several systems (SRB, MIAKT, $DM^2$). Transparent data transport to/from the processing nodes (Req. 5.c) is found in GAMA, IXI, MIAKT, and Pegasus.

Support to collaborative work (Req. 6) typically consists of providing access to physically dispersed data and computing resources (Req. 6.b), being available in most systems. Patient data privacy is usually obtained in isolated environments (e.g., inside the hospital's firewall), whereas global grids are used mostly in experimental settings (GAMA). Mechanisms for collaboration along the MIA method lifecycle (Req. 6.a) were not found in the studied systems.

### B. A MIA Application: MMBE

The matched masked bone elimination (MMBE) method is a preprocessing technique developed at the Academic Medical Center (AMC, Amsterdam, The Netherlands) to enhance the

[3] eXtensible Markup Language, http://www.w3.org/XML

visualization of vasculature in CT angiography (CTA) scans of the head and neck [30], [31]. This method fully automatically eliminates bone voxels from CTA images by using an additional nonenhanced CT scan. First, the two scans (nonenhanced and contrast-enhanced) are aligned using image registration to compensate for movements of the patient in between image acquisitions. A mask is derived from the nonenhanced scan by thresholding and aligned with the contrast-enhanced scan according to the computed registration parameters. By application of the mask, the bone tissue is removed, resulting in a bone-free enhanced scan that is directly suitable for volumetric visualization. The method is currently applied to all CTA examinations of the brain and neck region performed at the AMC [32].

The MMBE method has evolved according to the lifecycle described in Section III. The original method [30] adopts rigid registration, being applicable only to brain scans. An extended version of this method [31] adopts piecewise registration to cope with bones that move with respect to each other. Current efforts aim at improving the segmentation of bone located near to contrast-enhanced vessels with a multi-scale approach [33]. The component-based approach was adopted to implement the method using open-source libraries (VTK, ITK). Phantom studies were performed for parameter optimization. An evaluation study based on patient data was conducted to determine the method's performance for clinical data.

A distributed workflow management system (DWMS [34]) has been adopted to insert the method into the clinical routine in a seamless fashion. After image acquisition, the scans are automatically sent to a DICOM node that triggers the proper MIA workflow based on rules described in terms of metadata contained in the DICOM header. The workflow consists of a network of components that performs image analysis (e.g., MMBE) and the necessary data logistics (transfer images to/from the processing, archival, and visualization units). The network also includes components that send a notification to the radiologist upon completion, indicating where the data are available for inspection. When a failure occurs, the support team is also notified. The DWMS was implemented using a web-service-oriented architecture, adopting SOAP[4] messages for communication among components. The processing load is automatically balanced by distribution on a grid of pre-registered computers using priorities and monitoring functions. Failed workflows are automatically recovered. The system also stores provenance data (status of performed workflows, parameters, and results).

Although no integrated MIASS environment was available during the evolution of the MMBE technique, its deployment in the clinical routine has been accomplished with a system that partially fulfils requirements 3–6.

## VI. Discussion and Conclusions

In this paper, the development, evaluation, and usage of MIA software is presented as a process composed of phases in which varied activities are performed by users assuming different roles (Section III). The suggested phases serve as a framework for the development of MIA solutions that are intended for use in clinical tasks. In the ideal and simple scenario sketched in Section IV-A, the activities at all phases are supported and facilitated by an integrated computational environment that fulfils the requirements defined in Section IV-B. Although the presented scenario is very simple, it illustrates well the potential of adopting an integrated environment for MIA method development, evaluation, and clinical application. More complex scenarios could also be envisioned to illustrate the potential of other functionalities defined by the requirements. For example, facilities for virtual collaboration and remote access to resources could be exploited when users, image servers, data storage servers, or computation nodes are physically dispersed. In another example, a broader repertoire of components could be available to support methods that process different data types, such as geometric descriptions or values corresponding to features measured from the image. And finally, facilities for HPC and synchronization of interactive and automated execution could be adopted for real-time image-guided surgery. In summary, the proposed requirements involve a broad range of functionalities that are potentially useful in a large number of situations found in medical image analysis. Moreover, the requirements can be used as a framework to evaluate, compare, and improve existing software systems to support MIA development. As shown in Section V, many of the requirements are fulfilled, even if partially, in a large number of existing systems. However, none of the studied systems completely fulfils all the requirements to support the lifecycle of MIA methods as in the suggested scenario.

The current goal of our research is to pursue the development of a system approximating the ideal MIASS as much as possible. This is performed in the context of the Virtual Laboratory for e-Science (VL-e[5]), a project that aims at the development and experimental evaluation of generic functionalities of PSEs to support a wide class of e-Science application environments.

Initially, the properties of existing systems have been studied based on the requirements in Section IV-B. The first candidates were systems developed within the scope of VL-e, namely DeVIDE, DWMS, GAMA, and VLAM-G, which offer complementary functionality. DeVIDE fulfils several requirements for component and network construction and interactive execution (1.a, 1.b, 2.a, 2.d, and 3.a), offering a comfortable environment for the development and experimentation with new MIA methods. DWMS aims at the integration of MIA applications in the radiological ICT environment, improving the interoperability between image acquisition devices, HPC resources, clinicians, and researchers (requirements 3.b, 4, 5.a, 5.b, 5.c, and 6.b). GAMA focuses on the transparent employment of HPC for processing in dedicated applications, fulfilling requirements 5.a and 5.c. Finally, VLAM-G offers transparent access to HPC resources to execute networks in an interactive environment (requirements 2.a, 3.a, and 5.a). It additionally supports collaborative and remote control of experiments (requirement 6.b) via a database that can be customized to particular applications.

[4]Simple Object Access Protocol, http://www.w3.org/TR/soap

[5]http://www.vl-e.nl/

The next step will be to investigate how these systems can be combined and/or extended to implement the required functionality, which is a very challenging task. The requirements proposed in Section IV-B have been useful to bring structure and a clear target to our current research efforts in the construction of an integrated environment to support the development of MIA methods.

## REFERENCES


[1] J. Cunha, O. Rana, and P. Medeiros, "Future trends in distributed applications and problem-solving environments," *Future Generat. Comput. Syst.*, vol. 21, no. 6, pp. 843–855, Jun. 2005.

[2] C. Johnson, S. Parker, D. Weinstein, and S. Heffernan, "Component-based, problem-solving environments for large scientific computing," *Concurrency Comput. Practice Experience*, vol. 14, no. 13–15, pp. 1337–1349, Jan. 2002.

[3] C. Upson, T. F. Faulhaber Jr., D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. van Dam, "The application visualization system: A computational environment for scientific visualization," *IEEE Comput. Graph. Appl.*, vol. 9, no. 4, pp. 30–42, Jul. 1989.

[4] J. Rasure and C. S. Williams, "An integrated data flow language and software development environment," *J. Visual Languages Comput.*, vol. 2, no. 3, pp. 217–246, Sep. 1991.

[5] C. Botha, "DeVIDE—The Delft visualization and image processing development environment," Technical University of Delft, Delft, The Netherlands, Tech. Rep., May 2005.

[6] D. Rex, J. Ma, and A. Toga, "The LONI pipeline processing environment," *NeuroImage*, vol. 19, no. 3, pp. 1033–1048, Jul. 2003.

[7] B. Ludascher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the Kepler system," *Concurrency Comput. Practice Experience*, vol. 18, no. 10, pp. 1039–1065, 2005.

[8] H. Afsarmanesh, R. G. Belleman, A. Belloum, A. Benabdelkader, G. B. Eijkel, A. Frenkel, C. Garita, D. L. Groep, R. A. Heeren, Z. W. Hendrikse, L. Hertzberger, E. Kaletas, V. Korkhov, C. de Laat, P. A. Sloot, D. Vasunin, A. Visser, and H. Yakali, "VLAM-G: A grid-based virtual laboratory," *Sci. Program.*, vol. 10, no. 2, pp. 173–181, 2002.

[9] I. Taylor, M. Shields, I. Wang, and O. Rana, "Triana applications within grid computing and peer to peer environments," *J. Grid Comput.*, vol. 1, no. 2, pp. 199–217, Jun. 2003.

[10] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li, "Taverna: A tool for the composition and enactment of bioinformatics workflows," *Bioinform. J.*, vol. 10, no. 17, pp. 3045–3054, Jun. 2004.

[11] J. L. Brown, C. Ferner, T. Hudson, A. Stapleton, R. Vetter, T. Garland, A. Martin, J. Martin, A. Rawls, W. Shipman, and M. Wood, "GridNexus: A grid services scientific workflow system," *Int. J. Comp. Inform. Sci. (IJCIS)*, vol. 6, no. 2, Jun. 2005.

[12] A. Rowland, M. Burns, T. Hartkens, J. Hajnal, D. Rueckert, and D. Hill, "Information extraction from images (IXI): Image processing workflows using a grid enabled image database," presented at the DiDaMIC Workshop (MICCAI), Rennes, France, 2004.

[13] M. Burns, A. Rowland, D. Rueckert, J. Hajnal, K. Leung, D. Hill, and J. Vickers, "Information extraction from images (IXI): Grid services for medical imaging," presented at the DiDaMIC Workshop (MICCAI), Rennes, France, 2004.

[14] N. Shadbolt, P. Lewis, S. Dasmahapatra, D. Dupplaw, B. Hu, and H. Lewis, "MIAKT: Combining grid and web services for collaborative medical decision making," presented at the UK e-Science All Hands Meeting, Nottingham, U.K., 2004.

[15] J. Montagnat *et al.*, "Medical images simulation, storage, and processing on the european datagrid testbed," *J. Grid Comput.*, vol. 2, pp. 387–400, Dec. 2004.

[16] H. Duque, J. Montagnat, J. Pierson, L. Brunie, and I. E. Magnin, "DM2: A distributed medical data manager for grids," in *Proc. IEEE CCGRID*, Tokyo, Japan, 2003, pp. 606–611.

[17] L. Liu, D. Meier, M. Polgar-Turcsanyi, P. Karkocha, R. Bakshi, and C. Guttmann, "Event-driven workflow management for medical image processing and analysis in a large image database," presented at the DiDaMIC Workshop (MICCAI), Rennes, France, 2004.

[18] A. Rajasekar, M. Wan, R. Moore, W. Schroeder, G. Kremenek, A. Ja-gatheesan, C. Cowart, B. Zhu, S. Chen, and R. Olschanowsky, "Storage resource broker—managing distributed data in a grid," *Comput. Soc. India J.*, vol. 33, no. 4, pp. 42–54, Oct. 2003.

[19] I. Foster, J.-S. Vöckler, M. Wilde, and Y. Zhao, "Chimera: A virtual data system for representing, querying, and automating data derivation," in *Proc. 14th IEEE Int. Symp. Sci. Database Manag.*, Edinburgh, U.K., 2002, pp. 37–46.

[20] E. Deelman, J. Blythe, Y. Gil, C. Kesselmana, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny, "Pegasus: Mapping scientific workflows onto the grid," in *Proc. Across Grids Conf.*, Nicosia, Cyprus, 2004, pp. 139–145.

[21] A. Bucur, R. Kootstra, and R. G. Belleman, "A grid architecture for medical applications," in *Proc. Healthgrid 2005,* ser. Studies in Health Technology and Informatics, vol. 12, 2005, pp. 127–137.

[22] F. Brooks, *The Mythical Man-Month: Essays on Software Engineering*. Reading, MA: Addison-Wesley, 1995.

[23] D. Parsons, A. Rashid, A. Speck, and A. Telea, "A framework for object oriented frameworks design," in *Proc. Technol. Object-Oriented Languages and Syst.*, Nancy, France, 1999, pp. 141–151.

[24] D. Abramson, R. Sosic, J. Giddy, and B. Hall, "Nimrod: A tool for performing parametised simulations using distributed workstations," presented at the 4th IEEE Symp. High Perform. Distrib. Comput., Pentagon City, VA, 1995.

[25] S. Smith, M. Jenkinson, M. Woolrich, C. Beckmann, T. Behrens, H. Johansen-Berg, P. Bannister, M. D. Luca, I. Drobnjak, D. Flitney, R. Niazy, J. Saunders, J. Vickers, Y. Zhang, N. de Stefano, J. Brady, and P. Matthews, "Advances in functional and structural MR image analysis and implementation as FSL," *NeuroImage*, vol. 23, pp. 208–219, 2004.

[26] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit, 3rd ed.*, New York, NY: Kitware Inc., 2002.

[27] L. Ibánez, W. Schroeder, L. Ng, and J. Cates, *The ITK Software Guide*. New York, NY: Kitware Inc., 2003.

[28] Y. Huang, I. Taylor, D. W. Walker, and R. Davies, "Wrapping legacy codes for grid-based applications," in *Proc. 17th Int. Parallel and Distrib. Process. Symp. (IPDPS)*, Nice, France, 2003, pp. 139–145.

[29] K. Leung, M. Holden, R. Heckemann, N. Saeed, K. Brooks, J. Buckton, K. Changani, D. Reid, D. Rueckert, J. Hajnal, and D. Hill, "Use of data provenance and the grid in medical image analsyis and drug discovery—an IXI exemplar," presented at the UK e-Science All Hands Meeting, Nottingham, U.K., 2004.

[30] H. Venema, F. Hulsmans, and G. den Heeten, "CT angiography of the circle of Willis and intracranial internal carotid arteries: Maximum intensity projection with matched mask bone elimination-feasibility study," *Radiology*, vol. 218, no. 3, pp. 893–8, Mar. 2001.

[31] M. van Straten, H. Venema, G. Streekstra, C. Majoie, G. den Heeten, and C. Grimbergen, "Removal of bone in CT angiography of the cervical arteries by piecewise matched mask bone elimination," *Med. Phys.*, vol. 31, no. 10, pp. 2924–33, Oct. 2004.

[32] M. van Straten, C. Majoie, H. Venema, L. Ciancibello, and K. Subramanyan, "Automatic bone removal in CT angiography," *Medica Mundi*, vol. 49, no. 1, pp. 4–8, May 2005.

[33] H. G. van Andel, H. Venema, G. Streekstra, M. van Straten, G. den Heeten, and C. Grimbergen, "Multi-scale matched mask bone elimination (multi-scale MMBE): Improved automatic bone removal in computed tomography angiography (CTA) images," in *Proc. RSNA Conf.,* Chicago, IL, 2005, p. 480.

[34] J. G. Snel, S. D. Olabarriaga, J. Alkemade, H. G. van Andel, A. J. Nederveen, C. B. Majoie, G. J. den Heeten, M. van Straten, and R. G. Belleman, "A distributed workflow management system for automated medical image analysis and logistics," in *Proc. 19th IEEE Int. Symp. Computer-Based Med. Syst.,* Salt Lake City, UT, 2006, pp. 733–738.


Authors' photographs and biographies not available at the time of publication.