# An Integrated Instrument Control and Informatics System for Combinatorial Materials Research

M. J. Harvey, D. Scott, and P. V. Coveney*

Centre for Computational Science, Department of Chemistry, University College London,
Christopher Ingold Laboratories, 20 Gordon Street, London WC1H 0AJ, United Kingdom

The use of high-throughput synthesis and characterization techniques is increasingly prevalent in materials science research. We describe the London University Search Instrument, a research apparatus designed for the high-throughput synthesis and characterization of thick-film sample libraries of ceramic compounds. The instrument is constructed largely from commodity components, which pose particular engineering challenges for achieving the automated operation required for efficient high-throughput experimentation. This paper describes the architecture and implementation of the software system that provides integrated instrument control and data management functions.

## 1. INTRODUCTION

High-throughput, combinatorial synthesis and screening techniques are increasingly utilized in materials science research.[1] High-throughput experimentation (HTE) places particular engineering demands on instrument design: in order to operate efficiently, extensive device automation is required and an integrated approach to data management must be adopted.

High-throughput screening techniques have the potential to generate large volumes of data both directly from measurements performed during screening and indirectly from the metadata arising from the initial sample synthesis and later processes. These metadata are essential for maintaining the provenance library samples over their life cycle. Thus, a critical component of a HTE system is an informatics system for managing data and preserving the relationships between them.

In essence, an informatics system provides the infrastructure for data collection, storage, processing, and presentation. Desirable qualities for the design of informatics systems for combinatorial research are variously discussed in refs 2 and 3. Although commercial software is available, both for data management[4] and device control,[5] it is frequently costly and often additional work is required to integrate it with specific instrumentation. Consequently, local solutions are often developed,[6] although these are typically customized for a particular application and may suffer from a lack of generality.

In this paper, we present an overview of a flexible architectural approach to device automation and data management developed to control the London University Search Instrument (LUSI). Section 2 gives an overview of LUSI; Sections 3 and 4 detail data management and device control systems, while in Section 5, we describe user interaction with the instrument and its use in a grid computing context[7] in which a remotely accessible, Web service[8] control interface for the instrument is provided.

* Corresponding author e-mail: p.v.coveney@ucl.ac.uk.

## 2. THE LONDON UNIVERSITY SEARCH INSTRUMENT

**2.1. Overview.** LUSI[9] is a research tool for the investigation of high-throughput synthesis and screening techniques for discrete-sample, thick-film libraries of ceramic compounds synthesized from high-component-number oxide inks. It is assembled from commodity components and is intended to be flexibly reconfigurable, permitting the addition or exchange of individual devices as research demands dictate. Current research[10] involves studies of ferroelectric and dielectric characteristics of perovskite systems. The ink-jet printing technique for ceramic compound production is discussed in refs 11−13.

The LUSI equipment is comprised of the following systems:

• An eight-nozzle aspirating−dispensing ink-jet printer workstation (ProSys 6000, Cartesian Ltd, U. K.). Each nozzle is independently controlled by 192 000-step syringes. The printer has a 20 nL dispensing capability.

• An A3 (295 × 420 mm$^2$) $X$−$Y$ table sample building site with a capacity for 100 sample slides and three 96-well plates used for ink mixing.

• A furnace with four independent programatically controlled (Eurotherm model 2408 with Modbus interface) temperature zones.

• A precision $X$−$Y$ measurement table with a programatically controlled 700 K hotplate (Omron Electronics Ltd, U. K.).

• A $Z$-axis probe armature (LabMan Ltd, U. K.) colocated with the $X$−$Y$ table. $Z$ displacement is controlled by direct application of force by the picker.

• An impedance phase analyzer (Agilent/Hewlett-Packard model 4194A).

These devices are installed within a gantry frame from which is suspended a robotic picker (LabMan Ltd, U. K.) used to transfer library slides between devices.

With the exception of the gantry and picker, which were designed to the specific requirements of the instrument, all of the devices are commodity items. The instrument is
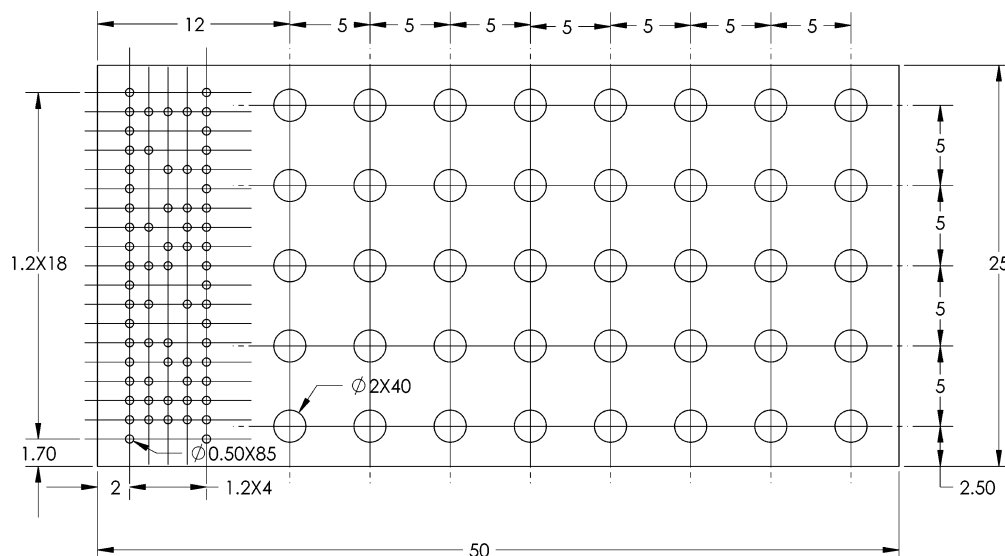
**Figure 1.** Layout of a typical library slide. In this case, samples are arranged in an 8 × 5 array. The dot region to the left of the slide contains a binary representation of the unique serial number of the slide and is machine-readable. Measurements in mm. Reprinted courtesy of L. Chen, Queen Mary, University of London.

intended to be flexibly reconfigurable, permitting the addition or exchange of individual devices as demands dictate.

The instrument is currently employed within the EPSRC-funded project Discovery of New Functional Oxides (FOXD; http://www.foxd.org; EPSRC, Engineering and Physical Sciences Research Council).

**2.2. Library Fabrication.** The elements of the combinatorial library produced by the instrument consist of two-dimensional arrays of discrete samples of varying composition printed onto alumina (99%) substrates. A representative layout is shown in Figure 1. To accommodate different analytical objectives, slides may be prepared with a coating of platinum dag to provide a ground electrode.

From a high-level perspective, the fabrication process is as follows:

• Suspensions of the appropriate precursor oxides are prepared by hand and manually dispensed into well plates.

• The printer is used to create mixtures representing the samples within the compositional range specified by the user.

• Sample mixtures are deposited on prepared substrates according to a specified arrangement.

• Slides are transferred to the furnace for sintering according to a particular temperature profile.

Following fabrication, slides are transferred to storage units. Once in storage, slides may be either shipped off-site for independent screening or screened in situ by LUSI using locally available analytical equipment before being returned to storage.

### 3. DATA MODEL

**3.1. Data Model.** The informatics system stores metadata associated with library sample compositions and synthesis, related raw measurement data, and subsequently derived data. The relationships between these entities are explicitly defined and shown in Figure 2.

A software tool is provided to facilitate the generation of an experiment plan specification which parametrizes the synthesis procedure as follows:

• A set of percent volume ranges and step intervals for precursor ink combinations. Samples are produced with compositions satisfying this specification.

• A sintering profile, expressed as temperature set points and durations.

• A slide layout, specifying the substrate coating, physical layout of the samples on the slide, and the number of samples to be produced for each particular composition.

The public interface layer (see Section 4.4) is responsible for generating a manifest of sample compositions and slide layouts from the specification. For production purposes, slides are packed into batches of 100. This value reflects a hardware limitation on the maximum number of slides which may be printed and sintered simultaneously.

Measurement data may be associated with either entire library slides or individual samples. Results arising from a subsequent analysis can also be stored within the system. In this instance, the relationship between the original and derived data is also preserved. For data provenance purposes, all measurement and analytical data sets are associated with the user responsible for their creation.

Frequently, the researcher may wish to record notes or observations concerning some aspect of an entity which does not fit into any particular structure. To capture this often valuable data, a facility is provided to associate a free-form text annotation with any database entity. Client tools provide an electronic notebook function for creating and viewing these annotations.

**3.2. Database Implementation.** The data model representing the experimental metadata is expressed as a relational database schema and implemented on a PostgreSQL[14] relational database. Access to the database is mediated using either a Java database connectivity (JDBC)[15] connector or an Open Grid Services Architecture data access and integration (OGSA-DAI)[16] interface. The latter is discussed in Section 5.2.

Relational databases make a poor choice of storage for the multivariate multidimensional arrays of floating-point numbers typical of the measurements made during the
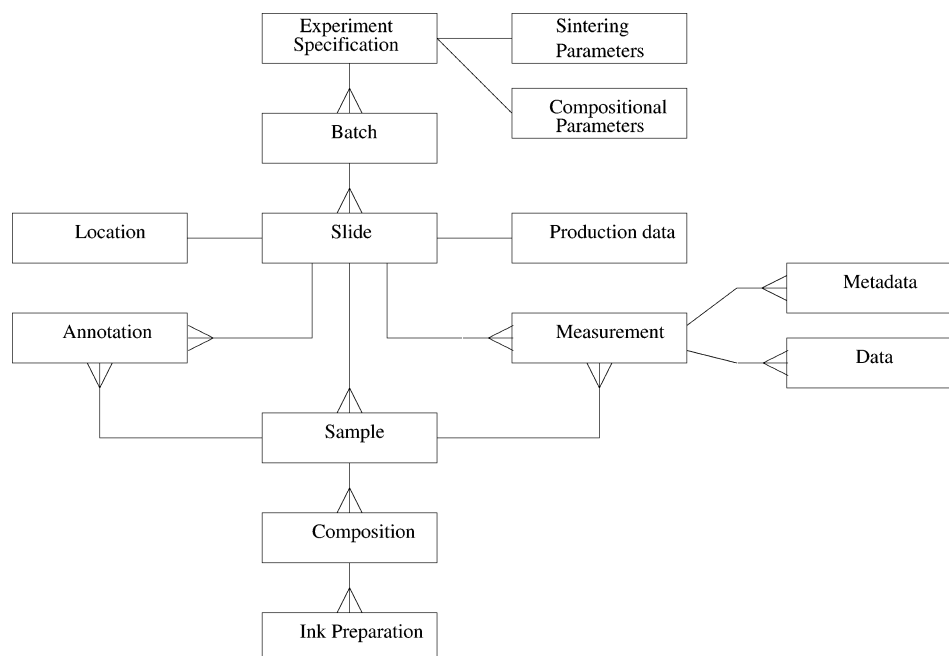
**Figure 2.** Simplified entity-relationship diagram of the LUSI database. The design reflects the synthesis and screening workflow. Forked connectors represent one-to-many relationships.

screening process performed by LUSI. Further, such data often require a numerical analysis which is not conducive to expression in a SQL query. Consequently, measurement data are stored directly on a file system in a filestore with a directory structure. The structure of the directory hierarchy reflects the entity relationships and is of the form

> [root]
> / <specification ID>
> / <slide ID>
> / <sample ID>
> / <measurement ID>
> / [data files]

Within the filestore, data sets are stored as XDR-encoded (XDR, eXternal Data Representation) byte streams.[14] XDR provides an efficient, platform-independent encoding which has near-universal support. To accommodate multivariate or multidimensional data sets, each data set file has an associated XML[17] document describing its format.

**3.3. Data Collection.** Analytical instruments employed within the FOXD project include an evanescent microwave probe, X-ray diffractometer, impedance analyzer, and focused ion beam secondary ion mass spectrometer (FIB-SIMS).

With the exception of the impedance spectrometer, these devices are not co-located with the instrument and are operated independently of the software described herein. Each device has a provision for automated high-throughput screening and produces output electronically. The public interface for the informatics system provides programmatic and manual mechanisms for uploading measurements and associating them with sample records (see section 4.4).

Each measurement device produces data in a custom electronic file format, for each of which a parser has been developed to extract the salient data (for provenance purposes, the source files are retained in the filestore). Multivariate or multidimensional data blocks are stored on the file system, while metadata or single-valued quantities

which are likely to be the subject of a database query are stored directly in the database.

This scheme facilitates the automatic analysis of measurement data by incorporating the analysis procedure immediately after the upload and parsing step.

## 4. SOFTWARE ARCHITECTURE

As a consequence of the design of the LUSI instrument, each constituent device must be independently controlled via a vendor-specific interface or software package.

To present a unified interface to the instrument, in which each device may be treated as a constituent of a subsystem, it is necessary to construct a software system to manage each component. The design chosen is hierarchical, with each layer representing increasing abstraction in device operation. Figure 3 gives a block-diagram view of the individual tiers of the software stack. Each layer is described below in Section 4.1.

The logical control, scheduler, and public interface software is developed in Java.[18] Java provides a stable, high-level, object-oriented programming environment. Although designed as a platform-independent environment and lacking functions for directly communicating with hardware devices, Java provides the ability to programmatically interface with native C code or libraries (with C calling semantics). This capability is used for interfacing with devices which require direct hardware control or which have vendor software provided as native libraries.

**4.1. Device Control.** The design of LUSI is inherently modular and mutable, with each device within the instrument having particular control interface requirements. For each device, a simple software component is created which encapsulates implementation details of communicating with and controlling the device. To permit control of the device by higher levels of software, each component provides a network-visible interface.
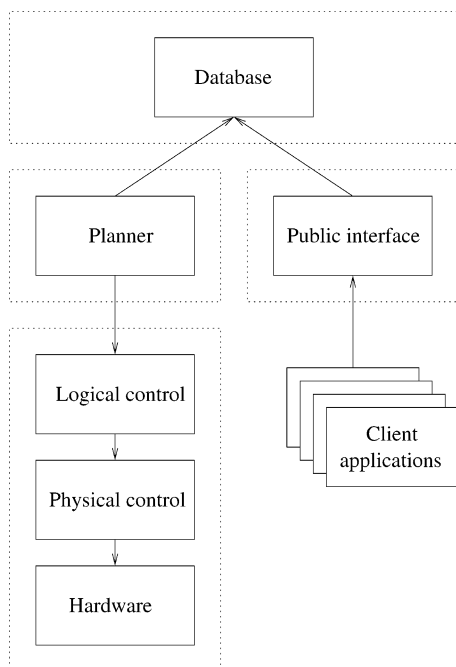
INSTRUMENT CONTROL AND INFORMATICS SYSTEM

*J. Chem. Inf. Model., Vol. 46, No. 3, 2006* **1029**



**Figure 3.** Block diagram of software architecture. Dotted box indicates separate administrative domains. Arrows between boxes indicate direction of communication.

The commands exposed through the interface consist of a simple set of directives which describe the functions of the device. Commands are human-readable American Standard Code for Information Interchange (ASCII) text with responses consisting of a set of numeric codes representing success and possible error states. While the command sets obviously vary between devices, the communications semantics remain constant, permitting the reuse of client-side communications code.

Each control component operates in a separate process space and is stateless with respect to communications with clients. This permits components to be transient (i.e., the software may be terminated and subsequently restarted) without affecting the stability of the system, beyond causing a controlled (and not necessarily fatal) fault within any procedure which requires the presence of that component.

**4.2. Logical Control.** The logical control software layer provides a single, unified interface to the instrument. This interface exposes high-level, process-oriented operations.

The logical model abstracts the instrument into a set of interacting subsystems. Each subsystem is associated with zero or more physical devices and is represented by a software component. The software retains all states associated with the subsystem and manages the constituent devices (see Figure 4).

Each subsystem is defined according to the following:

• **Spatial Extent.** This is the volume of space, defined within the coordinate system of the enclosing robot gantry, in which the subsystem is taken to exist (see Figure 5). Within this volume, the subsystem software component has exclusive control of the picker, which may be operated arbitrarily. This is necessary to accommodate subsystems which exhibit interactions between constituent devices: in the case of the printer, the print head obscures picker access to slide locations and must be moved appropriately in order to access certain slide locations.

• **Transfer Points.** These are the points residing on the surface of the subsystem volume (red squares in Figure 5) which indicate the points at which picker control can be acquired or relinquished by the subsystem software.

• **Slide Capacity.** This indicates the locations within the volume which are valid positions for a slide. The subsystem
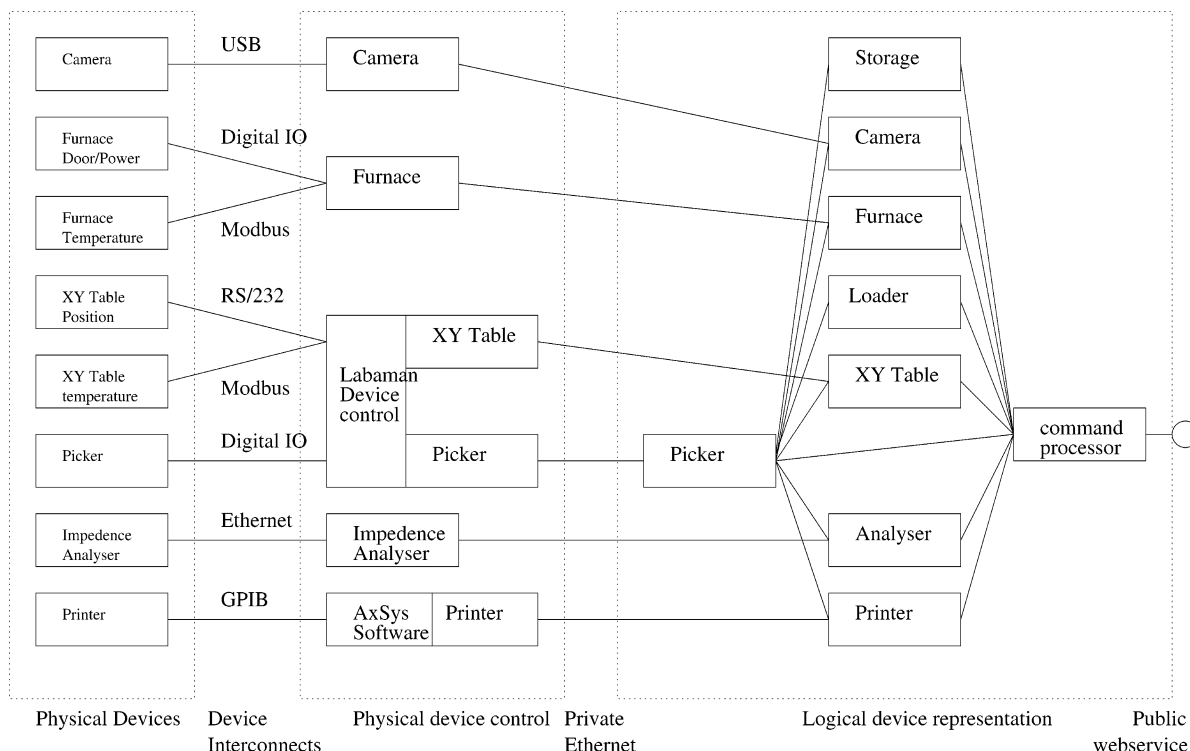


**Figure 4.** Logical structure of the LUSI control software. Individual hardware devices (left) are controlled via physical control layer components either directly or via vendor software (e.g., printer and picker). Software components that provide logical representations of the instrument subsystems (right) direct the physical device control layer. The command processor (far right) provides the public interface to the instrument, decomposing external commands into logical layer directives. All communications are initiated from right to left.
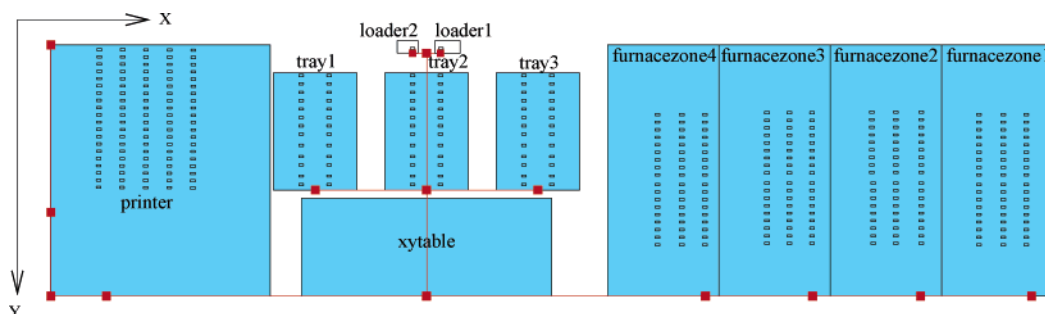
**Figure 5.** Plan view of the subsystem layout. Blue rectangles indicate the extent of the volume occupied by the devices comprising each subsystem. Small black rectangles within these indicate valid slide locations. Red lines represent the predefined routes for picker motion. Route endpoints are marked by solid red rectangles. The $z$ coordinate increases into the page. Subsystem volumes are taken to occupy the entire $z$ range. Routes are located on the $z = 0$ plane.

software maintains records of the locations and serial numbers of slides within the subsystem.

• **Associated Devices.** These are the physical devices which the subsystem software requires access to. Not all subsystems control physical devices: the loader subsystem (Figure 4), for example, simply represents a location at which fresh library slides are stacked.

In addition to related support functions, the subsystem software component implements the following major functions:

• **is_accessible.** This indicates whether the device is in a state which permits the picker to access the slide locations within it.

• **make_accessible.** This performs subsystem-specific operations to attempt to return the subsystem to a state in which the picker may access slide locations.

• **insert_slide.** This is an attempt to insert the slide presently held by the picker into a slide location.

• **remove_slide.** This is an attempt to remove the slide with the specified serial number from the subsystem, returning the picker to a transfer point.

• **operate.** This performs a subsystem-specific operation, for example, *print* or *sinter*. Operation-dependent parameters may be passed as arguments.

Subsystems are interconnected via predefined routes between predetermined waypoints (red lines and rectangles in Figure 5), which determine paths used by the picker when transferring slides. Routes are defined such that movement between adjacent waypoints requires picker movement along only a single axis, restricting movement to specific loci.

The use of manually determined, static waypoints has the benefit of reducing the likelihood of collision or other adverse interaction between the picker and equipment at the expense of nonoptimal routing. Because the printing and sintering durations dominate the synthesis time, this is considered a negligible cost.

The logical control software provides a single Web service interface, access to which is secured using authentication via HTTPS transport layer security and is restricted to connections from the planner layer (recall Figure 3).

**4.3. Planner.** The planner layer implements the workflow algorithms which describe the experimental synthesis and screening procedures. A representation of the main loop is given in Figure 6. The main loop is executed repetitively, with a sleep period of 5 s between iterations.

Each process involving interaction with LUSI is decomposed into instrument control commands which are communicated to the logical layer for execution.

Inevitably, there remain tasks which cannot be automated with the available hardware and which necessitate the intervention of a human operator. Outstanding operator tasks are logged in the database and an alert sent to users who are registered as instrument operators. For safety reasons, the instrument is placed into a safe state in which all devices are inactive until the task is marked as complete. For accounting purposes, the time of completion and name of the operator are recorded in the production logs.

The workflow encapsulated in the planner is manually coded: changes to the workflow require concomitant programmatic changes.

**4.4. Public Interface.** The public interface to the instrument exposes access to the operations which govern the function of the integrated instrument. These operations fall into several categories:

• **Workflow** operations provide the means to submit input describing a particular experiment which the user wishes the instrument to perform. The planner layer constructs an appropriate workflow based upon this input.

• **Operator** operations allow the technician operating the instrument access to status and diagnostic information about individual devices and provide the means for the planner layer to direct the operator to perform a specific manual (unautomated) task (e.g., "load blank slides", "replace mixing well plate #2", etc.). The operator is expected to provide a binary "success/fail" response.

• **Informatics** operations permit a user access to the datastore to perform search queries, add annotations, and upload measurement files.

• **Administration** operations allow the management of user accounts and access privileges. User accounts with the "administrator" privilege may manage the accounts of other users and may conditionally restrict access to particular public interface operations.

The public interface is implemented with a Web service interface, access to which is secured using mutually authenticated HTTPS sessions. The credentials provided by the client (in the form of an X.509 certificate[19]) are used to authorize the user's access to the system. Our use of Web services is a consequence of the criteria discussed in Section 5.2.

Providing a programmatic interface to the system reflects the usual user- interface programming convention of main-
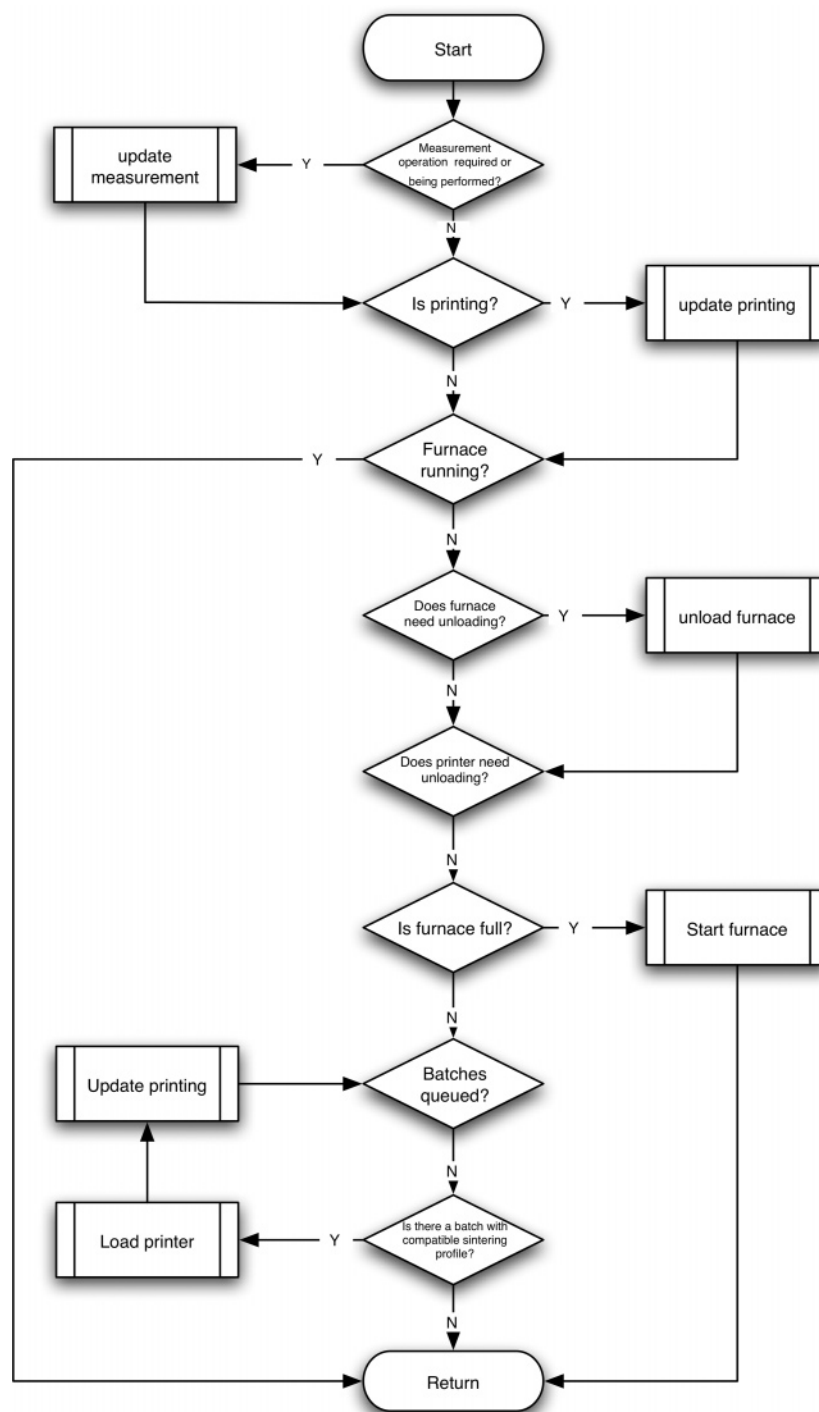
**Figure 6.** Flowchart representation of the process workflow algorithm for operation of the instrument. Subcomponents of the algorithm are represented by the bordered boxes. This algorithm is implemented in the planner layer (Section 4.3).

taining a separation between program logic and user-interface presentation. Observing this division permits the independent development of the instrument software and client applications. Taking advantage of this, several client applications have been developed including a Web site presentation and an interactive production design and data analysis Java application.

## 5. INSTRUMENT OPERATION

**5.1. Client Tools.** To effectively use the instrument, the user requires a simple graphical or textual interface to the software system. The client that has been developed is Web-based and provides functions for experiment design (specification of synthesis and screening parameters, Figure 7a), importing of measurement data sets, and common data query operations. For interactive data visualization operations, Web pages provide a poor user experience principally because of the latency incurred in data transfer between the client and the remote server.[2] Consequently, an interactive data analysis application has been developed in Java (Figure 7b).

It is at the client tool level at which adapter code for providing interoperability with other systems is best implemented. For example, a future client application may be developed to convert between the native data representation
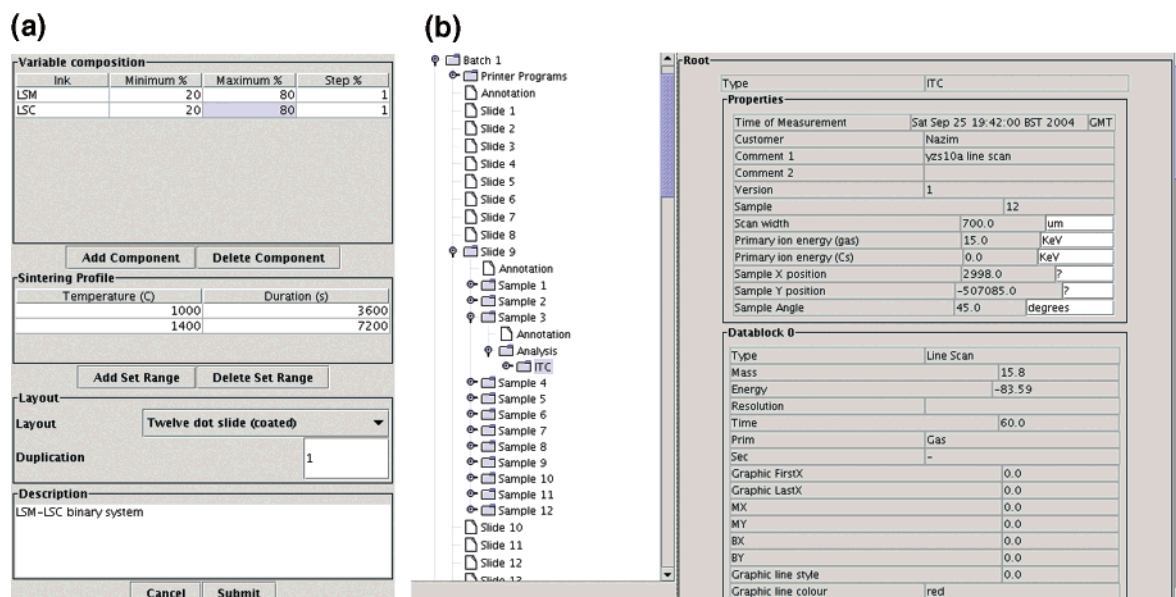
**Figure 7.** Two views of the user interface: part a shows the tool with which the parameters of an experiment are specified; part b shows a view of the data representing the resulting library (left) and a representative set of measurement data which is associated with the selected sample (in this case, sample 3 of slide 9 of batch 1) (right).

and MatML,[20] a widely used XML schema for the representation of materials information.

**5.2. Operation within a Grid Computing Environment.** The grid computing model[21] of distributed computing promotes transparent use of computational resources which are distributed across administrative and geographical domains.

The prevalent software model for grid computing is that of the service-oriented architecture (SOA). Within a SOA environment, software components comprise loosely coupled, highly interoperable application services ("grid services"). SOAs are predominantly implemented as Web services: entities accessible via SOAP[22] operations, the capabilities of which are described by Web service description language[23] documents.

The SOA methodology is incorporated in the design of the public interface to the LUSI software: its Web services presentation facilitates the integration of the instrument into a larger system of software components. This has direct benefits when the instrument is used as part of a complex workflow.

For example, a multiobjective parameter optimization (in which compositions yielding simultaneously optimal values of a set of properties are sought) is most practically conducted iteratively (Figure 8). The presentation of a programmatic interface to the instrument facilitates the development of "robot scientist" software capable of driving cycles of experimentation in a closed loop by employing techniques from artificial intelligence (AI) research to generate and test hypotheses about composition−property relationships.[24,25] It is anticipated that such "closed- loop", AI-driven experimentation will constitute a significant component of future research with the instrument.

In addition to providing a SOA presentation, the informatics software has been extended to couple with an OGSA-DAI[16] Web service, which provides a SOA representation of a database system. This abstracts implementation and vendor-specific details of the database system behind a
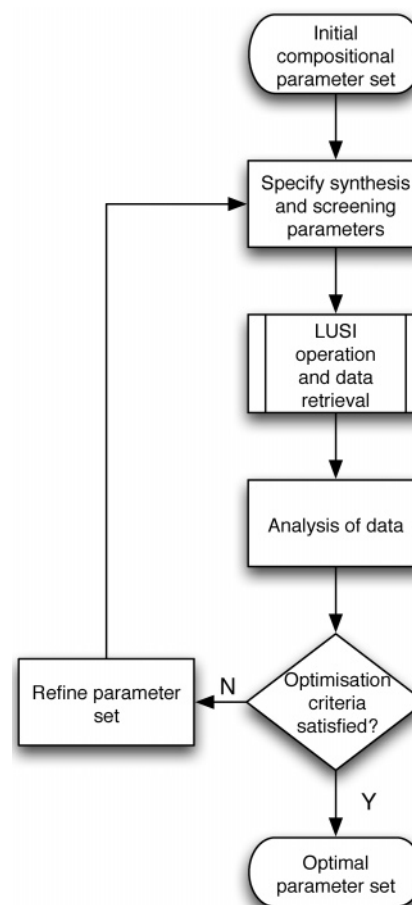


**Figure 8.** Representation of the workflow for an iterative optimizing parameter search in which the LUSI instrument is treated as a data source.

uniform Web service. Consequently, the data may be transparently replicated across or relocated to new database systems without requiring reconfiguration of the informatics system. This is of benefit when upgrading the computational resources used to support the database as the volume of

INSTRUMENT CONTROL AND INFORMATICS SYSTEM

*J. Chem. Inf. Model., Vol. 46, No. 3, 2006* **1033**

resident data increases.

## 6. CONCLUSIONS

HTE, in which experimental procedures traditionally performed manually are automated for increased efficiency, is increasingly prevalent in materials science research. For efficient HTE work, particular consideration must be given to equipment design: typically, several experimental steps will be automated (for example, sample production followed by analytical screening), which necessitates the integration of separate pieces of apparatus into a single, coordinated system.

We have presented an approach to designing software for the control of instrumentation composed of independent devices. This architecture employs a modular scheme in which each individual hardware device is represented by an abstract software component which encapsulates the details of device control. The workflow that characterizes the operation of the entire instrument is encoded in the planner software component, which in turn controls the operation of the constituent devices through their software representations. Closely integrated with the device control software is an informatics system which maintains records of instrument operation and data generated by analytical devices.

This design has been adopted for the software system of the London University Search Instrument. We have found that the modularization of device control, coupled with the centralized workflow in the planner, yields a system which is readily adaptable to changes to the constituent devices and intended operation of the instrument.

Furthermore, the provision of the programmatic, grid service interface to the instrument facilitates anticipated future research in which LUSI is itself treated as a component in a larger system.

## ACKNOWLEDGMENT

## REFERENCES AND NOTES

(1) Woo, S.; Kim, K.; Cho, H.; Oh, K.; Jeon, M.; Tarte, N.; Kim, T.; Mahmood, A. Current Status of Combinatorial and High-Throughput Methods for Discovering New Materials and Catalysts. *QSAR Comb. Sci.* **2005**, *24*, 138−154.

(2) Meguro, S.; Ohnishi, T.; Lippmaa, M.; Koinuma, H. Elements of Informatics for Combinatorial Solid-State Materials Science. *Meas. Sci. Technol.* **2005**, *16*, 309−316.

(3) Zhang, W.; Fasolka, M. J.; Karim, J.; Amis, E. J. An Informatics Infrastructure for Combinatorial and High-Throughpuit Materials Research Built on Open Source Code. *Meas. Sci. Technol.* **2005**, *16*, 261−269.

(4) Adams, N.; Schubert, U. S. Software Solutions for Combinatorial and High-Throughput Materials and Polymer Research. *Macromol. Rapid Commun.* **2004**, *25*, 48−58.

(5) LabVIEW Graphical Instrument Control. http://www.ni.com/ (accessed Dec 2005).

(6) Frantzen, A.; Sanders, D.; Scheidtmann, J.; Simon, U.; Maier, W. F. A Flexible Database for Combinatorial and High-Throughput Materials Science. *QSAR Comb. Sci.* **2005**, *24*, 22−28.

(7) Berman, F.; Fox, G.; Hey, T. *Grid Computing: Making the Global Infrastructure a Reality,* 1st ed.; J. Wiley: Chichester, U. K., 2003.

(8) W3C, Web Services Activity, 2002. http://www.w3.org/2002/ws (accessed Dec 2005).

(9) Evans, J. R. G.; Edirisinghe, M. J.; Coveney, P. V.; Eames, J. Combinatorial Searches of Inorganic Materials Using the Ink-Jet Printer: Science, Philosophy and Technology. *J. Eur. Ceram. Soc.* **2001**, *21*, 2291.

(10) Rossiny, J. C. H.; Fearn, S.; Kilner, J. A.; Zhang, Y.; Chen, L. Combinatorial Searching for Novel Mixed Conductors. *Appl. Surf. Sci.* **2005**, in press.

(11) Teng, W. D.; Edirisinghe, M. J.; Evans, J. R. G. Optimization of Dispersion and Viscosity of a Ceramic Jet Printing Ink. *J. Am. Ceram. Soc.* **1997**, *80*, 486−494.

(12) Teng, W. D.; Edirisinghe, M. J. Development of Ceramic Inks for Direct Continuous Jet Printing. *J. Am. Ceram. Soc.* **1998**, *81*, 1033−1036.

(13) Blazdell, P. F.; Evans, J. R. G.; Edirisinghe, M. J.; Shaw, P.; Binstead, M. J. The Computer-Aided Manufacture of Ceramics Using Multilayer Jet Printing. *J. Mater. Sci. Lett.* **1995**, *14*, 1562−1565.

(14) Internet Engineering Task Force, RFC 1014, XDR: External Data Representation Standard, 1987.

(15) Java Database Connectivity: JDBC Technology. http://java.sun.com/products/jdbc (accessed Dec 2005).

(16) OGSA-DAI: Open Grid Services Architecture Data Access and Integration. http://www.ogsadai.org.uk (accessed Dec 2005).

(17) W3C, Extensible Markup Language (XML) 1.0, Third Edition, 2004. http://www.w3.org/TR/REC-xml (accessed Dec 2005).

(18) Joy, B.; Steele, G.; Gosling, J.; Bracha, G. *The Java Language Specification*, 2nd ed.; Addison-Wesley: Reading, MA, 2000.

(19) ITU, ITU-T Recommendation X.509 (03/2000) ISO/IEC 9594-8:2001, 2001. http://www.itu.int/ITU-T/asn1/database/itu-t/x/x509/2000/ (accessed Dec 2005).

(20) MatML Schema Development Working Group, MathML Materials Markup Language, 2005. http://www.matml.org (accessed Dec 2005).

(21) Coveney, P. V. Scientific Grid Computing. *Philos. Trans. R. Soc. London, Ser. A* **2005**, *363*, 1705−2095.

(22) W3C, SOAP, version 1.2, Part 1: Messaging Framework. http://www.w3.org/TR/soap12-part1 (accessed Dec 2005).

(23) W3C, Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl (accessed Dec 2005).

(24) Langley, P. *Int. J. Hum.-Comput. Stud.* **2000**, *53*, 393−410.

(25) King, R. D.; Whelan, K. E.; Jones, F. M.; Reiser, P. G. K.; Bryant, C. H.; Muggleton, S. H.; Kell, D. B.; Oliver, S. G. Functional Genomic Hypothesis Generation and Experimentation by a Robot Scientist. *Nature* **2004**, *427*, 247−252.