Software Tools for Technology Transfer manuscript No.
(will be inserted by the editor)

Workflows for Quantitative Data Analysis in The Social Sciences

Kenneth J. Turner · Paul S. Lambert

21st March 2014

Abstract The background is given to how statistical analysis is used by quantitative social scientists. Developing statistical analyses requires substantial effort, yet there are important limitations in current practice. This has motivated the authors to create a more systematic and effective methodology with supporting tools. The approach to modelling quantitative data analysis in the social sciences is presented. Analysis scripts are treated abstractly as mathematical functions and concretely as web services. This allows individual scripts to be combined into high-level workflows. A comprehensive set of tools allows workflows to be defined, automatically validated and verified, and automatically implemented. The workflows expose opportunities for parallel execution, can define support for proper fault handling, and can be realised by non-technical users. Services, workflows and datasets can also be readily shared. The approach is illustrated with a realistic case study that analyses occupational position in relation to health.

Keywords e-Social Science · Quantitative Data Analysis · Scientific Workflow · Service-Oriented Architecture · Statistical Analysis

1 Introduction

This section introduces the process of statistical analysis as typically carried out in the social sciences, highlights the role of software tools in performing this, and explains the value of workflows as a means of enhancing the capabilities made available to social scientists for analysing quantitative data.

1.1 Need for Effective Analysis Tools

The statistical analysis of quantitative data constitutes a major activity in the social sciences (sociology, economics, psychology, etc.). Software is used as a matter of routine to store, retrieve and analyse numeric data such as the responses of different people to a questionnaire item. Software is also used to handle important metadata such as information about the content of a survey and the nature of its answer options.

There are, of course, many statistical packages. The goal of the work in this paper has been to allow social scientists to make more effective use of such packages. Statistical software is often complex and requires specialised technical knowledge that social scientists may not have. Social science researchers are also unlikely to be familiar with advanced techniques such as scientific workflows. Effective tool support is thus needed to let social scientists take advantage of quantitative data analyses defined by themselves and others. The work reported here focuses on supporting tools that serve as a bridge between computer science and the social sciences. In this way, transfer is achieved of advanced computing technologies into mainstream social science.

Computing Science and Mathematics, University of Stirling, Scotland FK9 4LA, UK E-mail: kjt@cs.stir.ac.uk · Applied Social Science, University of Stirling, Stirling FK9 4LA, UK E-mail: paul.lambert@stir.ac.uk

Although the paper uses quantitative data analysis in the social sciences as its illustration, the approach is general. The tools are equally applicable to statistical analysis in other disciplines. Even more generally the tools can be used to support scientific workflows in many fields, where the component services perform any computational task.

1.2 Background to Statistical Analysis in The Social Sciences

Science is said to have evolved through four paradigms: empirical, theoretical, computational, and data-intensive [17]. The social sciences are regarded as being 'data rich', with vast amounts of useful quantitative data available for analysis [29]. Nevertheless, it is perceived that the social sciences have not exploited the available resources to their full potential due to difficulties in organising and analysing relatively complex information resources [25]. Examples of influential types of quantitative data within the social sciences include:

Social Surveys: A wide range of social survey datasets are used. Some involve sampling small numbers of respondents on specialist topics, but many of the most influential analyses make use of large-scale secondary survey data. This is typically accessed online through a data centre (e.g. the Research Information Network, *www.rin.ac.uk*). An illustrative example used in this article is the BHPS (British Household Panel Survey, *www.iser.essex.ac.uk/bhps*), a longitudinal survey of individuals living within British households. The BHPS originally sampled 10,264 adults within 5,511 households in 1991. It has aimed to re-interview these people every year subsequently, in addition to some new respondents. The BHPS is a general-purpose survey and features extensive questions about socio-economic aspects, attitudes, values, lifestyles and subjective well-being. A second example, also used in this article, is the SHS (Scottish Health Survey, *www.scotland.gov.uk/Topics/Statistics/Browse/Health/scottish-health-survey*). This is a periodic survey instituted in 1995, and repeated (with different respondents) in 1998, 2003, and from 2008 annually. The SHS focus is on health outcomes, and features both subjective and objective information about respondents, along with background information on socio-economic and demographic circumstances.

By-Product or Administrative Data: High-volume quantitative data of potential relevance to social scientists is generated as a by-product of administrative activities. Examples include government tax records, health records, and data collected through educational institutions. Common characteristics of by-product data are its large volume, challenging formats (e.g. with errors and inconsistencies), and controls over security that often lead to stringently controlled access arrangements [3]. Several agencies currently make efforts to support social science research using by-product data, e.g. the National Centre for Research Methods Admin project (*www.ioe.ac.uk/research/16083.html*) and the Administrative Data Liaison Service (*www.adls.ac.uk*). The restrictive conditions on much by-product data make careful organisation and preparation of data analysis procedures especially important.

Aggregate Level or Macro Data: This kind of data refers to collections of statistical summaries derived from a secondary source. In the UK, the Economic and Social Data Service (*www.esds.ac.uk/international/about*) curates an influential collection of aggregate data and supports researchers in analysing this. Macro level data can deal with many topics but is particularly likely to involve socio-economic profiles (e.g. unemployment statistics for a region). The datasets used are often relatively small in size by comparison to other resources. One common use of aggregate data, aside from its individual statistical analysis, is merging it with another quantitative dataset to add some contextual data to the latter resource [10]. This often supplements the main process of analysing a social survey or administrative dataset.

Statistical software packages are typically used interactively, either through a GUI or by processing short segments of scripts in some command language. The latter approach is widely regarded as the more satisfactory method of undertaking analysis [33]. The extended sequence of software commands is stored in a text file usually known as a 'syntax file'. The commands support functions to access and process datasets such as loading and saving files, or adapting relevant metadata and the data itself (e.g. recoding or transforming variables). The commands also typically support a wide range of statistical operations on data such as generating summary statistics, estimating statistical models, and generating graphs. The datasets used in research can be large (tens of megabytes to gigabytes) and the complete set of statistical scripts within syntax files can be complex (hundreds to thousands of lines of code).

There is a long history to the development of statistical software packages for social researchers. General-purpose software packages such as SPSS (originally Statistical Package for the Social Sciences, *www-01.ibm.com/software/analytics/spss*) and Stata (*www.stata.com*) provide easy-to-use facilities through proprietary packages. Numerous other general-purpose and specialist packages are available on a proprietary basis. Among open-source packages, R (R Project for Statistical Computing, *www.r-project.org*) is well-known. Statistical packages normally treat datasets as tables, with columns identifying variables (such as a measure of income or age) and rows giving particular values of these for each case in the dataset.

Most social scientists are trained to use statistical packages for commonly required statistical analyses, following instructions given in textbooks (e.g. [33]) or popular websites (e.g. *www.ats.ucla.edu/stat*). Although statistical software scripting languages have many elements of general programming such as macros, sequencing, iteration and alternatives, the users of these packages are not ordinarily formally trained in these elements. This results in a wide variation among day-to-day practices. Moreover, the packages themselves often lack more sophisticated programming features such as event handling, exception handling, compensation handling (recovery from errors) and support for parallel execution.

1.3 Motivation for A New Approach

Social scientists usually develop their own statistical scripts to analyse quantitative data. This is commonly done in a long and complex manner involving the sequential execution of a series of different data preparation and data analysis activities. The statistical scripts produced are often a vital part of the research, and require substantial effort. Although most social scientists do recognise the desirability of developing their scripts in a well-organised and documented manner (using advice such as [22]), it is not common practice for scripts to be clearly organised, wholly replicable or shared with others. Accordingly there are a number of apparent limitations in the statistical analysis of quantitative datasets within the social sciences:

- Many social scientists do not have a comprehensive knowledge of the statistical packages that they use, so they may not develop scripts that take full advantage of available facilities. More readily available building blocks (i.e. scripts) could allow better exploitation of data resources.
- Statistical analysis scripts are often developed in an *ad hoc*, one-off manner, with limited reuse. For example, researchers typically copy and adapt segments of their own scripts from earlier projects in future work, but in an application-specific manner. It is uncommon for researchers to develop scripts specifically for reuse.
- Relatively few sources provide common scripts for use by the wider community. Those which do exist tend to focus on methodologically innovative and advanced facilities (e.g. the extension programmes published in *The Stata Journal* and *The R Journal*), or focus on specialist data resources concerned with particular measures and their qualities (e.g. *www.geode.stir.ac.uk* and *www.methodbox.org*). However the majority of day-to-day scripts are never disseminated.
- Documentation about scripts is often limited, typically having minimal comments for the benefit of the original author alone. Many scripts are complex so it is hard to gain an understanding of how they work simply from inspection. A comprehensible way of presenting the features of statistical analysis scripts would offer a much better means for understanding scripting jobs.
- The higher-level organisation of scripts into coherent tasks is rarely considered. Many social scientists use the term workflow to describe the detailed sequence of commands within one particular statistical package [22]. This might however be better termed a 'micro-flow', in comparison to higher-level 'macro-flows' which link individual blocks of analysis and activity. Although the benefits of higher-level workflows have gained acceptance in some applied research areas (e.g. bioinformatics), such workflows are not generally recognised for statistical analysis in the social sciences.
- Batch processing of scripts is relatively rare in the social sciences. A few data access arrangements require this, but most researchers are used to having a local copy of the dataset and to working through groups of commands in an interactive manner. As a result, scripts are not normally broken up into robust separable components, and are not checked for overall coherence – neither of these aspects is a strong requirement when working in an interactive style.
- Many social science datasets are large and their analysis can be computationally intensive: some jobs are highly time-consuming, taking hours or days to complete. Moreover, the tendency to combine a series of

tasks into a single sequence means that researchers often unnecessarily re-run segments of analysis within long scripts. When the analysis embodies separable calculations, it can save substantial time to perform these in parallel using multiple computers.
- Although the outcomes of quantitative data analysis can be important, there is little software support for social scientists to check the accuracy of their scripts.

1.4 Objectives of The Work

Transparency and repeatability are central tenets of a scientific approach, yet the limitations noted in section 1.3 mean that neither is widespread within quantitative social science. Historically there have been efforts to make datasets available to others, but similar attention has not been given to the analytical scripts which underlie most research results [14]. Many of the limitations described in section 1.3 act against transparency and repeatability in the use of software for statistical analysis. These limitations have been recognised in software engineering, and techniques to address them have been created.

This article describes an approach to developing quantitative data analyses using best practice from scientific workflows. In particular, the objectives of the work have been:

- to develop general techniques and tools for workflows involving external programs as components, but to demonstrate the approach on statistical analysis scripts
- to ease the task of creating new statistical analyses using existing scripts
- to encourage reuse of statistical analysis scripts by describing their flows at a high level, and by making their building blocks available to the community via workflow servers
- to support description and curation of statistical analysis scripts as workflows so that they can be used by others to repeat the analysis procedures
- to adapt the techniques of SOA (Service-Oriented Architecture) for statistical analysis scripts so as to gain benefits such as loose coupling of components and remote access
- to adapt the techniques of HPC (High-Performance Computing) for workflows using statistical analysis scripts, allowing concurrent execution via a processor pool
- to adapt the techniques of formal methods for statistical workflows, allowing the completeness and correctness of workflows to be verified and validated.

1.5 Overview of The Article

Section 2 describes related work by others in the fields of e-social science and computational workflows. Section 3 explains the approach to modelling statistical analyses as workflows and the subsidiary scripts that act as their building blocks. Section 4 introduces the notation used to describe statistical analysis workflows in a graphical fashion that non-specialists can easily grasp. A case study illustrating the approach appears in section 5, namely an analysis of the relationship between occupational position and health. Section 6 summarises the key points of the work, discusses its contributions, and gives pointers to future developments.

2 Related Work

This section presents related work by others on e-social science and computational workflows. Some general techniques can be adapted to support workflows for quantitative data analysis.

2.1 e-Social Science

During the past decade, grid computing (e.g. [13]) has been heavily developed to provide networked computational facilities for large-scale scientific computing. myGrid (*www.mygrid.org.uk*) is an overarching framework for tools development in e-science generally. This includes several of the tools mentioned later.

Much of the focus on grid computing has now moved onto work on cloud computing (e.g. [2]) as a way of providing networked computational facilities through third parties. Although cloud computing is relevant to scientific research, it is more generally focused on computing in general (including commercial services). The emphasis on the grid has also been replaced by a focus on e-infrastructure as a way of supporting e-science. That is, the focus is now on the networked capabilities (storage, computation and communication) needed for data-intensive science.

Work on e-social science in the UK was supported by the Economic and Social Research Council through the National Centre for e-Social Science and the Digital Social Research programme (*www.digitalsocialresearch.net*). Research funded under this theme included the DAMES project that provided the context for the work in this article.

DAMES (Data Management through E-Social Science, *www.dames.org.uk*) was a collaboration between social scientists and computer scientists to develop an e-infrastructure in specific areas. In particular, DAMES developed portals that support the collection, description and dissemination of data resources on ethnicity and immigration, occupation, and educational qualifications. Additional portals support the analysis of data on social care and on mental health.

The DAMES portals were designed for use by social scientists. They are therefore intended to support social science research without the computing aspects getting in the way. For example, the collection of metadata is vital when distributing data resources, yet social scientists are unlikely to be familiar with standards such as DDI (Data Documentation Initiative) which can be used for this purpose [4]. DAMES therefore offers a user-friendly wizard that allows researchers to curate their datasets with automatic creation of metadata. The datasets described can be archived on a DAMES portal, or may be stored remotely with just the metadata being held on the portal.

Typically the DAMES portals are used by researchers to search for data resources that match the required content (e.g. an occupational classification scheme). Resources can also be ranked by users to indicate their likely usefulness. These resources include numeric data, but also include records of procedures such as statistical analysis scripts or workflows. The portals are therefore a valuable resource for researchers, and they increase in value as more data and procedures are added. Indeed, two cognate projects have developed similar models for collating and distributing relevant information resources: Methodbox (*www.methodbox.org*) has focused on data resources including analytical scripts concerned with health surveys, and PADLS (Portal of the Administrative Data Liaison Service, *http://www.adls.ac.uk/padls*) has focused on the distribution of scripts and related data files for research specifically concerned with use of administrative data.

Several other projects have supported quantitative data analysis in the social sciences. As two recent examples, NeISS (National e-Infrastructure for Social Simulation, *www.neiss.org.uk*) and e-STAT (*www.bristol.ac.uk/cmm/research/estat*) are mentioned here. NeISS has studied techniques and tools to support computational simulation of socially relevant problems such as city traffic and disease spread. This has led to the construction of portals in which pools of simulation code are made available. This code can be invoked through an online interface with suitable parameters to perform computationally challenging analytical tasks on remotely stored datasets. e-STAT has focused on interoperability of software applications for advanced elements of statistical analysis (such as advanced statistical models). This project has created a software invocation engine, Stat-JR, which uses templates of pre-prepared analytical scripts in the languages of statistical analysis packages. These scripts can be applied in response to requests for specified data with given parameters [6]. e-STAT has also developed a form of electronic book which allows users to document statistical models as applied to their data in a dynamic context [6]. The developments of NeISS and e-STAT are limited, however, to their particular specialist domains: social simulation analysis and statistical modelling respectively.

2.2 Computational Workflows

Workflows originated in management science as a way of describing the flow of work through an organisation. Although workflows were originally devised for business purposes, they were subsequently adapted for scientific applications. Scientific workflows (e.g. [28, 32]) have therefore become an established part of e-science.

Computational workflows use specialised languages such as BPEL (Business Process Execution Logic [1]). Although a number of other workflow languages exist, BPEL is the main standard in this area and is widely supported by industrial and research organisations as well as tool developers. There is also extensive documentation

and training material on BPEL. This meant that the work reported here could build on established techniques and tools, rather than having to create these from scratch. BPEL has therefore been the language of choice for the work described in this article.

BPEL supports what is called service orchestration: combining the execution of different components. (From a more formal perspective, the term 'service composition' is also used.) BPEL is designed specifically for orchestrating web services: networked components that offer their facilities through standards like WSDL (Web Services Description Language) and SOAP (originally Simple Object Access Protocol). Web services offer significant advantage such as loose coupling of components, platform independence and implementation language freedom.

Discovery Net [16] was one of the first approaches to scientific workflows. Originally focused on applications such as the life sciences, it evolved to support applications in other areas such as bioinformatics and health informatics. Workflows are described visually with a separation of data and control flows. For the work in this article, the authors favoured the use of the BPEL standard as a basis. Support for formal analysis is also feature of the work reported here.

Taverna [26] was developed to model web service workflows – specifically for bioinformatics. It introduced SCUFL (Simple Conceptual Unified Flow Language) to model grid applications in a specialised workflow language. Early work was undertaken on the NeISS project mentioned in section 2.1 to use Taverna for workflows describing social simulation analysis. The approach proved relatively intensive technically. Instead a bespoke portal linking the relevant simulation tasks proved to be a more attractive solution for the social scientists involved (see *drupals.humanities.manchester.ac.uk/neiss3/tools*). Again, the work in this article differs in focusing on BPEL as its standardised foundation and in providing support for formal analysis.

jABC (Java Application Building Center [30]) is a flexible approach for building complex software applications. jABC uses Lightweight Process Coordination to realise applications, building on the concept of Service Independent Building Blocks as the component elements. Model checking can be undertaken using temporal logic properties. Bio-jETI (Bioinformatics Java Electronic Tool Integration [24]) makes use of jABC to create bioinformatics workflows that are designed graphically and can be verified through model checking. The work in this article shares a common interest in formal methods, though the work also supports formally based validation in addition to formal verification.

JOpera [27] is a service composition tool for building new services by combining existing ones. It provides a visual composition language and also a run-time platform to execute services. JOpera claims to offer greater flexibility and expressivity than BPEL. Although the advanced approach of jOpera is attractive, the authors value the benefits that come from using BPEL as a standardised workflow approach. Even though BPEL is used indirectly in the work reported here, the concepts are similar for anyone who is familiar with BPEL.

The SENSORIA project (Software Engineering for Service-Oriented Overlay Computers) has studied a number of aspects of service design, including larger case studies using web service orchestration. UML (Unified Modeling Language) is used to describe service structure, evolution and activities. A number of process calculi were created for modelling web services [39]. These are coupled with techniques for functional or performance analysis. UML has also been been used in [19] to describe web services at a high level, e.g. using activity diagrams or state diagrams. These are translated into Finite State Processes for model checking to find deadlocks or problems with synchronisation. Although the use of UML agrees with the authors' preference for standards, the above formalisms for analysis are not standardised. This article also tries to support a methodology, techniques and compact toolset that do not require specialised knowledge.

OMII-BPEL (Open Middleware Infrastructure Institute BPEL [38]) aims to support the orchestration of scientific workflows with a multitude of service processes and long-duration process executions. It provides a customised BPEL engine, and supports a set of constructs desirable for specification of scientific workflows. This work resembles the approach taken in this article in that the same workflow language and underlying workflow engine are used. However, OMII-BPEL is for workflows in general and has not been specialised for social science. OMII-BPEL also lacks key features of the work described here: graphical description of statistical workflows, automated formal analysis, and automated testing.

The myExperiment project (*www.myexperiment.org* [9]) has developed techniques for sharing workflows among scientists in a manner reminiscent of social networking. myExperiment has been well integrated with Taverna as a means of sharing bioinformatics workflows. Other kinds of workflows can also be used, though they are not so well integrated.

2.3 Formal Methods for Workflows

Formal methods are widely used in computer science to create and analyse mathematical models of systems and software. In particular, they have been used to specify and analyse workflows. The approaches of SENSORIA, jABC and Bio-jETI were mentioned above.

LTSA-WS (Labelled Transition System Analyser for Web Services [12]) is a finite state approach to specifying web services. Abstract service scenarios and actual service implementations are generated through behavioural models in the form of state transition systems. Verification and validation are performed by comparing the two systems. The approach is restricted in its handling of data types. The work in this article differs in generating the formal model and the implementation from a single abstract description, and in allowing arbitrary data types.

PROPOLS (Property Specification Pattern Ontology Language for Service Composition [40]) is a pattern-based specification language for temporal business rules. A behavioural model combines rules using their respective finite state automata. The process model can then, in principle, be transformed into BPEL. The approach does not, however, deal with data types. The work reported here generates both the specification and the implementation from the same description, dealing fully with data.

WSAT (Web Services Analysis Tool [15]) is used to analyse and verify composite web services, particularly focusing on asynchronous communication. Specifications are translated into Promela and model checked using SPIN. WSAT is able to verify synchronisability and realisability. However, the tool does not support the full range of capabilities found in BPEL (e.g. error handling and compensation).

StAC (Structured Activity Compensation [7]) is a process algebra that has been used to specify the orchestration of long-running transactions. This can be used with the B notation to allow specification of data types. Most of BPEL can be translated into StaC, but the emphasis is on reasoning about transactions rather than support for BPEL. [21] also focuses on verifying web transactions, but is even further from BPEL.

As an important standard for formal methods, LOTOS (Language Of Temporal Ordering Specification [18]) is able to describe both the behaviour of a system and the data it operates on. As for BPEL, the authors favour the use of standards and have therefore used LOTOS in the work of this article.

[8, 11] are the closest to the formal aspects of the work reported here. These approaches automate the translation between BPEL and LOTOS. The work in this article differs in that no specification is required of either BPEL or LOTOS. Instead a graphical notation, accessible to the non-specialist, supports abstract service descriptions that are translated into BPEL and LOTOS automatically. This is an advantage as social scientists are very unlikely to be familiar with either BPEL or LOTOS.

3 Modelling Quantitative Data Analysis

This section explains the approach to modelling quantitative data analysis as performed in the social sciences. It will be seen that workflows call services that wrap analysis scripts in a service-oriented way.

3.1 Statistical Analysis Scripts as Functions

Like nearly all programs, statistical analysis scripts are deterministic (i.e. they produce repeatable results). A script can therefore be considered as a function that takes parameters and produces results. Most of the work of a script involves reading datasets from files, processing these, and generating new datasets or results. The parameters of a script can be strings (including file names), numeric values and booleans. The results (such as new data, graphs, or statistical results) are represented by their file names.

However, social science practice is often loose about how scripts are written. One problem is that script inputs are often implicit rather than being proper parameters. For example a script may implicitly read files that are not identified in its call, and it may rely on environment values such as global macros. This makes it difficult to assess what a script does without reading it in detail and understanding its context. It also mitigates against reuse of a script (with different parameters or by others). Many social scientists are aware of reusability challenges and seek to minimise the use of implicit parameters [22]. However these are rarely eliminated, perhaps as a result of the complex nature of many data resources and files. The approach taken in this article encourages the script

developer to use explicit parameters. This is not mandated, however, and implicit parameters can still be used. A legitimate use of implicit parameters is where these can be regarded as constants such as literal values or fixed external datasets. An example of the latter would be an aggregate level dataset on occupational codes stored online, which might be linked with another dataset on the basis of information about shared occupational codes (see section 1.2).

A different problem arises concerning the outputs of a script. Without reading a script in detail it can be hard to know what files it creates or modifies, so it may be difficult (or even impossible) to determine what its precise outputs are. This makes script reuse problematic. The approach of this article therefore encourages the script developer to be explicit about outputs. Again, this is not mandatory but is strongly encouraged unless the outputs do not play a role in further analysis: an example would be the log file from a script. In social science practice, key information is often placed into the log file (e.g. a description or summary of important variables). This can legitimately be an implicit output from the script, but is nonetheless useful to the researcher.

Although the focus of this article is on statistical analysis scripts and workflows, in fact the approach is completely general. The scripts can be regarded as programs (calling external statistical packages) that are used by the workflow. The approach is therefore easily generalised to calling any external programs. Provided these can be treated in a functional way (which is true of many programs) then they can be incorporated into a workflow.

3.2 Statistical Analysis Scripts as Services

The functional view of statistical analysis scripts corresponds naturally to services in SOA (Service-Oriented Architecture). Scripts can therefore be regarded as black-box services that perform tasks for clients. Like a script, a service takes input parameters and produces output results.

Web services are the most widespread realisation of SOA. Script parameters and results correspond to operation parameters and return values. Like a script, a web service can take zero or more parameters – but it can return only zero or one results. If several results are needed (e.g. multiple output files are generated), these must be stored in a single structured return value.

It is necessary to consider the faults that can result from service execution. In fact, statistical analysis scripts in software applications are often weak in dealing with errors: typically the script stops with an error in the log, and there may be no other indication as to why processing failed. Web services are explicit about failures, reporting them as faults to the client. The tools described later treat a non-zero exit code from an external program, or output from this to standard error, as an exception and wrap this up as a web service fault.

3.3 Statistical Analysis Scripts in Workflows

A computational workflow defines the logic that relates calls of individual services. For quantitative data analysis, statistical analysis scripts are the building blocks of workflows and correspond to services. A simple linear flow might be sufficient, but there can also be iteration or alternative paths. More complex flows allow for parallel execution, event handling, compensation handling and fault handling. A nice feature of computational workflows is that they are services in their own right. That is, a workflow can call other workflows as well as atomic services. This allows complex workflows to be decomposed hierarchically into simpler workflows and ultimately into individual services.

An interesting question concerns the appropriate level of granularity for the scripts called by statistical analysis workflows. There are two extremes: individual software commands as workflow activities, and entire scripts as workflows.

Making individual commands visible in a workflow is obviously very low-level. However, it does suggest an intriguing possibility: trying to devise a workflow language that can be mapped to different software scripting languages. With the plethora of statistical packages, researchers are often obliged to use just one package and cannot easily reuse scripts written for another package. A low-level statistical analysis workflow could, in principle, be shared among researchers and interpreted for different packages. However, this is a challenging goal that is difficult to realise. The e-STAT project mentioned in section 2.1 used elements of this approach. However, this was found to be tractable only by packaging scripts as limited bundles of commands associated with specific statistical models. Moreover, although analysis scripts can make use of iteration and alternation, in practice they

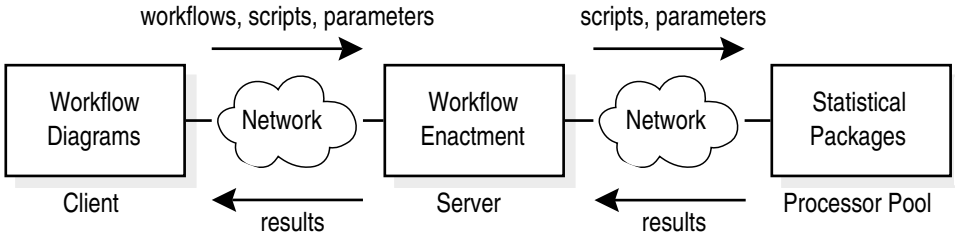

Fig. 1 Workflow Execution

are generally linear. As a result, workflows at this level would usually be little more than lengthy sequences. Unsurprisingly, this kind of low-level workflow is not an attractive proposition.

Equally, treating an entire script as one workflow is at too high a level. Indeed, it would negate the point of having a workflow at all. There would also be little opportunity to reuse such scripts in different analyses since the extended script would be highly specific to the application area.

An appropriate level of granularity is therefore somewhere between these two extremes. Statistical analysis scripts are often written as monolithic elements ('master files' that call on further scripts as 'subroutines'). However there is often an opportunity to break scripts up into smaller components, as will be seen in the case study of section 5. This makes the individual scripts more modular, reusable and comprehensible. It also allows opportunities for parallel execution to be discovered, and reduces unnecessary replication of computationally intensive tasks within an extended script. The authors consider that, in practice, scripts with 5 to 20 commands are typically a good size in terms of reusability and comprehension.

3.4 Using Workflows

Workflow execution is shown in figure 1. The social science researcher prepares workflow diagrams, scripts and datasets on the client system (typically a desktop PC). If desired, a workflow can first be formally validated and verified before committing to implementation and a possibly lengthy execution. When the researcher is confident in the workflow, it is submitted to the server for execution. Individual scripts and their parameters are sent to the processor pool for execution. Results are passed back, initially to the workflow engine, then as required to the client system.

The client, server and processor pool can correspond to the same physical system. More typically, these are all different physical systems linked via a network. For example the DAMES server and its processor pool are part of the Computing Science network at the University of Stirling. Client systems are spread and access the server via the Internet. Workflow execution has been designed to minimise the volume of file transfer. Unless a file changes, it is transferred only once to another system; it is also compressed during transfer.

Although specialised datasets to be analysed are often stored on the local client system, more general datasets can be stored anywhere that is network accessible. A benefit of the DAMES approach is that datasets can be curated and stored on the workflow server for use by other researchers. Datasets created by other research projects are often stored in public repositories. Yet other datasets may be described on the server but be located on an individual researcher's system. The workflow support allows for all of these cases.

By default datasets are named relative to the workflow, for example *job/soc90_camsis.dta* would refer to a Stata data file inside the workflow's *job* directory. However, absolute file names can also be used (e.g. */data/survey/bhps1995.dta* or *C:\data\shs1995.dta*). Since file names within a workflow are relative to its directory, file names are exported from one workflow to another in absolute path form.

If a dataset is located on a remote server, it is identified by a URI (Uniform Resource Indicator, normally a URL). As a convenience, a special form of URI is used for datasets curated on the DAMES server (e.g. *dames:/psl/camsis/soc90.dta*). Some datasets have restricted access that require HTTPS (HyperText Transfer Protocol Secure). Unfortunately support for HTTPS by statistical analysis packages is patchy. In such a case, a temporary local copy of the dataset is automatically downloaded for use by a workflow. The client system does not need to have the relevant statistical packages, since these are required only on the server side. It is also

possible for a researcher to use a workflow to achieve interoperability between software packages. For example, this might allow use of a script written in R (say) when the researcher's own expertise is restricted to Stata. This offers a valuable service to social scientists since lack of transferability between statistical packages is a frequent blockage in research activities. For commercial packages there may be licensing limitations on the possibilities that are allowed, but at the minimum the system can support execution of tasks using packages that researchers themselves can access.

Once a script has been been converted into the form of a service, it can be made available to any other user of the server. This strongly encourages reuse. For example, a script for data fusion might be developed for one workflow. It can then be made available to other workflows (whether defined by the same researcher or not). The workflows themselves are services and can therefore be reused in other workflows. The tools create services and workflows in per-user 'sandboxes' so they do not interfere, but can be made available to other users in a controlled manner.

Making services available on the server is potentially a security risk. The whole DAMES server is therefore properly protected against unauthorised usage. Services that might be used from outside the server (such as a workflow) are protected by HTTPS. This ensures that only authorised users can access services and therefore any datasets they employ.

Dissemination of statistical analysis scripts as services offers exciting opportunities for improving the use of software in quantitative analysis. Creating complex scripts from scratch can be challenging, and may well be inefficient if others have already undertaken similar exercises. Once deposited on the workflow server, scripts can be searched and ranked by other researchers, leading to cumulative improvements in script resources.

4 Statistical Analysis Workflows with CRESS

This section introduces the CRESS notation and toolset used in this work, with particular reference to statistical analysis workflows. The complete workflow development methodology is explained, though only certain aspects of this need be used. The diagrammatic notation used for statistical analysis workflows is also described.

4.1 The CRESS Notation and Toolset

Statistical analysis workflows are supported by a diagrammatic notation and supporting toolset called CRESS (Communication Representation Employing Systematic Specification, *www.cs.stir.ac.uk/cress*). This offers a general approach for describing services of many kinds. It has been applied to the Intelligent Network, Internet telephony, Interactive Voice Response, Web Services, Grid Services and Device Services (home automation). The work reported here is a new application of CRESS.

A usability assessment of the CRESS workflow notation has been carried out with a number of developers [36]. This evaluation provided CRESS training materials that developers were able to absorb in an average of 41 minutes. This was followed by a series of five exercises that developers conducted unaided. The developers were able to use CRESS with 90% accuracy, taking an average of 16 minutes to complete all the exercises.

The CRESS service notation is graphical for easy comprehension by non-technical users. The toolset is highly automated, and requires little more than creating service diagrams to obtain automatic analysis and implementation of services. CRESS supports several front-end diagram editors, though CHIVE (CRESS Home-grown Interactive Visual Editor, *www.cs.stir.ac.uk/~kjt/software/graph/chive.html*) is preferred as it is well integrated with the toolset. CRESS supports several back-end compilers that translate diagrams into various languages such as BPEL (for implementation) and LOTOS (for specification).

The CRESS toolset is portable across multiple platforms, being written in Perl and Java. CRESS is accompanied by a number of supporting tools for service analysis. Formal properties of a service can be verified through model checking with CLOVE (CRESS Language-Oriented Verification Environment, *www.cs.stir.ac.uk/~kjt/research/clove.html*). A service can be formally validated using MUSTARD (Multiple-Use Scenario Test And Refusal Description, *www.cs.stir.ac.uk/~kjt/research/mustard.html*). The implementation of a service can be rigorously tested and analysed for performance bottlenecks with MINT (MUSTARD Interpreter, *www.cs.stir.ac.uk/~kjt/research/mint.html*).

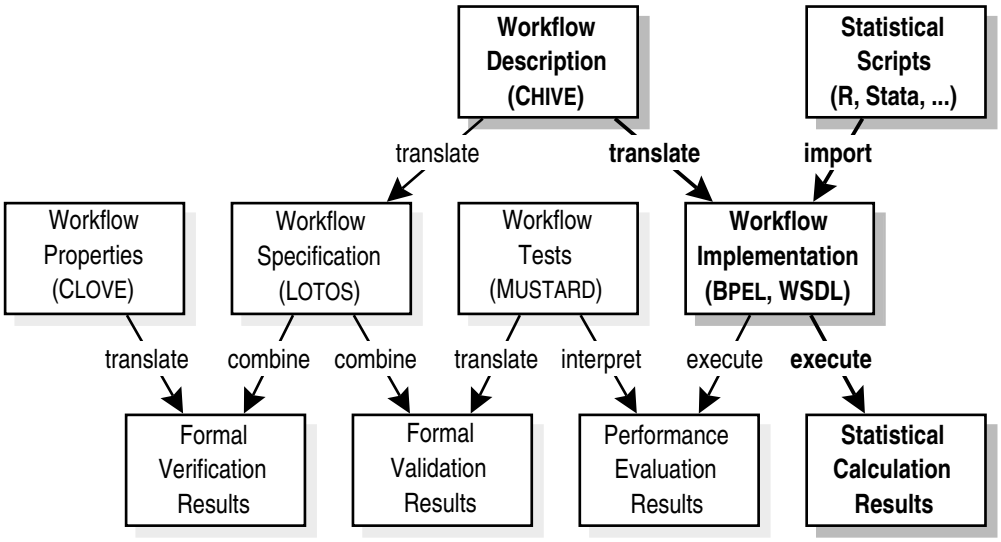

Fig. 2 Workflow Development Methodology

4.2 Workflow Development Methodology

Besides the notation and toolset, CRESS also offers a comprehensive methodology for service development. Figure 2 shows the full methodology as it applies to statistical analysis workflows. In fact only the part shown in bold to the right is essential – the other activities are optional although they contribute to confidence in workflow design. Formal aspects of the methodology are not covered in this article but can be followed up through the given citations.

The primary activity is creating a workflow description, usually with the CHIVE diagram editor. A workflow diagram is self-contained and provides both data and behavioural definitions. Large workflows can be multi-page if necessary, and can use connectors between different parts of a complex service.

If accuracy of the workflow is critical, it is beneficial to formally verify it using CLOVE for completeness and correctness. This is achieved by proving that the workflow specification respects key properties [37]. Some properties are generic, for example the workflow should not deadlock (i.e. reach a point where it can make no further progress) and should not livelock (i.e. become stuck in an unproductive internal loop). These properties are automatically checked without any effort from the workflow developer.

Specialised properties can be defined for safety and liveness, e.g. that a request will not generate an inappropriate response and will always result in an output. Application-specific properties can also be defined using the high-level CLOVE notation that is internally translated into modal μ-calculus [5]. Although this is useful for complex workflows in other applications, it is not expected that analysis of quantitative data in the social sciences will require such properties.

Formal verification can be computationally and intellectually challenging. As a pragmatic alternative, MUSTARD can be used to formally validate a workflow description. This involves mathematically-based testing of the workflow specification. Like other forms of testing it is necessarily limited, but formal validation can achieve useful results when formal verification is impracticable [35]. Some effort is required from the workflow developer to define MUSTARD tests. However, there are several gains from this: the workflow description can be validated before any attempt is made to implement it, the workflow implementation can be tested for functional correctness, and the workflow performance can be checked. If the workflow or the scripts it calls change in future, regression testing is automatic using the originally defined tests. All these activities are enabled by writing one set of tests, so the modest effort is amply repaid. Although users are not obliged to do this, they are encouraged to do so.

The main route through the methodology involves defining the high-level workflow and the scripts that it calls. In many cases, these scripts will already exist and just need small alterations to fit into a workflow model (e.g. avoiding implicit inputs, and reporting failures suitably). CRESS takes the workflow description and scripts,

automatically creating an implementation using BPEL for the workflow and WSDL for the service descriptions. Many auxiliary files are also created automatically to support the workflow. Workflows are executed by the ACTIVEBPEL workflow engine (*www.sourceforge.net/projects/activebpel502*). Scripts are executed by calling the appropriate statistical package.

Workflow execution is coordinated in one place (the server), but individual activities in the workflow (the scripts) can be executed in a distributed way. In fact there are three options for executing workflows and scripts: on the user's local computer, on a specified remote server, or within a specified Condor pool. Condor [31] is a distributed job scheduler that can be used to execute tasks within a pool of computers. Particularly where opportunities for parallelism can be identified within a workflow, this can lead to significant speed-ups in calculation times. When scripts are executed by the processor pool, key parts of the file system are shared using NFS (Network File System) so that it is not necessary to copy files to processors in the pool.

The client installation needed to use workflows is minimal, requiring just a workflow editor. The CHIVE editor is written in Java and so is portable. If the client system does not yet have a JRE (Java Run-time Environment), this is readily downloaded. CHIVE can also be provided pre-packaged with the required Java subset using Excelsior JET (*www.excelsior-usa.com/jet.html*): this offers a standard installation package that may be more convenient for less technical users.

CHIVE is well integrated with CRESS. It can call the CRESS tools on the local system if they are installed there, but can also call CRESS on a remote server. Using the CRESS tools requires simple menu choices in CHIVE: Check (workflow syntax check), Deploy (create a workflow specification or implementation on the server), Validate (formally test a workflow specification), Verify (formally prove properties of a workflow specification), Invoke (call a workflow with specified parameters) and Undeploy (remove the workflow files from the server).

4.3 CRESS Diagrams for Statistical Analysis Workflows

A CRESS diagram is a directed graph that shows the flow of activities. Numbered diagram nodes define inputs and outputs (communication with other services) or actions (internal to the service). In a workflow, an activity can terminate successfully or can fail (due to a fault).

Arcs in a CRESS diagram link activities. These can be qualified by a condition such as an event that must happen or an expression that must hold for the arc to be followed. Assignments can also be placed on an arc. Branches in a CRESS diagram normally reflect alternatives, but parallel paths can also be defined with **Fork** and **Join**.

CRESS supports a wider range of constructs than is described here. The subset of CRESS activities appearing in this article is explained in table 1, with concrete examples appearing in figures 3 and 4. For statistical workflows, some constructs are interpreted in a specific way. For example, a web service is invoked as *service.port.operation*:

- As an example for a workflow, consider the service invocation *job_health.dames.combine* in figure 3. The service name is that of the workflow (e.g. *job_health*). The port name is developer-defined, but by convention is named after the workflow server (e.g. *dames*). The operation name is developer-defined (e.g. *combine*).
- As an example for a script, consider the service invocation *health.stata.get_health* in figure 3. The service name is developer-defined but is usually related to the script (e.g. *health* for a *get_health* script). The port indicates which statistics package is used to execute the script (e.g. *Stata*). The operation name corresponds to the script (e.g. a *get_health* operation for a Stata script uses the command file *get_health.do*).

Explicit faults can be listed for all script invocations. However since any script can fail, an implicit fault is assumed if none is given (e.g. *healthError.reason* in the case of the *health* service, giving some *reason* text explaining the failure).

Rounded rectangles called rule boxes are used to define data types, variables and subsidiary workflows (which the current workflow depends on). A rule box has the form:

Uses *declaration** (/ *workflow*+)?

If a workflow makes use of subsidiary workflows, these are given following '/' (e.g. *JOB_HEALTH* in figure 4). Such workflows are automatically included along with the main workflow.

Declarations give a type followed by variables conforming to the type. Examples of simple types are **String** and **Float**. An example of a complex (structured) variable is:

Construct	Meaning
(*workflow*:)?*name*	a variable or fault name defined by a particular workflow (the current one if no workflow name is given as a prefix)
service.port.operation	an operation for the given service and port
name(*.variable*)? | *.variable*	a fault name with optional variable or just a fault variable
/ variable <– value	an assignment associated with an arc or node
Catch *fault*	how to handle a fault; a fault unmatched in the current scope is sought in higher-level scopes
Compensate	used after a fault to undo previous work by calling compensation handlers in reverse order of activities
Compensation	undoes work due to a fault; enabled once the corresponding activity completes successfully
Fork	introduces parallel paths; may be nested to any depth
Invoke *operation output* (*input fault**)?	one-way for output, or two-way for output and input; potential faults are listed statically but happen dynamically
Join *condition*	matches a **Fork**; an explicit join condition refers to successful termination of prior numbered activities, e.g. '3 && 5'
Receive *operation input*	an initial **Receive** creates a new workflow instance; matched by a **Reply** for the same operation
Reply *operation* (*output* | *fault*)	typically provides an output at the end of a workflow, though a fault may also be signalled
Start	identifies the start of a workflow if this is ambiguous

Table 1 CRESS Constructs used in Article ('?' optional, '*' zero or more, '+' one or more, '|' alternative)

{ **String** healthFile **String** healthVar **Float** healthMax } healthIn

A field in such a variable is accessed as *healthIn.healthFile* for example.

4.4 Other CRESS Diagrams

Although not used in this article, CRESS also supports feature diagrams that extend workflow diagrams in specified ways. This can be useful for behaviour that is common to several workflows. A feature is a macro-like workflow that can add to or replace part of the main workflow. Features are identified by patterns that are matched to the main workflow. For simplicity, feature diagrams are not used in this article. However, [34] can be consulted for information on how features are used.

Besides the workflow diagrams, CRESS also requires a configuration diagram that defines key parameters of workflows and their scripts. Tool options are also given in the configuration diagram, e.g. which host should be used for workflow execution. For a statistical workflow, the configuration diagram simply declares its name; its inputs and outputs are fully defined in the workflow diagram. For a statistical analysis service (i.e. script), the configuration diagram declares its outputs; its inputs are explicitly defined in the workflow diagram.

CRESS supports EPRs (Endpoint References) for web services. These are unique references to service resources like files. An endpoint reference can be passed between services without having to provide the file contents (which could be extensive). This is common practice when CRESS is used with web or grid services. However, it is inconvenient for statistical services since it would then be necessary to provide wrappers for scripts that convert input EPRs into local copies of files, and convert output files into EPRs. Instead a much simpler approach is followed: a file name string is treated as a resource reference.

A script can create zero, one or many outputs. As noted in section 3.1, it can be time-consuming or impossible to determine the precise outputs of a script. CRESS therefore requires an explicit statement of what these outputs are. In the simplest case, script outputs are fixed (e.g. a file called *merged.dta*). In some cases, the output file names depend on the input parameters (numbered positionally as '$1', '$2', etc. in CRESS). An output file may be named according to the parameters which defined the statistical analysis. For example, the output file *bhps1995-shs1995.dta* is generated from the pattern *$1-$2.dta* when used with input files *bhps1995.dta* and *shs1995.dta*.

A script may be simultaneously executed by several remote users. It is therefore necessary to ensure that output file names are uniquely qualified by using $$. (This notation is used in Unix, where it means the unique identifier of the process.) As an example (the graph script in figure 4), a script may be defined to have outputs *results/$1$$.eps* and *results/$1$$.txt*.

5 Case Study: Relating Occupation and Health

This section describes a realistic case study involving the analysis of occupations in relation to health. This makes use of two workflows and their associated scripts.

5.1 Original Conventional Analysis

Several applications illustrating statistical analysis for social science research purposes have been modelled by the authors. The example in this section is a relatively complex application that involves making time-based comparisons of the relationship between health and socio-economic circumstances. This general topic is a popular theme in research investigations [23]. It is fairly easy to measure both socio-economic circumstances and health indicators for the same people; a moderate correlation between the two would ordinarily be expected. The topic also raises interesting questions about the measurement of health and of socio-economic circumstances, for both of which there are numerous alternative candidate measures.

The case study uses datasets from the BHPS (British Household Panel Survey) and the SHS (Scottish Health Survey) introduced in section 1.2. These datasets are available for download from the UK Data Service (*www.ukdataservice.ac.uk*). The original analysis was conducted by a social scientist using conventional techniques, with eight Stata scripts ranging in length from 50 to 300 lines of code.

The original analysis extracted health and occupation information for individuals from the two surveys. Preliminary processing was undertaken to select and standardise health measures from a range of possible survey indicators, covering both subjective and objective measures. Preliminary processing was also performed to translate data about occupational titles (e.g. 'nurse') into occupation-based measures which could be used for direct statistical analysis [20].

In the original approach, comparisons were made between occupation-based measures and health measures for the same people at various points in time: it was possible to select data from any of the different years covered by the studies. For the case of the BHPS, which re-interviews the same respondents at different points in time, it was also possible to link a measure of socio-economic circumstances from one year with a measure of health in another. In such cases, the information from different sources was combined using shared identifier codes. Once the data was collected, it was possible to run a simple statistical summary of the relationship between occupational position and health at various times. Summaries of these were generated in both graphical and statistical forms.

The analysis results suggested a modest but statistically significant associations between subjective health outcomes and occupation-based measures which change little over the time periods examined. This is a consistent finding from such bivariate analyses. It is also well known that if the analysis includes additional controls for other related aspects of an individual's circumstances, especially age and gender, then a somewhat stronger association between health and occupation might be expected from the analysis.

5.2 Workflow-Based Analysis

The conventional approach described above was completely re-done using workflows. An investigation of the original analysis identified several opportunities for generalisation and re-use. One involved the extraction of health and occupation information from the relevant datasets: this became the *job_health* workflow described in section 5.2.1. This was then used as the foundation for higher-level analyses such as the *bhps_shs* workflow described in section 5.2.2. The smaller scripts used in the original analysis were largely preserved in the workflow solution, but were generalised to have clearly defined inputs, outputs and failures. For brevity formal aspects of the case study are not given in this article, but examples such as [37] provide more details.

5.2.1 Extracting Occupation and Health Information

Figure 3 shows the CRESS description of the *job_health* workflow that underpins all the higher-level analyses. The workflow starts with a request to combine occupational and health datasets (*job_health.dames.combine* is received in node 1). The *jobHealthIn* variable contains the names of two survey files, the names of the survey variables that identify occupation and health, and the maximum health scale value. Health is coded on a numerical scale; the highest value (indicating perfect health) depends on the survey and so is a workflow parameter.

The occupation and health measures are then processed in a way which draws on other external information. The occupational codes are converted to occupation-based measures on the basis of published tables of relationships, whilst the original measures of health are re-standardised using the information on the highest value held. Although not in the original problem definition, these tasks can be performed simultaneously to save on execution time. The workflow therefore has two parallel branches (fork at node 2). Health parameters are set up on one branch, and the *get_health* script is executed (*health.stata.get_health* is invoked in node 3). Similarly occupation parameters are set up on a second branch, and the *get_job* script is executed (*job.stata.get_job* is invoked in node 5). These parallel paths then converge (join in node 7). The join condition requires both parallel paths to complete successfully before the workflow progresses (3 && 5 means nodes 3 and 5 complete).

New measures on health and occupations are then set up and the *fuse_id* script is executed to combine the data according to the individual's personal identifier (*fuse.stata.fuse_id* is invoked in node 8). Finally, the results of this fusion are returned to the caller (a reply is given for *job_health.dames.combine* in node 9).

The workflow also supports fault handling that was not done in the original. In particular, the health and occupation scripts are at risk of failing due to errors in the external datasets. In principle other script invocations can also fail, but this is less likely because these operate on internally generated data. Fault handling therefore focuses on health and occupation extraction.

All nodes that execute scripts can implicitly give rise to faults. For example, node 3 can implicitly report a *healthError* fault with an accompanying *reason* (e.g. a data format error). Such a fault is caught at the top level of the script (arc to node 10). It is possible to have local fault handlers associated with individual script invocations, but it is often sufficient to handle faults globally.

In the event of a fault, the workflow may be left in an untidy state (e.g. large intermediate work files may remain). Key script invocations (those in nodes 3 and 5) are therefore associated with compensation handlers. When an invocation completes successfully its compensation handler become available. This can subsequently be called to undo work following a fault. Suppose health data is successfully extracted, but extraction of job data fails. This will be caught at the top level, leading to a request for compensation (node 10). This calls compensation handlers from the most recent to the earliest, thus unwinding the workflow.

In this case, the *delete_files* script will be called to delete the generated health data (*delete.stata.delete_files* is invoked in node 4). Following compensation, the fault is returned to the workflow caller (a reply is given for *job_health.dames.combine* in node 11). More complex patterns of compensation can be defined, such as extended compensation code that attempts to recover from a problem, or invoking specific compensation handlers according to the fault. This allows workflows to be managed more thoroughly than the original approach allowed.

5.2.2 Analysing Occupation and Health

Figure 4 shows the CRESS description of the one of the higher-level analyses. This is the *bhps_shs* workflow that analyses the correlation between occupation and health in 1995, for both the BHPS and the SHS datasets (separately). Other workflows analyse other years and also the trend over each decade. The workflow starts with a request to correlate health and occupation information (node 1). Unusually for a workflow, there are no parameters (denoted '–') because the datasets to be processed are fixed.

Again the workflow takes advantage of parallelism that was not identified in the original problem. The workflow forks (node 2) into two parallel paths that extract data from the two different datasets, BHPS 1995 and SHS 1995, respectively (invocations in nodes 3 and 4). In each dataset, names for the related measures are different (e.g. job variable *zjbsoc* in the BHPS file corresponds to *soc* in the SHS file). For each data source, the workflow invokes the *job_health* analysis task defined in figure 3; this generates derived measures of health and occupation based on the original measures found in the surveys. If the two datasets are successfully extracted (join in node 5), they are then combined using the *append_files* script (invocation in node 6). The correlation between the measures of health and occupation is then determined by the *correlate_graph* script in both graphical and

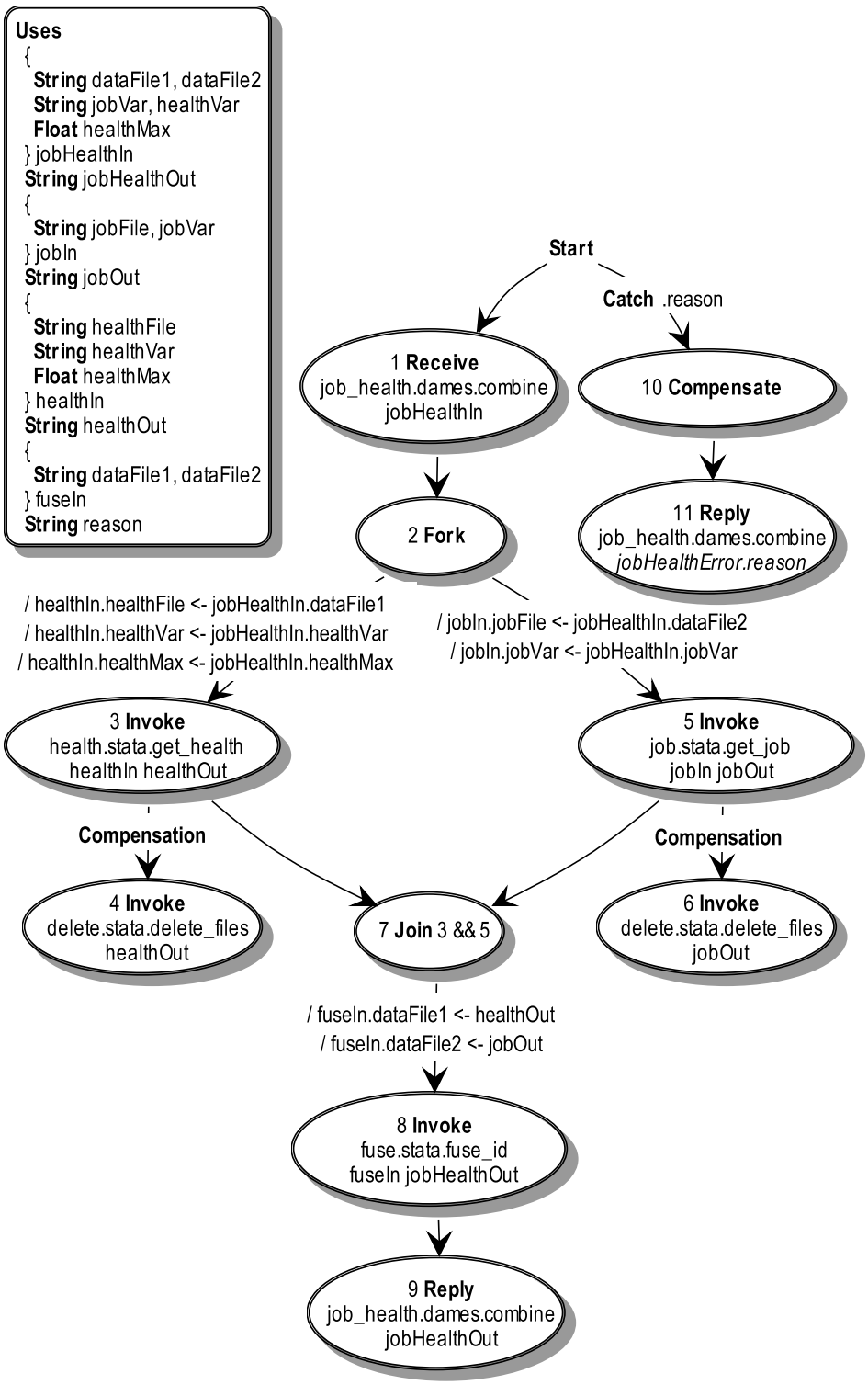

Fig. 3 Workflow to extract Health and Occupation Data

statistical form (invocation node 7). The *graphOut* results comprise two files: an EPS (Encapsulated PostScript) graph and a textual file with statistical summaries. Finally, this information is returned to the workflow caller (reply in node 8).

As for the workflow in figure 3, script faults are allowed for. However, there is no need for compensation as the scripts operate on internally generated data that should be reliable.

5.2.3 Analysis Results

As an example of the results from the high-level workflow (section 5.2.2), figure 5 shows a graphical summary of the relationship generated by the workflow for health and occupation in 1995. The graph layout uses the default Stata settings (though these can be adjusted). The scatter plot shows each individual's occupational advantage score (horizontal axis) and health outcome score (vertical axis). The horizontal line shows a linear regression best fit for the relationship between these two variables. The spread indicates the 95% confidence intervals, showing the margin of error in the regression line. In this particular instance, the relationship is relatively weak (i.e. there is not a strong differentiation in health outcome by occupational advantage).

A common research objective for this style of analysis is to compare the relationship between health and occupation across a number of different datasets, such as from different time points or geographical regions. This is a case where repeatability is needed (of the same analysis using data with different parameters) for which the workflow approach is ideal.

Besides the graphical output, a statistical summary (not shown here) is also generated for the same relationship and is stored as an output.

6 Conclusion

6.1 Summary

It has been seen that current practice in social science research suffers from limitations in how statistical analyses are developed: lack of technical expertise, very limited reuse, restricted public documentation, difficulty in making analyses repeatable by others, lack of high-level workflows, no exploitation of parallel execution, and little integrity checking. Nevertheless the results of statistical analysis in the social sciences can have far-reaching consequences. For example, Government policy on health service provision may be determined by the analysis of a health survey. Consequently it is desirable to improve standards in the analysis of quantitative datasets, particularly with regard to documenting and publishing records of the research.

The work reported in this paper is a response to these issues. The focus has been on effective tools that transfer advanced computing techniques into mainstream social science. It is believed that the objectives for this work in section 1.4 have been addressed. The approach is general, but has been illustrated with statistical analysis scripts in workflows. Reuse of statistical scripts is achieved through shared use of the services created from them, with workflows also being reusable. High-performance computing techniques are used to allow parallel execution where the workflow structure allows it. Formal methods can also be used to check workflows thoroughly prior to deployment and execution.

In the approach presented, statistical analysis scripts are modelled abstractly as mathematical functions and concretely as web services. This permits their composition into high-level workflows using BPEL. The client installation to use workflows is minimal: just a workflow editor like CHIVE. The client typically communicates with a server, where workflows are generated by CRESS and coordinated by ACTIVEBPEL. Scripts are executed by a designated host or a Condor pool under control of the workflow. The latter improves performance through load sharing and parallel execution.

The CRESS notation for statistical analysis workflows has been introduced. This has been illustrated through a realistic case study using actual datasets and statistical scripts supplied by a social scientist. This has demonstrated the viability of the approach.

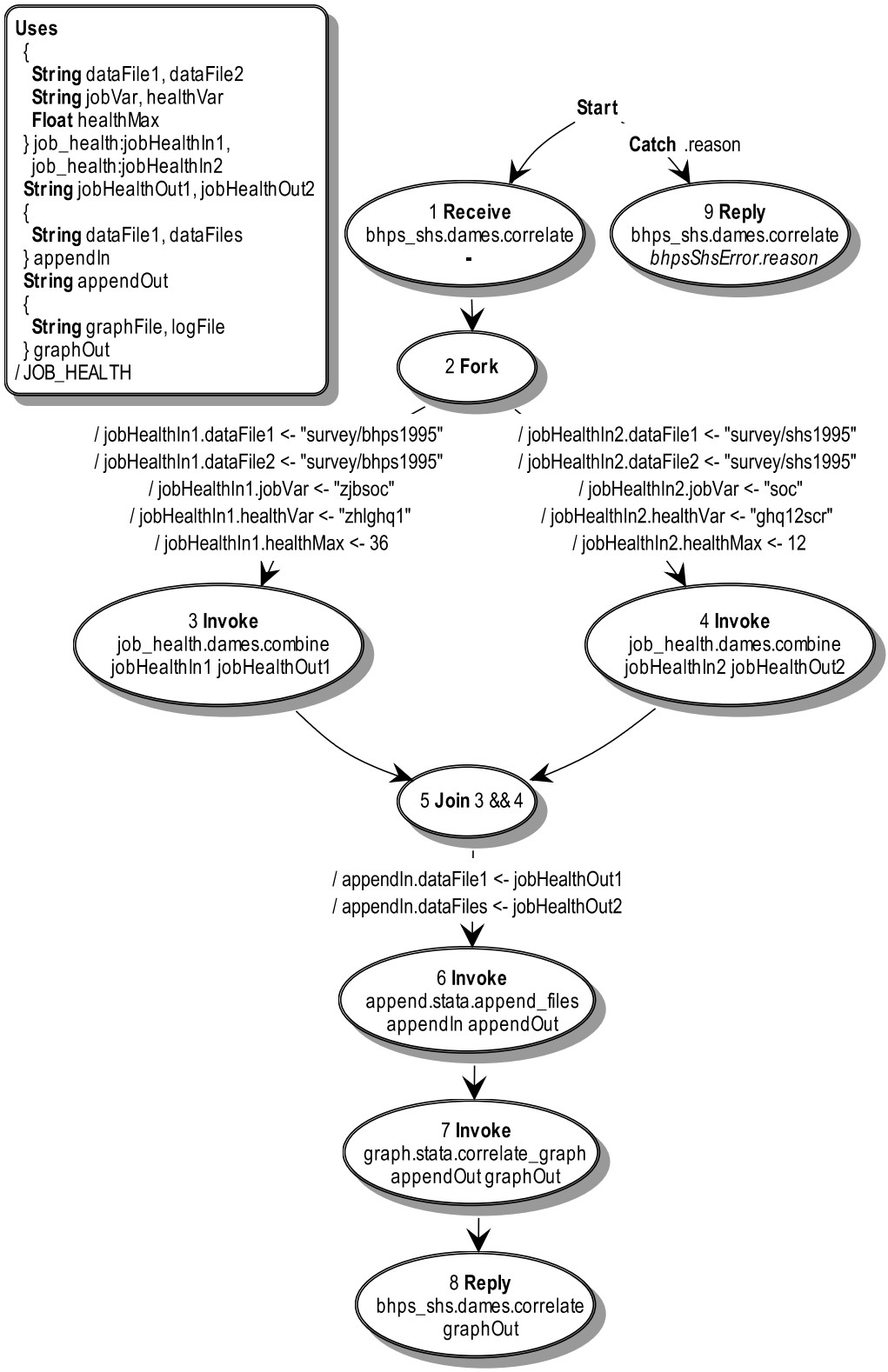

Fig. 4 Workflow to analyse BHPS and SHS Data

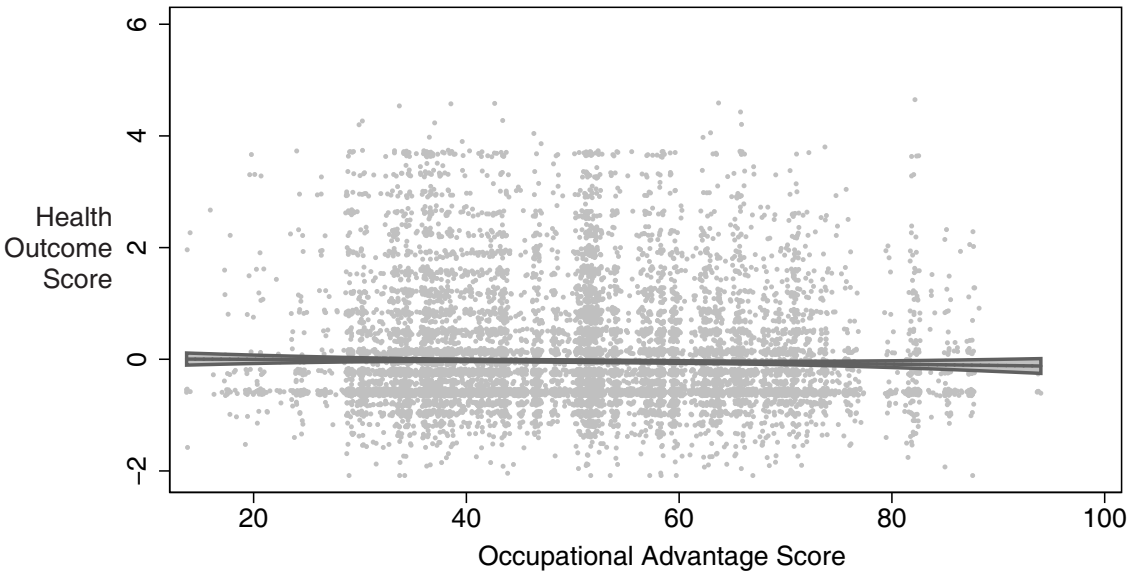

Fig. 5 Graphical Correlation generated for Health and Occupation

6.2 Discussion

This work is a contribution to data-intensive scientific research. In particular, it offers a workflow-based solution for performing large-scale statistical analysis in a more modular, reusable and efficient way. Computational workflows are not new, but the approach described in the paper offers distinctive advantages. Although scientific workflows have been developed in some disciplines, notably for bioinformatics, their application to quantitative social science is novel. Apart from the preliminary efforts on the NeISS project mentioned in section 2.2, the work presented here on social science workflows is unique as far as the authors are aware.

Some approaches have been designed to pre-populate analytical scripts in quantitative social science applications. For example e-STAT (section 2.1) supports templates in alternative statistical languages. Other approaches have focused on data processing rather than analysis. For example, PanelWhiz (*www.panelwhiz.eu*) prepares data extraction codes for Stata, and the Minnesota Population Centre (*www.ipums.org*) use a data extraction syntax generator. These cater for specific forms of data processing and analysis, whereas the workflow approach in this article applies much more generically across the many and varied tasks in quantitative social research.

The CRESS notation, toolset and methodology used in the work are mature, having been under development for 15 years. However, the specific developments to support statistical workflows are new. The strengths of CRESS are as follows:

- The CRESS notation is straightforward and easily learned by non-specialists. As noted in section 4.1, the usability of CRESS by developers has been successfully demonstrated.
- CRESS has demonstrated its flexibility by being used in multiple areas: Intelligent Network, Internet telephony, Interactive Voice Response, Web Services, Grid Services and Device Services.
- CRESS eases workflow analysis by providing high-level languages for formal verification and formal validation (CLOVE and MUSTARD). Some verification can be performed without any effort by the workflow developer. More specialised verification is possible, and is eased by automatic translation from high-level properties into temporal logic. Formal validation is more pragmatic and based on developer-defined tests. These can also be re-used for performance checking.
- The CRESS toolset is portable (being written in Perl and Java) and has been run on multiple platforms.

The case studies already conducted using the workflow approach are encouraging evidence that it will be suitable for social scientists and will scale up to larger problems.

6.3 Future Work

One promising area for future development lies in sharing services and workflows among researchers. This is currently achieved by depositing social science workflows on the DAMES server. It is planned to look into how CRESS workflows could also be integrated with myExperiment.

Because the approach does not rely on copying the files used by workflows, it can scale up adequately for large datasets. CRESS has also been demonstrated on very complex workflows (up to 26 pages of diagrammatic description). However, it would be useful to carry out actual performance measurements with larger datasets and more extended analyses.

Preliminary investigations have looked into using CRESS for orchestrating the analyses performed in neuroscience. Here, the building blocks are completely different (programs to analyse empirical data from neural measurements). The suitability of the approach will also be investigated for environmental science workflows, e.g. the analysis of crop data or fishery data.

Acknowledgements

The work reported in this paper was conducted on the DAMES project, which was led by the second author. DAMES was funded by the Economic and Social Research Council under grant number RES-149-25-1066. The authors are grateful to their colleagues on DAMES for collaboration on the general topic of techniques for e-social science.

Support of social science workflows was jointly developed by Koon Leai Larry Tan and the first author as part of the DAMES project. Guy C. Warner built the occupational and educational portals for DAMES, and created the server infrastructure needed for the work reported here. He also provided technical advice on using workflows in conjunction with the DAMES servers.

References

1. A. Arkin, S. Askary, B. Bloch, F. Curbera, Y. Goland, N. Kartha, C. K. Lie, S. Thatte, P. Yendluri, and A. Yiu, editors. *Web Services Business Process Execution Language*. Version 2.0. Organization for The Advancement of Structured Information Standards, Billerica, Massachusetts, USA, Apr. 2007.
2. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Paterson, A. Rabkin, I. Stoica, and M. Zaharaia. A view of cloud computing. *Communications of The ACM*, 53(4):50–58, Apr. 2010.
3. J. Bethlehem. Surveys without questions. In E. De Leeuw, J. Hox, and D. A. Dillman, editors, *International Handbook of Survey Methodology*, chapter 26, pages 500–511. Psychology Press, London, 2008.
4. G. Blank and K. B. Rasmussen. The Data Documentation Initiative: The value and significance of a worldwide standard. *Social Science Computer Review*, 22(3), Oct. 2004.
5. J. Bradfield and C. Stirling. Modal mu-calculi. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*. Elsevier Science Publishers, Amsterdam, Netherlands, 2007.
6. W. J. Browne, B. Cameron, C. M. J. Charlton, D. T. Michaelides, R. M. A. Parker, C. Szmaragd, H. Yang, and Z. Zhang. A beginner's guide to Stat-JR. Centre for Multilevel Modelling, University of Bristol, 2012.
7. M. Butler, C. Ferreira, and M. Y. Ng. Specifying and verifying web transactions. *Universal Computer Science*, 11(5):712–743, May 2005.
8. A. Chirichiello and G. Salaün. Encoding abstract descriptions into executable web services: Towards a formal development. In *Proc. Web Intelligence 2005*. Institution of Electrical and Electronic Engineers Press, New York, USA, Dec. 2005.
9. D. C. de Roure, C. A. Goble, and R. Stevens. PX: A system extracting programs from proofs. In G. Fox, K. Chiu, and R. Buyya, editors, *Proc. Int. Conf. on 3rd e-Science and Grid Computing*, pages 603–610. Institution of Electrical and Electronic Engineers Press, New York, USA, Dec. 2007.
10. Economic and Social Data Service and University of Manchester. Linking international macro and micro data. *www.esds.ac.uk/international/elearning/limmd*, Nov. 2012.
11. A. Ferrara. Web services: A process algebra approach. In *Proc. 2nd International Conference on Service-Oriented Computing*, pages 242–251. ACM Press, New York, USA, Nov. 2004.
12. H. Foster. *A Rigorous Approach to Engineering Web Service Compositions*. PhD thesis, Imperial College, London, Jan. 2006.
13. I. Foster. What is the grid? A three point checklist. *Grid Today*, 1(6), July 2002.
14. J. Freese. Replication standards for quantitative social science: Why not sociology? *Sociological Methods and Research*, 36(2):153–171, 2007.
15. X. Fu, T. Bultan, and J. Su. Analysis of interacting BPEL web services. In *Proc. 13th. International World Wide Web Conference*, pages 621–630. ACM Press, New York, USA, May 2004.

16. M. Ghanem, Y. Guo, A. Rowe, and P. Wendel. Grid-based knowledge discovery services for high throughput informatics. In *Proc. 11th Int. Symp. on High Performance Distributed Computing*, pages 198–212. Institution of Electrical and Electronic Engineers Press, New York, USA, July 2002.
17. T. Hey, S. Tansley, and K. Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond, Washington, Oct. 2009.
18. ISO/IEC. *Information Processing Systems – Open Systems Interconnection – LOTOS – A Formal Description Technique based on the Temporal Ordering of Observational Behaviour*. ISO/IEC 8807. International Organization for Standardization, Geneva, Switzerland, 1989.
19. N. Kaveh and W. Emmerich. Validating distributed object and component designs. In M. Bernardo and P. Inverardi, editors, *Formal Methods for Software Architecture*, number 2804 in Lecture Notes in Computer Science, pages 63–91. Springer, Berlin, Germany, Sept. 2003.
20. P. S. Lambert and E. Bihagen. Stratification research and occupation-based classifications. In P. S. Lambert, R. Connelly, R. M. Blackburn, and V. Gayle, editors, *Social Stratification: Trends and Processes*, chapter 2, pages 13–28. Ashgate, Aldershot, UK, 2012.
21. J. Li, H. Zhu, and J. He. Specifying and verifying web transactions. In K. Suzuki, T. Higashino, K. Yasumoto, and K. El-Fakih, editors, *Proc. Formal Techniques for Networked and Distributed Systems (FORTE 2008)*, number 5048 in Lecture Notes in Computer Science, pages 168–183. Springer, Berlin, Germany, June 2008.
22. J. S. Long. *The Workflow of Data Analysis using Stata*. CRC Press, Boca Raton, Florida, USA, 2009.
23. J. P. Mackenbach, L. Stirbu, A. J. Roskam, M. M. Schaap, G. Menvielle, M. Leinsalu, and A. E. Kunst. Socioeconomic inequalities in health in 22 European countries. *New England Journal of Medicine*, 358(23):2468–2481, 2008.
24. T. Margaria, C. Kubczak, and B. Steffen. Bio-jETI: A service integration, design and provisioning platform for orchestrated bioinformatics processes. *BMC Bioinformatics*, 9(4):1614–1631, Apr. 2008.
25. S. McVie, A. P. M. Coxon, P. Hawkins, J. Palmer, and R. Rice. ESRC/SFC scoping study into quantitative methods capacity building in Scotland. Economic and Social Research Council, 2008.
26. 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. *Bioinformatics*, 20(17):3045–3054, 2004.
27. C. Pautasso. JOpera: An agile environment for web service composition with visual unit testing and refactoring. In *Proc. IEEE Symposium on Visual Languages and Human Centric Computing*. Institution of Electrical and Electronic Engineers Press, New York, USA, Nov. 2005.
28. J. Qin and T. Fahringer, editors. *Scientific Workflows – Programming, Optimization, and Synthesis with ASKALON and AWDL*. Springer, Berlin, Germany, Aug. 2012.
29. S. N. Smith, S. D. Fisher, and A. Heath. Opportunities and challenges in the expansion of cross-national survey research. *Social Research Methodology*, 14(6):485–502, 2011.
30. B. Steffen, T. Margaria, R. Nagel, S. Jörges, and C. Kubczak. Model-driven development with the jABC. In E. Bin, A. Ziv, and S. Ur, editors, *Hardware and Software, Verification and Testing*, number 4383 in Lecture Notes in Computer Science, pages 92–108. Springer, Berlin, Germany, May 2007.
31. T. Tannenbaum, D. Wright, K. Miller, and M. Livny. Condor: A distributed job scheduler. In W. Gropp, E. Lusk, and T. Sterling, editors, *Beowulf Cluster Computing with Linux*, pages 307–350. MIT Press, Boston, USA, Nov. 2003.
32. I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, editors. *Workflows for E-Science: Scientific Workflows for Grids*. Springer, Berlin, Germany, 2007.
33. D. J. Treiman. *Quantitative Data Analysis: Doing Social Research to test Ideas*. Jossey Bass, New York, 2009.
34. K. J. Turner. Analysing interactive voice services. *Computer Networks*, 45(5):665–685, Aug. 2004.
35. K. J. Turner. Validating feature-based specifications. *Software Practice and Experience*, 36(10):999–1027, Aug. 2006.
36. K. J. Turner. Flexible management of smart homes. *Ambient Intelligence and Smart Environments*, 3(2):83–110, May 2011.
37. K. J. Turner and K. L. L. Tan. Rigorous development of composite grid services. *Network and Computer Applications*, 35(4):1304–1316, Feb. 2012.
38. B. Wassermann, W. Emmerich, B. Butchart, N. Cameron, L. Chen, and J. Patel. Sedna: A BPEL-based environment for visual scientific workflow modelling. In I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, editors, *Workflows for E-Science*, pages 428–449. Springer, Berlin, Germany, 2007.
39. M. Wirsing, A. Clark, S. Gilmore, M. Hölzl, A. Knapp, N. Koch, and A. Schröder. Sensoria process calculi for service-oriented computing. In E. Najm and J.-F. Pradat-Peyre, editors, *Proc. Formal Techniques for Networked and Distributed Systems (FORTE 2006)*, number 4229 in Lecture Notes in Computer Science, pages 24–45. Springer, Berlin, Germany, Jan. 2006.
40. J. Yu, J. Han, P. Falcarin, and M. Morisio. Using temporal business rules to synthesize service composition process models. In M. van Sinderen, editor, *Proc. 1st Int. Workshop on Architectures, Concepts and Technologies for Service Oriented Computing*, pages 86–95. INSTICC Press, Setúbal, Portugal, July 2007.