Journal of Visual Languages and Computing, Vol 26, Feb 2015, pp 99–125, (c) Elsevier 2015

A suite of visual languages for model-driven development of statistical surveys and services

Chul Hwee Kim
Fides Treasury Services Ltd.
Badenerstrasse 141
P.O. Box CH-8036
Zürich, Switzerland
ckim001@gmail.com
Phone +41-44-298-6559

John Grundy
Centre for Computing & Engineering Software Systems
Swinburne University of Technology
PO Box 218, Melbourne, Australia
jgrundy@swin.edu.au
Ph: +61-3-9214-8731

John Hosking
College of Engineering and Computer Science
Australian National University
Canberra, Australia
john.hosking@anu.edu.au
Ph: + 61-2-6125 8807

Abstract

Objective: *to provide statistician end users with a visual language environment for complex statistical survey design and implementation.* ***Methods:*** *we have developed, in conjunction with professional statisticians, the Statistical Design Language (SDL), an integrated suite of visual languages aimed at supporting the process of designing statistical surveys, and its support environment, SDLTool. SDL comprises five diagrammatic notations: survey diagrams, data diagrams, technique diagrams, task diagrams and process diagrams. SDLTool provides an integrated environment supporting design, coordination, execution, sharing and publication of complex statistical survey techniques as web services. SDLTool allows association of model components with survey artefacts, including data sets, metadata, and statistical package analysis scripts, with the ability to execute elements of the survey design model to implement survey analysis.* ***Results:*** *we describe three evaluations of SDL and SDLTool: use of the notation by expert statistician to design and execute surveys; useability evaluation of the environment; and assessment of several generated statistical analysis web services.* ***Conclusion****: we have shown the effectiveness of SDLTool for supporting statistical survey design and implementation.* ***Practice Implications****: we have developed a more effective approach to supporting statisticians in their survey design work.*

1. Introduction

Statistical surveys have extensive roots running back to ancient times [27]. The development of probability theory and mathematical statistics provided a scientific foundation for statistical surveys [1] and they have become a valuable and ubiquitous tool for obtaining trustworthy information. As such, statistical surveys are a very common tool for obtaining trustworthy information about a set of objects comprising a target population in many diverse domains, including market research, government policy development, and academic research. The goal of a survey is to describe the population by one or more parameters defined in terms of measurable properties. This in turn requires a *frame,* providing access to the population (eg a phone book or electoral roll), and a method for sampling from that frame [1]. Figure 1 illustrates the typical iterative process involved in defining and executing a statistical survey.

In addition to the survey process, other characteristics of a survey that need to be modelled include survey data, data analysis and relationships between process, data and analysis, and low-level data collection and analysis tasks. Many tools support aspects of realizing the survey process, including SurveyCraft [41] and Blaise [42] for questionnaire design and response collection, SAS [39] for complex data analysis, and SPSS [40] for general statistical process design. However most statistical survey support tools have been limited to supporting only parts of the surveying process [8][17], typically low-level statistical *technique implementations* including data capture, processing and analysis. Many other aspects such as statistical metadata, heterogeneity in statistical data semantics, and other non-mathematical activities are less well addressed [10], [19]. Available tools are also usually narrow in their focus and there is no agreed notation for defining a statistical survey overall.

We have been developing a higher-level statistical survey process support tool incorporating a range of domain-specific visual languages to enhance statistical survey design, implementation and reuse [20], [23]. To this end we developed the Survey Design Language (SDL) and its support tool, SDLTool, an integrated set of visual notations aimed at supporting survey design in a similar way that the Unified Modelling Language (UML) [36] supports software development. Our aim was to design a set of visual notations for statistical survey design that fill a similar role as the Unified Modelling Language (UML) [37] does in software design. Through these notations we aimed to:

- make statistical survey design more accessible.
- improve the effectiveness and efficiency of implementing statistical surveys
- provide better tool support for survey designers

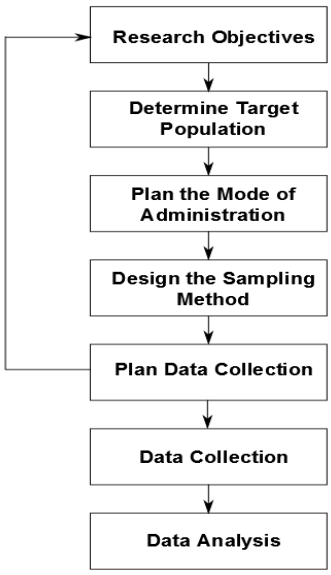


Figure 1. Basic statistical survey process (from [2]).

Our research concerns practical issues in supporting survey processes. From a survey practitioner's viewpoint, statistical computing is well supported by high quality software packages like R [10], so low-level statistical technique implementation is not a major operational concern. However many other aspects such as statistical metadata [31] [19], heterogeneity in statistical data semantics [33], making complex techniques accessible and reusable by others, documenting complex survey processes and tasks, and other non-mathematical activities are less well addressed. SDLTool includes support for the generation of complex web services that provide data capture, processing and analysis support via off-the-shelf survey analysis tools. These survey implementations are generated as web services from high-level models and allow heterogeneous 3rd party applications to make use of these packaged techniques implemented by remote COTS statistical analysis tools. Generation of rich documentation for these packaged statistical survey services is also supported.

SDL evolved from an initial set of design notations to become a fully integrated environment that supports design, implementation and publication of the statistical survey process. Our hypothesis is that when the semantics of survey designs are visually specified and modelled in a visual language the following benefits are obtained:

- Mitigation of communication overheads for both non-experts and experts. A difficult survey concept can be represented with a graphical metaphor, an abstraction isolating low-level details from high-level concepts.
- Harmonisation of disparate operational semantics.
- Model-driven management and execution of the survey process.

We begin with a motivating example for this work, the New Zealand National Survey of Crime Victims, and from this review related work and define a set of key requirements we wanted to address. We then provide an overview of our approach and integrated environment and the SDL domain-specific visual languages. We use a usage example to explain the features of the SDLTool including the association of resources with SDL elements, the execution of fully defined statistical techniques and operations, visualisation of results, and publication of survey documentation and code. We then describe SDLTool's key architectural and implementation features. We present three evaluations we have carried out to judge how well this research has met our key requirements: an expert statistician usage and feedback on SDL; a user evaluation of SDLTool informed by Cognitive Dimensions theory [16]; and assessment of generated statistical analysis web services and documentation. We conclude with some key directions for future research in this area.

2. Motivation, Requirements and Research Question

The primary source of crime victimisation data for e.g. government agencies and the news media is usually law enforcement agencies. However the reality of victimisation may not be sufficiently captured from the one source and therefore victimisation data might usefully be supplemented by a statistical survey to:

- Better identify at risk groups.
- Provide an alternative measure of crime victimisation

- Describe both explicit and implicit effects of crime.

This broadly is the theme of the New Zealand National Survey of Crime Victims in 2001 [33]. The survey was a door-to-door survey targeted at New Zealanders aged 15 or more. Each region was assigned a unique area code, and a stratification of the area codes done. Random samples of strata were compiled and a pre-defined visit pattern was applied in collecting data e.g. 10th street, 10th house.

Figure 2 shows how the survey outlined above is depicted and elaborated using an SDL *survey diagram* (described in detail below). This was conceived as a brainstorming view for early conceptualisation of the survey design. Key *survey attributes* are clustered around a set of user-defined *survey contexts*. Together these visualise key processes in formulating the high-level nature of the survey. SDL provides a variety of such domain-specific visual language diagram types with which to model complex statistical surveys. These include survey diagrams (to model the overall context of a survey); survey task and data diagrams (to model data collection and analysis); survey technique diagrams (to model low-level details of statistical analysis processes); and survey process diagrams (to model high-level survey task interactions).

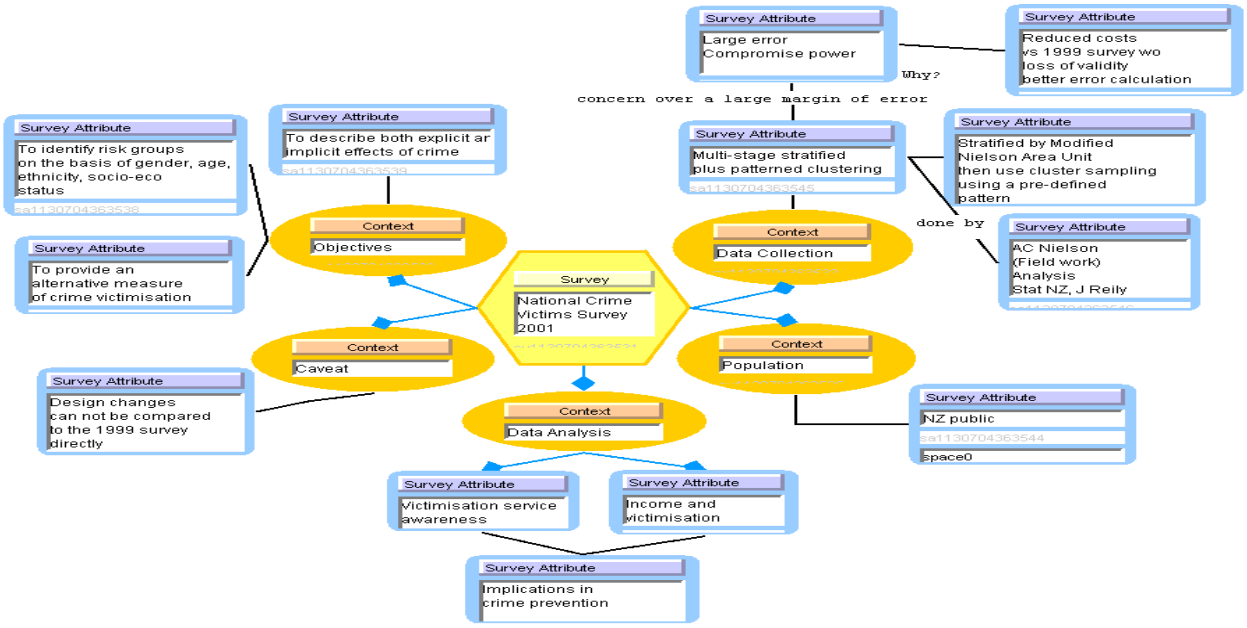


Figure 2. An example SDL Survey Diagram for the 2001 New Zealand Crime Victimisation Survey

(© 2007 IEEE. Reprinted, with permission, from [22])

Currently developers of such statistical surveys have to build custom models of the survey process in an ad-hoc fashion outside their off-the-shelf analysis tools; build survey-specific data capture, processing and analysis *technique implementations* with multiple statistical analysis tools, such as R [17]; and often have great difficulty sharing these technique implementations as most analysis tools use proprietary representational techniques. Our background in the development of visual languages and environments suggested that an integrated visual language approach could address these problems and led us to formulating the following key research question, which we attempt to answer in the remainder of this paper:

RQ1: Can visual notations with appropriate software tool support be used by end users, i.e. statisticians and other survey developers, to facilitate statistical survey design and implementation?

3. Related Work

A variety of work has been done in the End User Development space using visual languages to support non-technical end users in developing complex solutions [26]. Common domains include but are not limited to supporting user-defined mash-ups, web and mobile application design and scripting, information visualisation, supporting end-user testing of apps and services, and web services composition. The most relevant to our work includes approaches to model domain-specific processes and tasks, and to script computational techniques. These include a variety of domain-specific languages for end users including business process modelling [24], biological modelling with state charts [20], ontology modelling languages [24], and visual composition of applications from basic building-blocks [10]. All of these approaches define one or more visual languages allowing target end users to describe designs and solutions in the target

domain using visual modelling abstractions. In our work, we want to enable end user design of complex statistical surveys including processes, tasks, data, and analytical techniques.

A range of approaches have been developed to support end user visual modelling of complex web service compositions. The most relevant to our work includes support for web service compositions using composition-oriented techniques [45], work on specifying service compositions for learning management systems [12], service-based user interface orchestration [11], and end user development of various government and business services [13]. These approaches aim to enable target end users to define and compose parts of complex service-based systems leveraging their domain-specific knowledge. These end-user defined or configured services can then often be reused by others. In our work, we want to enable packaging and reuse of survey techniques as services.

Statistical surveys have become an essential quantitative information gathering and analysis tool in a wide variety of domains. Statistical data is used in the development of products, the analysis of organisational and government performance, the understanding of audiences for TV and radio, political and economic commentary and decision making, and teaching and research within Universities. Designing and implementing a good statistical survey requires a range of skills and experience, and ideally good survey design support tools. Additionally, the inception of large-scale surveys, such as the US census, brought enormous managerial complexity prompting development of statistical computing [5]. However, designing and implementing good statistical surveys is challenging.

Current survey supporting tools can be generalised into three broad categories: Computing-centric tools, statistical applications, and interviewing and data collection aids. Computing centric tools, which are almost synonymous with statistical computing, build on domain specific languages such as S and execution engines. They provide an excellent numerical computing environment, but their steep learning curve and the high level of investment in low-level implementations required can offset their advantages. Statistical applications provide an environment where users can be insulated from the low-level complexity by introducing a user centred interface for setting up high-level activities. This approach helps users focus their efforts on carrying out the survey process itself, however the proprietary lock-in of operational procedures and back end functionality which overlaps with the computing centric tools, where statistical applications lack the power of dedicated computing tools, are notable trade-offs. Interviewing and data collection aids focus on the early stages of the survey process thus their benefits do not flow into other parts of the survey process. Thus, while existing survey tools are generally useful they all tend to address only specific parts of the survey process. This forces users to weave a set of tools that lack a semantics to describe the overall survey process.

A small number of tools aim to support the survey process more generally. ViSta [46] provides a visually guided and structured environment for data analysis with multiple visual models to suit various end user groups. GuideMaps help users carry out data analyses even when they lack detailed technical knowledge. WorkMaps visualise a data analysis capturing valuable contextual information such as analytical steps taken. Task based organisation is well expressed but there is little support for merging tasks to support survey contexts, such as objectives, and data collection and analysis or for non-analytic tasks e.g. questionnaire design and metadata support is lacking. Many software packages have been developed to support aspects of the survey process including SurveyCraft [41] and Blaise [42] for questionnaire design and response collection, SAS [39] for complex data analysis and SPSS [41] for statistical process design. However, such tools are typically narrow in their focus and the lack of integration and cross tool support, hindered by the absence of mature and widely accepted inter-operable standards, are common sources of difficulty in the survey process. ViSta's Lisp-based extensions are a barrier to use given the dominance of S-based languages. ViSta also has poor back end integration support.

CSPro [8] assists users from the data entry stage to produce error and inconsistency free data. It supports data entry, batch edit, and tabulation applications with powerful logic programming support. It only covers some stages of the survey process, with no support for sampling design and data analysis. Thus it is not a solution platform for users to integrate multi-faceted aspects of statistical surveys but it is a very capable tool in the areas it covers.

Statistical data and metadata standard initiatives include Triple-S [19], DDI [9] and MetaNet [31]. These efforts tend to be localised or discipline-specific, but understanding them has given us useful insight into incorporation of statistical data and metadata into SDL.

Our work has been strongly influenced by UML [36], particularly its use of multiple notations, supporting multiple modelling spaces [44]. While the UML is too software domain-specific for direct use in survey design [31], the historical development of UML provides valuable insight into visual languages that deal with a complex multidimensional problems. It is no coincidence that the high level aims of SDL closely parallel that of UML: *Visualisation*: SDL notations and diagrams provide visual representations of a statistical survey, spanning the entire survey process, but with each diagram visualising a specific aspect of the survey; *Specification*: the SDL diagrams as a whole assemble specifications of survey artefacts, their attributes and relationships and resource descriptions mapping artefacts to physical resources and services; *Implementation and execution:* similar to UML's MDA approach, SDL diagrams can generate and orchestrate software solutions for a statistical survey; *Documentation:* SDL can largely automate the documentation of the survey process.

4. Our Approach

After initial informal discussion with statisticians, we postulated that a set of integrated, Domain-Specific Visual Languages (DSVLs), each designed to model specific aspects of statistical surveys, would provide statisticians with an appropriate set of high-level and low-level modelling facilities, matching their cognitive models of survey design. A support environment for the modelling of statistical surveys using these DSVLs was needed that must incorporate modelling of survey resources, associating resources to statistical survey design elements, running modelling surveys on the target population datasets, and providing a range of data visualisation support features. Generating reusable scripts for 3rd party packages eases the burden of producing such scripts, which are often quite complex and hard-to-maintain directly, and enables reuse of the modelled techniques. Generated web services providing remote statistical technique implementations then provide a set of analysis services, implemented by the existing 3rd party statistical analysis packages, for both applications and other analysis tools, without users needing knowledge of the implementation of the techniques. They also have the added advantage of allowing 3rd party analysis tools to be used by others without the need to have an installation of the tools themselves.

Figure 3 illustrates the process of designing, generating and using statistical technique implementations using SDL and its supporting SDLTool. One or more repositories store statistical survey meta-data and/or SDLTool survey models (1). Some of these repositories may be 3rd party while others might be SDLTool-specific. Users import one or more useful SDL model templates and/or meta-data formats from these repositories and then develop their own statistical survey designs (2). SDLTool supports collaboration with other users by sharing SDL models and diagrams using a CVS repository and synchronous collaborative editing (3). Once a statistician has completed detailed survey technique specification, scripts are generated by SDLTool (4) that drive external 3rd party statistical processing and analysis packages, such as R [17]. After testing these survey technique implementation scripts the user instructs SDLTool to generate web service interface implementations (currently in Java using the Java Web Service Development Kit framework and tools) (5). Additionally WSDL descriptions of these services are generated and deployed to a web host, currently Axis (6). External tools such as existing statistical survey packages or bespoke Java or .NET applications can discover and invoke the packaged survey technique implementations via their web services (7). Required data is passed to the scripts via the generated web service interface, and result data is similarly packaged by the generated web service implementation and returned to the 3rd party application.

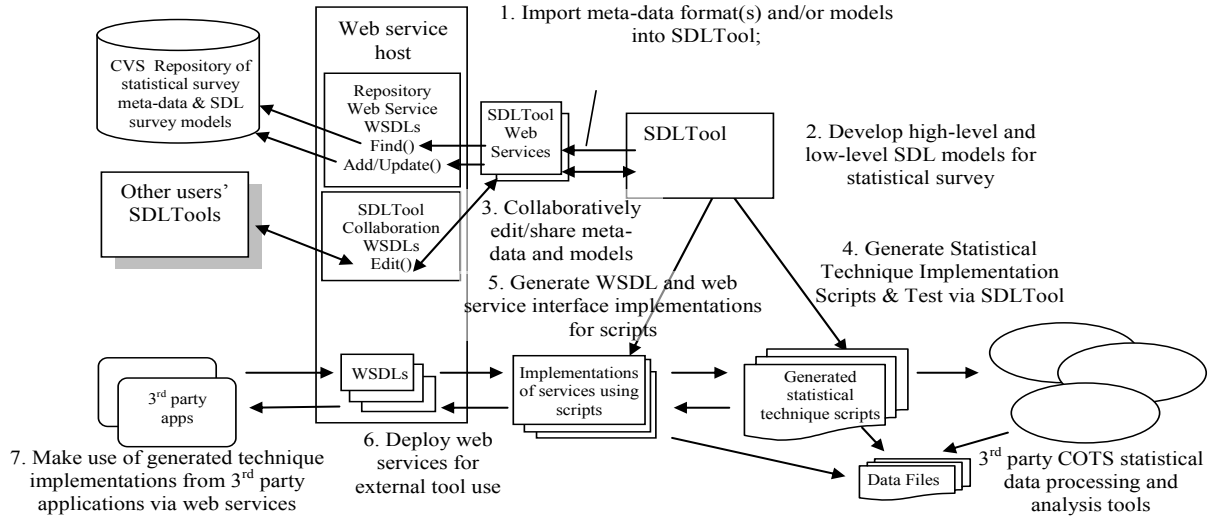


Figure 3. SDLTool usage overview.

Our approach to designing SDL was informed by several overlapping methodologies and our interest in answering the key research questions from Section 2. The first approach we used was Liu and Liebermann's of extracting semantics from natural language [28]. This was applied by examining a corpus of existing surveys and extracting key design elements and relationships. This helped establish an ontology for survey design and expressed key "building blocks" for statistical surveys. We used this ontology to provide the key elements for our underlying SDL meta-model and its associated visual languages. We incrementally enhanced this set of concepts using task analysis and from initial feedback showing statisticians our prototype SDL language designs.

The second approach we used was due to what we observed to be the close relationship between the requirements of survey designers and those of process centred software engineering environments [15], an earlier area of our research. Surveys are very process-centric, with each stage in a statistical process having associated resources (data), processing agents (human and machine), and so on. This suggested a process-oriented viewpoint would be an important component of SDL and would be used to link elements from data and analysis viewpoints. This led us to define SDL support for the process of survey construction based on earlier, successful software process support approaches.

A user-centric design approach was our third methodological tool [5]. We informally interviewed several experienced statisticians and asked them to walk through some example survey design tasks. These ranged from Professors of Statistics at the University of Auckland, to post-graduate statistics students, to non-statistician, more informal users of statistical surveys. We then conducted a task analysis to better understand what they did, when they did it and why they did it. From these identified tasks, we identified concepts in our ontological model that were being defined, refined and related during different statistical survey design tasks. Our task analysis clearly showed that survey construction often progresses "bottom-up" from a set of loosely connected goals and analysis tasks. During this task analysis we realised that our new SDL languages thus must strongly support "brainstorming" for determining survey objectives and survey design refinement. Consequences of this mean than users will incrementally define and build SDL diagrams in varying orders and completeness, and will iteratively refine and extend these models in practice. Additionally, we incrementally refined with these statisticians our initial findings from the ontological concepts and process-centric language examples above, and early SDL notation designs, described in the following section.

By application of these approaches, we identified, in conjunction with our statistician end-users, the following set of requirements needed to support more effective statistical survey technique definition, execution and sharing between analysis tools:

- *Modelling of statistical surveys.* Defining complex statistical surveys is very challenging with no agreed notations for many aspects of the survey e.g. overall process, particular techniques to use, specific tasks to carry out. As part of this modelling, informal brainstorming to determine survey objectives and survey design refinement is needed.
- *Association of Models with Resources.* An abstract survey definition needs to be instantiated with a myriad of specific population information sources, data collection and management tasks, statistical analysis technique implementations, coefficients, and result visualisation techniques.
- *Execution of surveys.* A support environment needs to provide exploratory task and technique application to partial datasets.
- *Generation of statistical technique implementations*. Once a set of statistical techniques has been fully specified, scripts implementing these for 3rd party statistical analysis tools can be generated. These scripts, often parameterised, can then be run to reuse the modelled techniques.
- *Wrapping of generated technique implementations in web services*. Tool users should be able to use techniques based on diagrammatic specifications outside of the environment. To make these technique implementations accessible to other tools, web services wrapping them need to be generated and deployed for discovery and invocation.
- *External tool interaction with generated technique web services*. 3rd party statistical tools can discover and invoke the generated statistical technique implementations via their generated web services.

Finally, we were strongly influenced by Burnett et al's guidelines for robust visual language development [6], particularly their four ideal characteristics for visual languages: fewer concepts, explicit depiction of relationships, a concrete programming process and immediate visual feedback. This led us to use general principles of multiple, separate notations with small numbers of clearly differentiated symbols; multiple levels of abstraction where appropriate; and clearly defined semantics for each notational symbol for each SDL diagram type.

5. Statistical Design Language (SDL)

Based on our findings outlined above, we developed a design for SDL that supports five diagram types, one of which supports two levels of abstraction. Figure 4 shows the relationships between the five main SDL notations. These are:

- *Survey diagrams* - provide an overview of a survey and support collaborative brain-storming of the overall survey design. The central element represents the survey, with the key survey contexts (eg the survey objectives) and the key survey attributes (eg individual objectives) linked from each survey context;
- *Survey data diagrams* - describe at two levels of abstraction the structure of survey data and operations that construct and transform this data. They are used to model datasets, metadata and data operations (e.g. sampling). The metaphor is dataflow, with data sets manipulated by sampling operations to generate sampled data sets. Data diagrams may be "drilled into" for more detailed information about their data structures.
- *Survey technique diagrams* - define in more detail a statistical technique's task sequencing /dependencies. These also use a dataflow metaphor, however, whereas survey data diagrams perform operations, such as sampling, which, in a statistician's eye, modify the data, technique diagrams describe non-modifying data analysis probes such as regression analysis or multivariate graphing.
- *Survey task diagrams* - define specific tasks to carry out for data analysis and related activities. These hierarchically represent tasks undertaken in the survey. They may be instantiated from a template representing a reusable survey design and can be viewed as a specialisation of Sutcliffe's task model [43]. Tasks may be associated with (bound to) survey artefacts such as data sets and reports.

- *Survey process diagrams* - define statistical procedures and the processes they are composed from and model the dynamic relationship between tasks. These aggregate tasks into stages and represent both flow between stages and use of task outputs as inputs to subsequent stages. Process diagrams can be "drilled into" to understand detailed relationships between a stage's tasks.

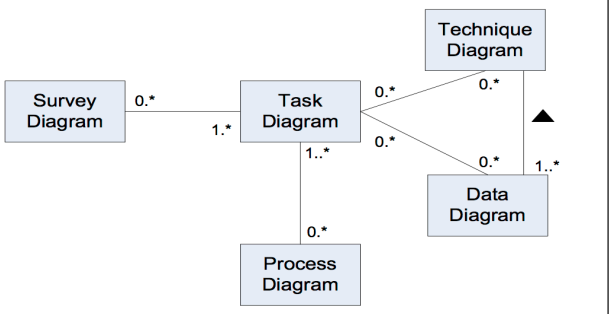


Diagram	Relationships	Detail
Survey Diagram	Task Diagram	A survey context consists of survey tasks
Task Diagram	Process Diagram Technique Diagram Data Diagram	A survey task may use technique & data diagrams. Survey tasks are survey process participants.
Technique Diagram	Task Diagram	Survey task's functional goals can be modelled in technique diagrams.
Data Diagram	Task Diagram	Survey task's functional goals can be modelled in data diagrams.
Process Diagram	Task Diagram	Tasks exist in relationship to the survey process.

Figure 4: Relationships between SDL diagrams

(© 2007 IEEE. Reprinted, with permission, from [22]**))**

There are six main design principles that shape the visual aspects of the SDL diagrams:

- Diagrams should be usable without software tooling. E.g. no elaborate layout schemes should be enforced.
- Terseness; in terms of a number of visual primitives
- Visual primitives in the context of a diagram are orthogonal and this results in having "one way of doing a specific task".
- All default colour schemes can be overridden thus no colour has any symbolic value.
- A SDL diagram can be executed and all significant run-time changes during an execution of a SDL invoke visual changes that are either change in colour or border thickness.
- The SDL diagrams that map closely to statistical computing, Survey Technique and Data diagrams make use of a data-flow metaphor and they emphasize the functional model involving datasets and statistical functions that operate on them. This accounts for lack of visual primitives that map to imperative operations.

In the following we describe each diagram type in more detail using the 2001 New Zealand Crime Victimisation Survey.

5.1 Survey diagrams

Survey diagrams provide an overview of a survey in terms of the various contexts it is organised by and the key survey attributes of those contexts. It supports interactive brainstorming to identify key aspects of a survey such as its requirements implications, analytical methodologies and time scale. Figure 2 (from Section 2) shows the survey diagram for the New Zealand Crime Victimisation Survey. The survey diagram consists of an icon representing the survey (hexagon), contexts by which that survey is organised (ovals connected to the survey) and a hierarchy of survey attributes for each context (text boxes connected in a tree by arrow connectors. Survey diagrams attempt to lower an entry barrier in terms of accessibility. This is particularly important for a survey diagram as it is expected to be used by the most diverse set of survey participants in the context of a survey process. A survey diagram has five visual primitives and has no visual primitives that represent statistical techniques. There is no diagram type like this in other existing notations e.g. UML, BPML etc.

The Survey diagram notation is summarised in Figure 5. The survey diagram in Figure 2 shows us that:

- The objective of the survey is to find out (i) crime impact on victims; (ii) key risk groups; and (iii) alternative measure of victimisation [top left of diagram];
- The survey's target population is the New Zealand public [bottom right];
- A caveat is that it can't be compared to previous surveys due to changes in method [bottom left];
- To collect data, victims are stratified by Nielsen Area Units then clustered, and samples are selected randomly from each stratum [top right]; and
- Data analysis will attempt to determine if any relationships exist between income and awareness of victimisation service, and if so they will be tested using using discriminant analysis [bottom].

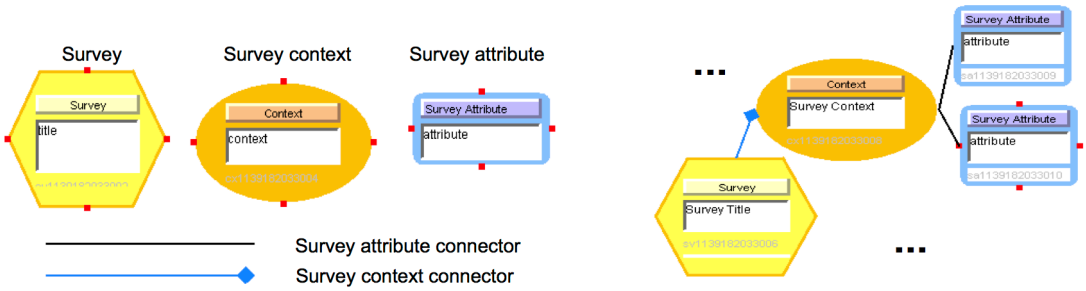


Figure 5. (Left) Survey diagram notation and (right) example usage.

A survey diagram's scope covers the entirety of a survey but expressed at a very high-level. There are no rules as to how abstract or explicitly a context is expanded. The relaxed approach has both advantages and disadvantages due to the varying levels of abstraction across contexts. This is discussed later.

5.2 Survey task diagrams

Survey task diagrams are used to abstract a survey into as a set of tasks to be carried out to realise the survey. They mirror typical knowledge-based research activities. Both visual and meta-model aspects of a survey diagram are influenced by the generalised task model developed by Sutcliffe [43]. The main visual primitives are chosen so that the breaking down of a task into sub tasks can be elicited with a simple box and arrow diagramming technique. Our statistical survey task model is a composite of discrete (or other packaged composite) tasks. This model shows how these surveying tasks are chained together via relationships to form cohesive task models. The representation of task models in task diagrams takes the form of a collection of rectangles (the survey tasks) directed connectors (the sub task connectors) and parallelograms (the survey artefacts). The hierarchical order between tasks is explicitly annotated by the use of the task connectors. We considered using adaptations of BPML process diagrams or activity diagrams from the UML. BPMN diagrams have many additional features to support much more complex business modelling domains. Unlike activity diagrams, we wanted to mix flow and hierarchy on the same diagram type and also link survey artefacts to tasks in the visual model.

The default mode of processing task diagrams is from the left to right by convention. That is sibling tasks at the left side precede those at the right side sequentially. A task is considered fulfilled only if all sub tasks accomplished their goals. The expressiveness of the task execution order is extended by the use of two types of task operators: OR and Iteration. If the OR operator is present, an upper-level task may be carried out by one of several variations. The Iteration operator signifies that a task is on-going and that it can be done in parallel to other tasks.

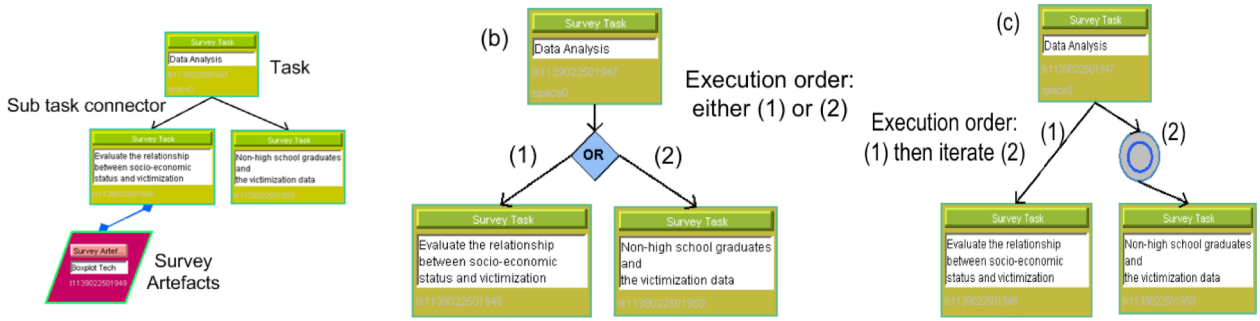


Figure 6. (a) Basic survey task diagram notation; (b) OR grouping; (c) iterative tasks.

Unlike a survey technique diagram, a survey task diagram does not have detailed specifications of its own and is not executable. For example, the iteration operator cannot be specified with a termination condition, which in consequence makes it infeasible to create an executable control flow out of it. However it is a very important diagram to co-ordinate together various aspects of statistical surveys expressed by the other three types of SDL diagrams: survey, survey data and survey technique. Survey tasks elaborate survey contexts (mapped from survey diagrams) and the survey artefacts they use map to survey datasets and survey techniques modelled by survey data diagrams and survey technique diagrams respectively. Therefore it provides an integration point for visually disparate types of SDL diagrams.

Figure 7 shows two survey task diagrams that show how two of the Crime survey contexts from Figure 2 (Data Collection and Data Analysis) map into tasks which are then decomposed into subtasks. The "Select Sampling Method" subtask highlighted in the Data Collection task decomposition is associated with a survey data diagram, as elaborated in the next section. In the Data Analysis task decomposition at right, relationships are also established to survey artefacts

corresponding to survey techniques to be used to carry out these subtasks. These survey techniques are associated with a survey technique diagram, shown in thumbnail form to the right, which elaborates how the techniques are composed together. This is explained in more detail in section 5.4.

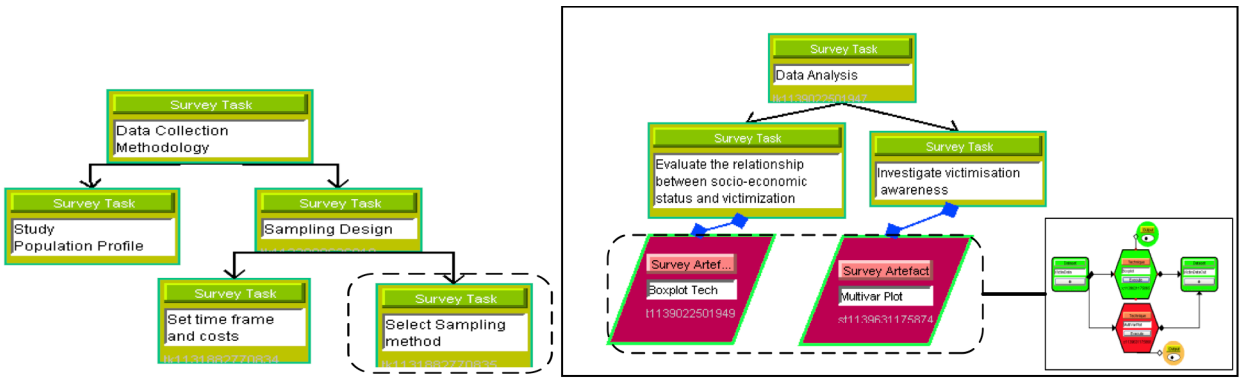


Figure 7. Basic crime survey task decompositions.

5.3 Survey data diagrams

Survey data diagrams provide a visual framework to support design of statistical sampling and data collection processes. They represent both the semi-structured data involved in the survey and data operations converting one data structure to another. The most important design aspect of the Survey Data diagram is its closeness of mapping to data collection activities from the viewpoint of a domain expert. All visual primitives of a survey data diagram map only to collected data, collection/sampling operations and output data that is filtered by such operations. Survey data diagrams can be executed and contain visual primitives and cues that are specifically designed for run-time outcomes. Survey data diagrams along with survey technique diagrams are the most immediately familiar form of SDL diagram to survey statisticians and data analysts alike, as they are familiar with data manipulation (specified in SDL data diagrams) and data processing and analysis (specified in SDL technique diagrams).

Survey data diagrams model statistical datasets, metadata and data operations (e.g. sampling). Major survey design concerns addressed by survey data diagrams include the visualisation of data flows, data operations, statistical metadata and the population to statistical data relationships. Survey data diagrams show data-level details as well as data-level operations that are usually derived from sampling techniques. In addition to showing statistical datasets using just one visual layer, survey data diagrams allow the relationship between statistical datasets and their metadata to be described using multiple diagram layers. Survey data diagrams are very different to ER model and UML class diagram data models in that they include both structural definitions and also include dataflow semantics. We chose to separate operations into standalone icons in a similar manner to other visual dataflow languages like LabView. We also chose to break out dataset structural elements as separate entities in response to user feedback that these are often critical to clearly see and work with. As survey data diagrams are translated into executable data processing services, they require sufficient dataset and operation properties to enable this service script generation.

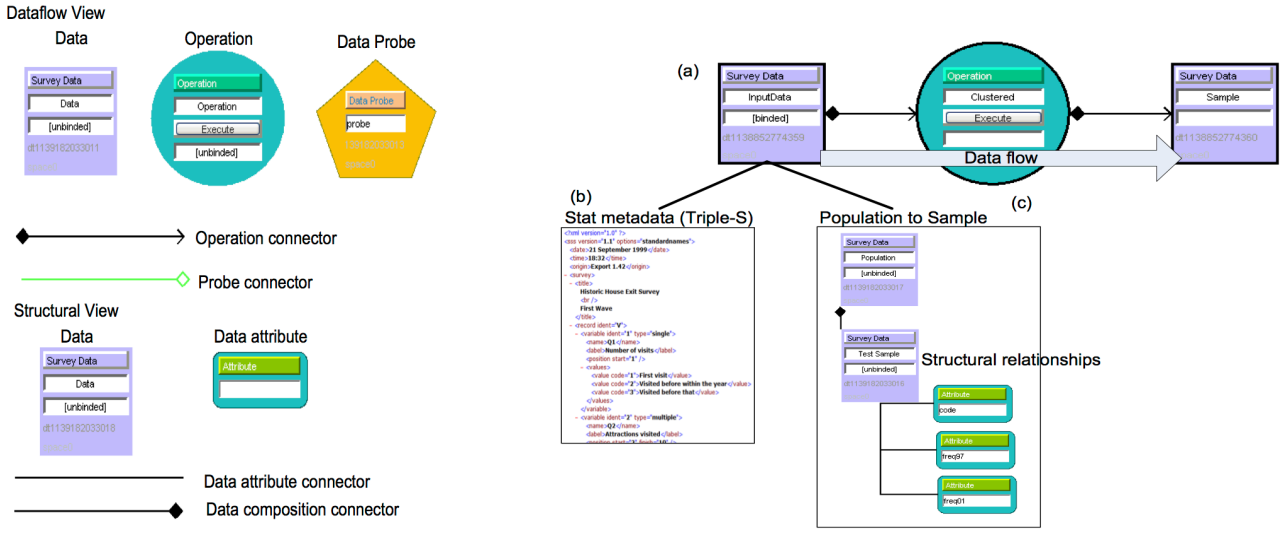


Figure 8. (Left) Survey data diagram notation and (right) example of usage in Crime survey.

Figure 8 (left) shows the main notational constructs in the Survey data diagram. The main theme of a survey data diagram is set by datasets (rectangles) and data operations (ovals). The pentagon icon represents data probing activities such as obtaining descriptive statistics on a dataset. Figure 8 (right) shows an example of the dataflow metaphor used and multi-layers of data encapsulated in data diagrams. (a) shows data described by a rectangle being processed by an operation to produce a new dataset. (b) is the underlying meta-data (Triple-S format in this example) being encapsulated. (c) shows the population we want to sample, with various features of the sample specified.

Figure 9 shows two examples of data diagrams from the crime victimisation survey design. The left side specifies the initial survey data source (Data Frame) and an initial 2-stage stratification operation (1) to produce the survey dataset (Data1). This is then further processed using a patterned clustering operation (2) to produce the final survey dataset (Sample). A mock statistical dataset can be bound to the 'Data Frame' icon for pre-testing purposes. In the post data collection stage, collected statistical data are bound to the 'Sample' icon. Shown on the right, stratified sampling produces a Sample. The next step in modelling the sampling process is to depict the sample to population relationships. This additional layer of information is associated with the survey data icon 'Sample'. A range of data attributes are specified on Region and Subject.

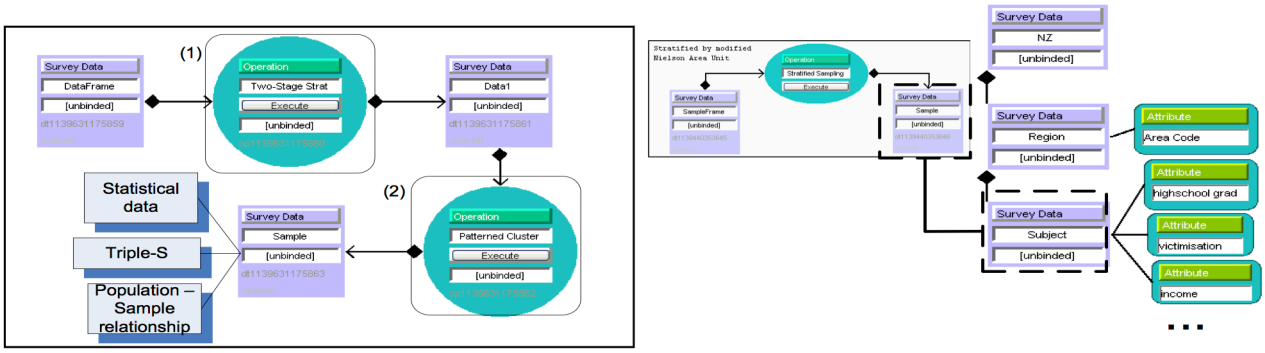


Figure 9. Two Crime survey data diagram examples.

5.4 Survey technique diagrams

Survey technique diagrams specify an individual statistical surveying technique to be used in detail. A technique corresponds to a data processing or analysis technique usually embodied in a script run by a statistical analysis tool on specified data. Often these scripts can be parameterised for reuse. The survey technique diagram notation, shown in Figure 10, reuses the data flow metaphor and the entity tree notation of the survey data diagram. It adds information on the technique (logical entity), data consumed by it (specified by input ports), and the data produced by it (specified by output ports). Data is "bound" to these "ports". The execution point (hexagon) to which the directed connector point represents the analytic technique and can be mapped to real-working version of the technique (e.g. an R procedure). All numerical dataset to dataset based flows are expressed with rounded rectangles and hexagon shapes and analytical results such as graphs are routed to oval shapes. This design choice visually annotates two types of data flows involved in using statistical techniques. One involves analytical outcomes of using a technique. Another involves piping of data from one technique to another.

While survey data and technique diagrams superficially appear similar, there are some significant differences in semantics and usage. In survey data diagrams the dataflow literally signifies the stream of datasets that are to be manipulated by sampling operations (e.g. selecting every nth data rows). This implies real physical changes in the datasets. However for survey technique diagrams no such analogy exists since generally statistical techniques imply only exploratory activities without explicit changes in datasets. Some examples of common statistical techniques are regression analysis, multivariate graphing, and multivariate data exploration. The survey technique diagram is another novel visual representation closer to visual functional languages such as Forms3 and Chameleon. However, we wanted to make the technique components clearly distinguishable i.e. data, technique, flow.

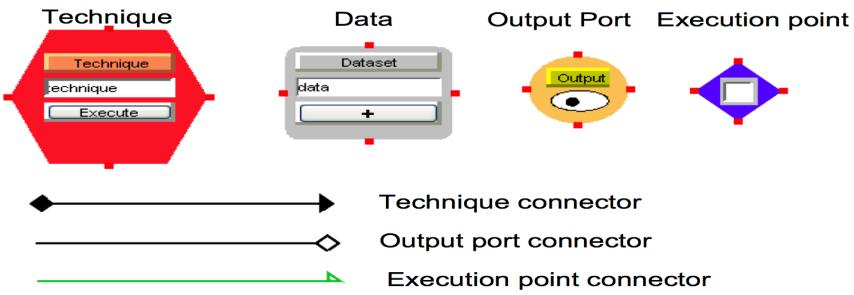


Figure 10. Statistical technique diagram notation.

Figure 11 shows two examples of the application of the survey technique diagram in the crime victimisation survey. Suppose we are interested in the level of education and crime victimisation in our crime survey example. To investigate the strength of association between two data variables, the use of the chi-squared approximation is chosen, as modelled via the left hand example SDL survey technique diagram. This has the sampled subjects as an input to the statistical technique 'ChiSq' and its output will decide the validity of the association. The ChiSq technique's visual and textual outputs are directed to the output port and the data icon at the end of the dataflow is bound to the resulting data and metadata produced by the chi-square test.

Two of the goals in the data analysis stage for the crime victimisation survey are to: (1) investigate the relationship between socio economic status and victimisation; and (2) investigate the public's awareness of victim support services and self-assessed safety. The collected data is published into a shared data repository and needs to be processed by two statistical techniques, boxplot and multivarplot, as shown in the survey task diagram Figure 7 (right) above. In this case study, the statistician wants to utilise visualisation methods to assess whether there is evidence of a statistical association between data variables. The two techniques are specified in the same diagram, shown in Figure 11 (right), as they consume the same statistical dataset.

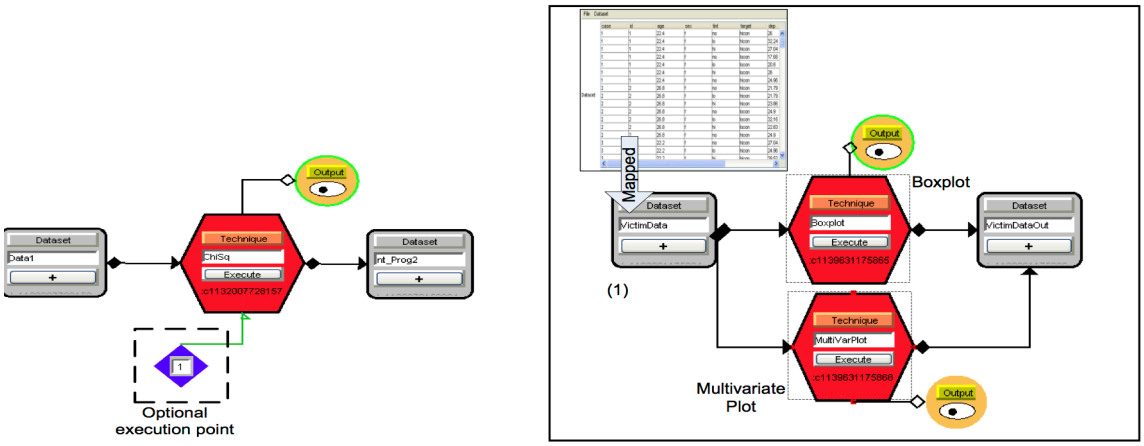


Figure 11. (left) Simple chi-square technique and (2) survey technique diagram for discriminant test.

5.5 Survey process diagrams

Having defined the high level overview of a survey and the data to be collected, the next step is to describe the analyses to be applied to the collected data using survey process diagrams.

A survey process diagram is used to denote various stages involved in a survey process and what tasks exist at each stage. Survey task diagrams manage each task independently but in survey process diagrams survey task to task relationship are expressed and their involvements in each stage is shown. Rectangular icons represent stages and ovals represent tasks to be carried out. Stage to stage flow is represented by an arrow connector. We chose to use a containment model and layering, very different from e.g. UML activity diagrams and BPMN process diagrams. We did this as we wanted to allow survey designers to indicate, in a summarised form, key sub-task diagrams, how these are ordered, and key survey artefacts consumed, manipulated and produced. Our survey process diagrams use stages in a similar manner to these other process-oriented visual languages, but incorporate task and artefact relationships.

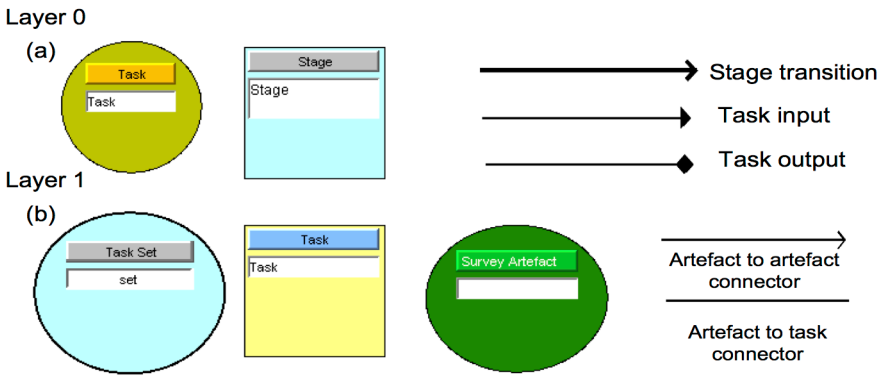


Figure 12. Survey process diagram notation.

A survey process diagram for our crime victimisation example is depicted diagrammatically in Figure 13. This survey process diagram describes two the main stages involved in the survey process: sampling design (stage 1) and data analysis (stage 2). It should be noted that the descriptions for the stages are much simplified and generalised for this paper. Our example involves three tasks: (1) Population study: The population profile is studied to formulate data collection strategies which apply to the sampling design decisions; (2) Sampling design: Sampling method is chosen to meet the required precision of the survey and a trade-off between cost and precision; and (3) Data analysis: To explore the association between the level of education and the crime victimisation. The last two tasks are related to each other in the same task model and the two stages are sequentially linked. This linkage is explored in more detail in the Layer 1 diagram below, which describes the stage tasks in more detail and shows how task(2) provides data for task(3) to consume.

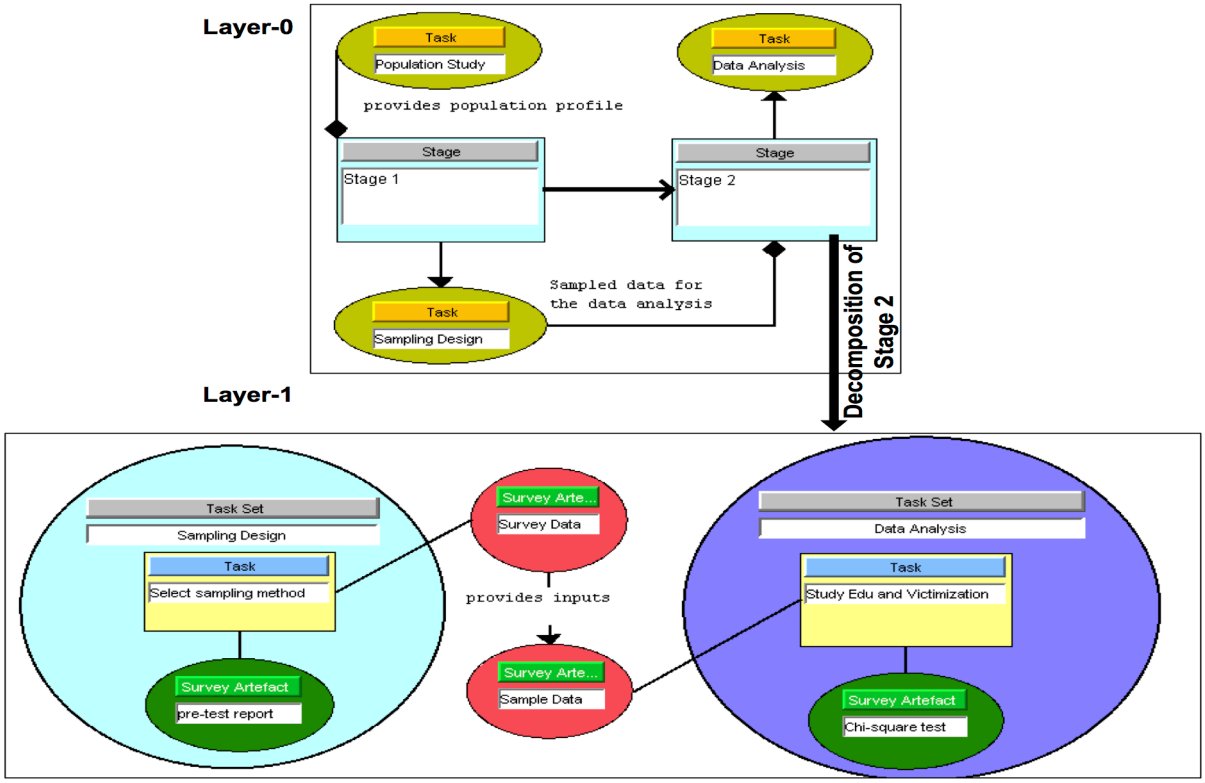


Figure 13. Survey process diagram for the crime victimisation survey.

6. SDLTool Example Usage

We have built a support environment for SDL– called SDLTool. This provides authoring support for SDL, support for importing statistical meta-data, binding data to techniques and data elements, visualising generated statistical data and technique analysis results, generation of reusable web services embodying specified techniques, and a model repository including reuse tools. In this section we use the New Zealand Crime Survey case study to elaborate the steps outlined in Section 4 when using our SDLTool to model the survey, test and visualise survey components, and generate reusable web services for techniques. In subsequent sections we outline the development of SDLTool from conceptual model to implementation.

Initially a statistician designs a survey using the top-level SDL survey diagram, an example of which is shown in Figure 2. The statistician may begin from scratch or reuse a design from an SDLTool repository. For example, in Figure 1, the *Survey* icon plus 4 contexts *Objectives*, *Data Collection*, *Population* and *Data Analysis* form a standard template that has been elaborated with the survey name, an additional context *Caveat*, and survey specific survey attributes for each context.

After creating the overall survey structure the statistician creates further SDL views to specify the survey in more detail. Several examples from the NZ Crime Victimisation Survey are shown in Figure 14 (some parts of this survey design have been simplified for the purpose of this paper). Figure 14 (a) is a hierarchical task diagram, specifying two data analysis tasks to be carried out on the survey data. During data collection, our main concern is to specify sampling techniques used in the survey process and types of statistical metadata related to collected data. Figure 14 (b) specifies the two sampling methods to be used in the survey. Here the sampling frame is stratified in two stages by the modified area unit (1) then household visits planned using patterned clustering (2). A mock statistical dataset can be bound to the 'Data Frame' icon for pre-testing purposes. Post data collection, collected data can be bound to the 'Sample' data icon. Figure 14 (c) shows a survey technique diagram describing the data analysis operations implementing the tasks in Figure 14 (a). Here, we use two visualisation methods (boxplot and multivariate analysis) to assess whether there is evidence of a statistical association between data variables.

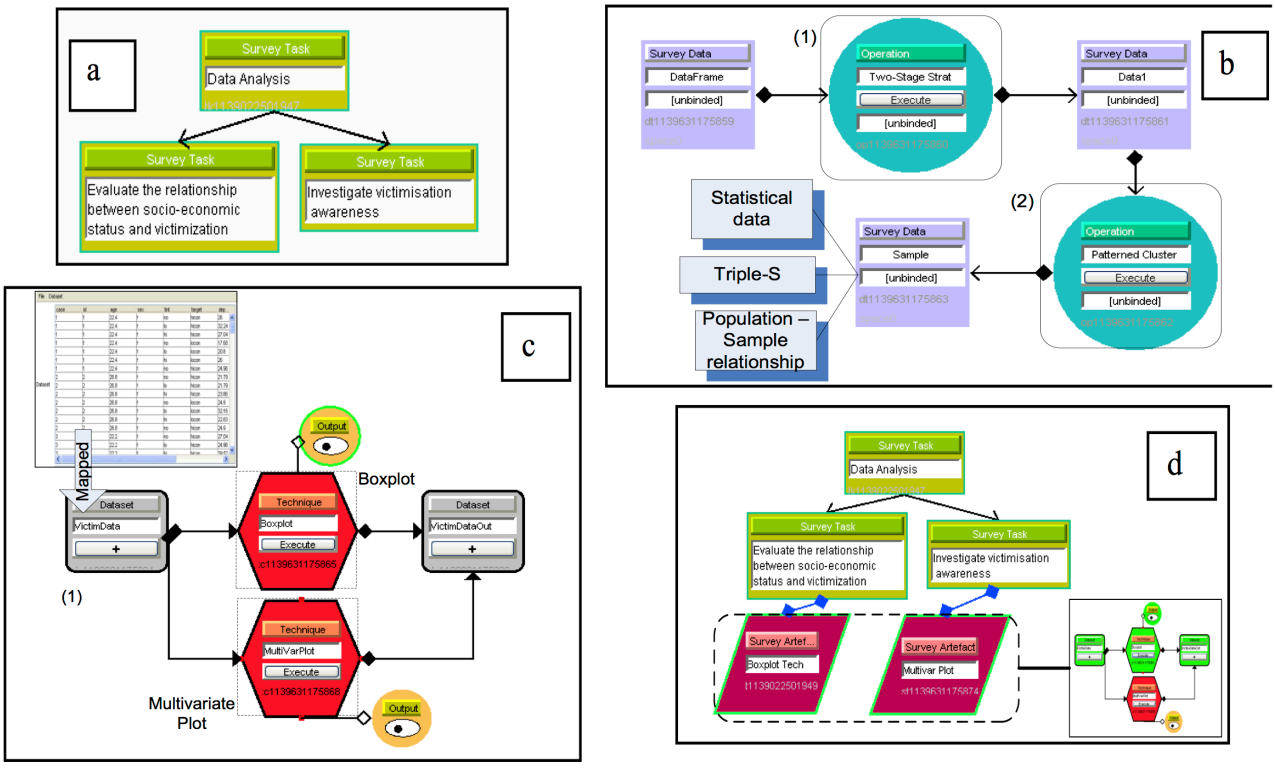


Figure 14. Examples of NZ Crime Survey SDL diagrams.

The two techniques are composed in the same diagram as they use the same statistical data. They are bound to external analysis methods implemented using R. The diagram may then be separately executed to produce a boxplot for the socio economic status and victimisation relationship and the multivariate plot to visualise the public awareness. The modeller also binds the resulting visualisation artefacts to the tasks they correspond to in the task diagram (Figure 14(d)).

6.1 Binding artefacts

Binding operations map SDL icons to various external resources (Figure 15). Menus associated with graphical icons (a,b) initiate resource mappings (dataset and technique respectively). Mapping forms are generated for the user to complete (c,d), with many fields automatically filled, inferred from dataflow and other bindings (maintaining inter diagram consistency). Aspects of the icon related to the mapping appear in separate view tabs where the user can view or edit the resource, eg a data structure (e), data set (f) or metadata description (g). A fully mapped SDL icon has a thicker border indicating readiness to execute.

6.2 Execution

When a survey data or technique diagram is mapped to all required statistical resources the user can execute it in real-time and explore its textual and visual outcomes. Figure 16 shows a typical diagram execution sequence. A completed survey technique diagram has all required graphical icons mapped to appropriate resources, indicated by the thicker icon borders (a). Each technique or data operation icon has a button interface used to execute the specification (with icon colour change indicating successful execution). Following execution, the user may select the output port of the executed technique or operation and probe into available outcomes. Figure 16 (b) shows detailed results of running the multiple linear regression technique on the dataset when double-clicking on the top technique diagram output icon. Modifiying the dataset or technique parameters and re-running will update the results shown. Figure 16 (c-f) show various SDLTool built-in visualization support techniques (c-e) and external visualization tools (f) invoked on the output of the MRPairPlot technique. The survey designer can select the desired visualization technique via pop-up menu by right-clicking on the lower output icon in the technique diagram. The visualization technique/tool is invoked and results displayed. Updating the source dataset and/or technique parameters result in internal SDLTool visualisations being immediately updated. External tools are re-invoked and their generated visualisations shown on SDLTool user request. Documentation is also important in managing and reusing the survey process. SDLTool can generate documentation in a form accessible to third-party tools or clients allowing them to understand the semantics of our visual diagrams without the need for the prototype tool. Figure 16 (g, h) show generated HTML documents describing a dataset and statistical technique respectively, as specified in Figure 16 (a).

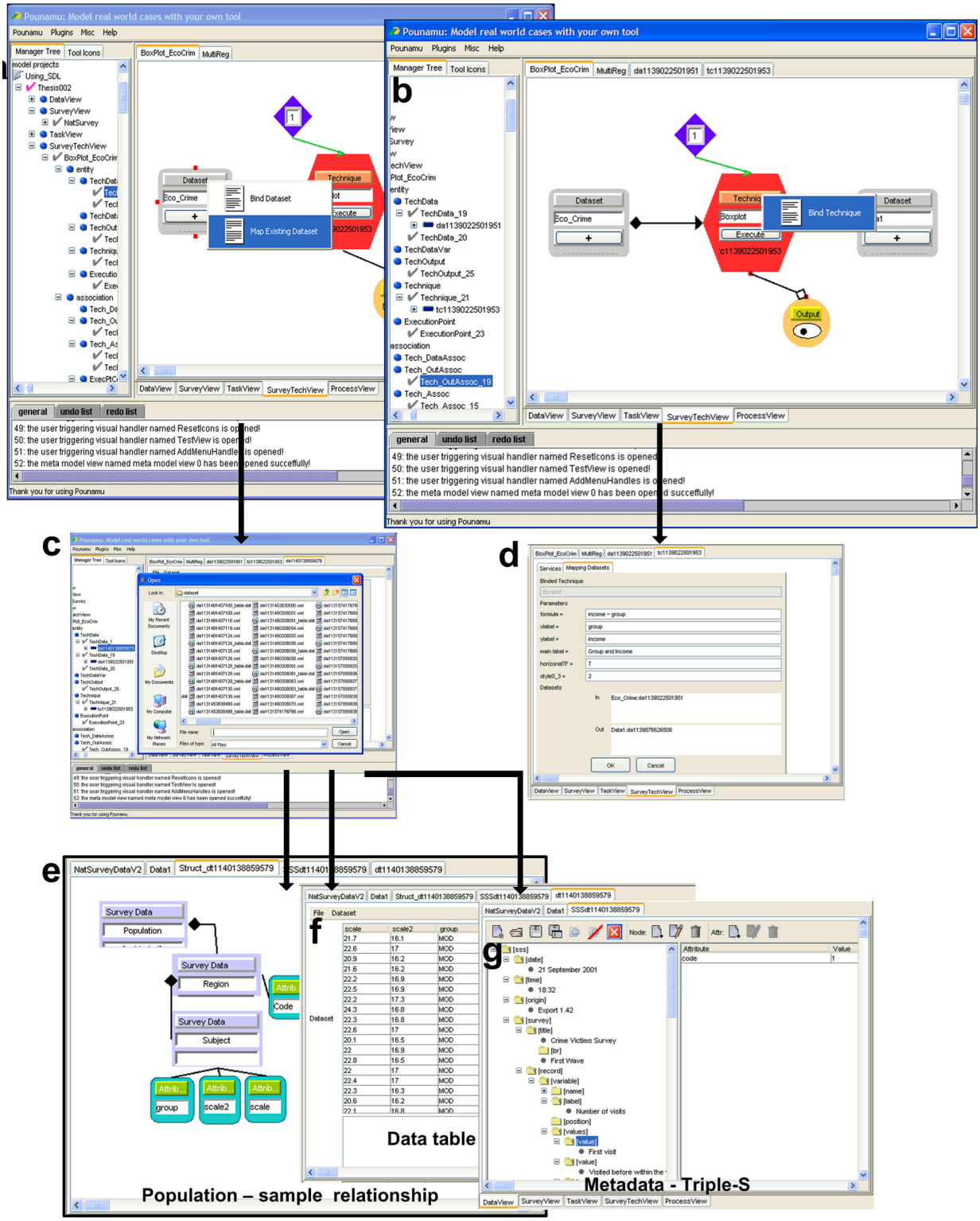


Figure 15: Binding statistical resources to SDL elements

(© 2007 IEEE. Reprinted, with permission, from [22])

All of these activities can be done from within the SDL Tool environment. This thus provides an integrated design, coordination, and implementation environment for complex statistical survey design. The environment is live, providing good progressive evaluation support, with the ability to try out techniques and operations as they are defined and bound to appropriate resources.

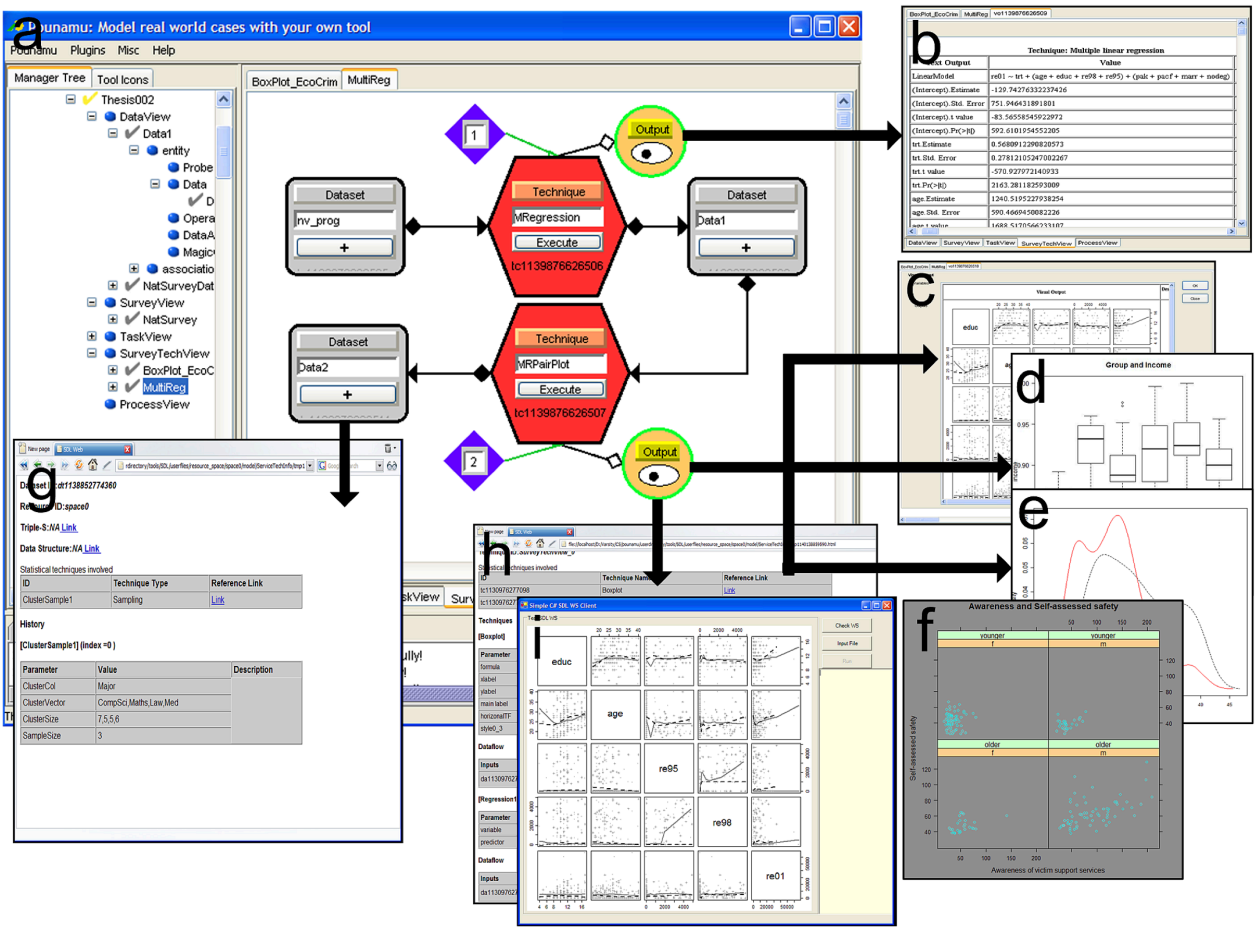


Figure 16: Examples of execution of statistical technique diagrams including results visualization

(© 2007 IEEE. Reprinted, with permission, from [22])

6.3 Web Service Generation

When the user is satisfied with the correctness of a functional technique diagram the diagram can be turned into Java code and exposed in the form of a web service. This eliminates the need to have the SDLTool environment and associated 3rd party statistical analysis packages in order to understand the survey and promulgate its results. To do this, a web service generation template is used to process technique diagrams with generated code stored in a web services repository. Clients can access the service specification via a WSDL interface. Exposure as a web service provides platform independent access to survey results. For example, Figure 16 (i) shows a .NET program using the java-implemented statistical technique of Figure 16 (a) via its web service interface.

The first step in implementing technique web service generation is to create a diagram processing sequence. The implementation of the process examines the underlying diagram structure for execution points according to the diagram's syntactic rules. Once the sequence is complete, metamodel files bound to visual icons are accessed by the service specification generator to create web service specifications. The specifications file encapsulates (i) Dataflows; Techniques (identified by unique identifiers and types); and Parameters assigned to techniques. Once the specification file is ready, the code to execute the techniques according to the defined sequence in the specification is generated. The generation process starts with a technique model definition that describes the data flow from one technique to another, types of techniques used, expected outputs, etc. Technique entities play a central role in code generation. Each technique is evaluated in the order of its assigned execution index starting from 1 and then the code template is used to build up code in Java. The code generation process entails no complex operations due to the autonomous nature of all techniques present in the survey technique diagram. Thus a simple template driven sequence can be applied to the first technique until the last one.

Figure 17 (a) shows the user invoking the contextual menu to generate a service based on the diagrammatic specification of the current diagram. The diagram is submitted to the SDLTool model repository as an XML document as in Figure 17 (b). The submitted model is processed according to a web service generation template and clients may access the generated web service via a generated WSDL interface, shown in Figure 17 (c). Figure 17 (d) shows a demonstration analysis and visualisation application that we built using the Microsoft .NET framework utilising the statistical technique via its web service developed and generated using our SDLTool.

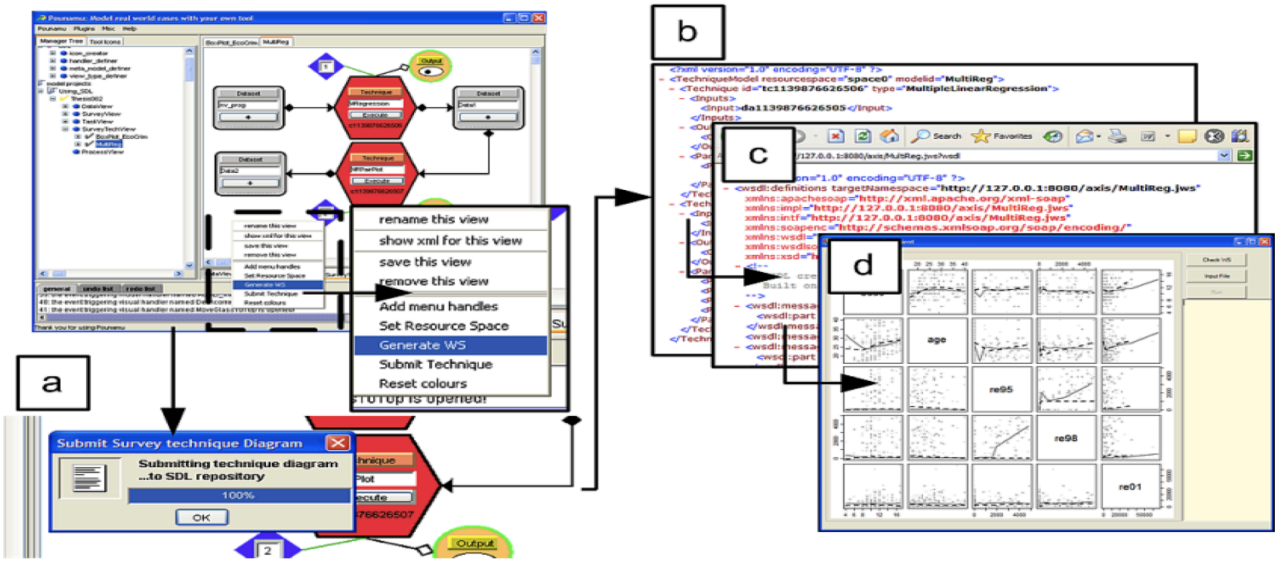


Figure 17. Generating and using a statistical survey technique implementation and its web service interface (© 2007 IEEE. Reprinted, with permission, from [22])

7. SDLTool Meta-model and Semantic Layer

As a first step towards realising SDLTool, we developed a detailed SDL meta-model [20]. This was done by a process of abstraction, classification, and generalisation on the collection of SDL visual models in a similar fashion to Model-Driven Architecture [30]. To integrate the SDL model as expressed by multiple SDL diagram types we needed to understand the relationship between elements in each diagram. Figure 4 showed the high level relationship between the diagrams. At a finer grained level, our approach to integrate the five diagram meta-models uses two main constructs: inter-diagram relationships and survey entities. Inter-diagram relationships lay down a broad inter-diagram network and survey entities belong to the network of the diagrams. A survey entity can be defined as follows:

- A survey entity can be a survey task, dataset, technique or non-connector graphical entity in the visual environment.
- A survey entity has its own unique identity though they may appear non-distinguishable visually. E.g. two dataset icons look the same but they are mapped to two unique survey entities.
- A survey entity belongs to at least one visual model.
- Inter-diagram relationships positively imply the existence of at least one survey entity that is shared by more than one diagram. In other words, a shared survey entity completes an inter-diagram relationship.

Survey entities thus provide the basis for both model integration (in the meta modelling process) and inter diagram consistency (at an operational level). The overlapping survey entities act as integration points to merge related views for a target domain, as visualised by Venn diagrams in Figure 18 (left). Users or tools can transverse the network of related diagrams by using overlapping survey entities as entry/exit points.

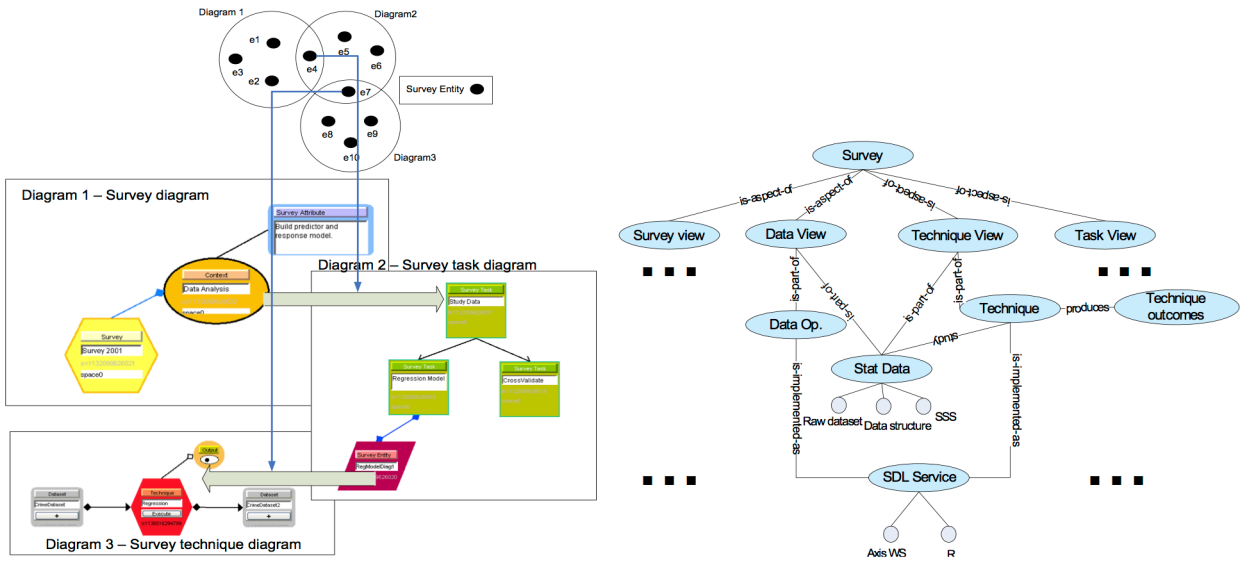


Figure 18: (left) Inter diagram mappings and (right) partial topic-map representation of SDL

(© 2007 IEEE. Reprinted, with permission, from [22])

The necessity for an additional semantic layer comes from SDL tool support. SDL diagrams as communication media from the perspective of human users require no explicit semantic support. However SDL diagrammatic notations anticipated the development of supporting tools and a model-based approach for generating services. Our current proof-of-concept tools do not fully utilise the semantic layer, relying instead on static rule based reasoning to infer model semantics from the metamodel layer. However the semantic layer will be the primary tool for extending the language base of SDL thus we include a brief description here for completeness.

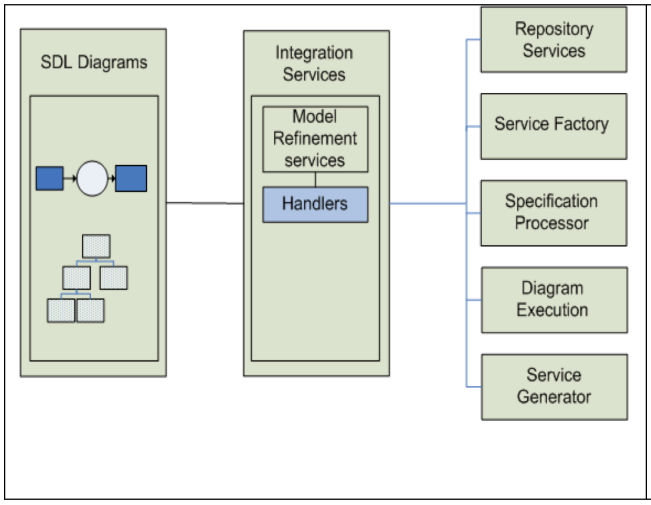
The SDL semantic layer is primarily organised by *topic maps* [37]. Topic maps offer heterogeneous information repositories [3] to tie the underlying semantic of the metamodel to real world statistical survey topics. The primary constructs in SDL diagrams such as dataset entities are organised into topics and they are explicitly related to the overall conceptual structure of SDL by means of association and instance membership (see Figure 18 (right)). The ontology layer consists of taxonomies of statistical techniques, metamodel structures and relational templates to bind the metamodel to the semantic layer.

If all important aspects and diagrammatic notations of SDL are considered to be "topics", SDL diagrams are "occurrences" and all inter-diagram relationships are "associations" then we can visualise topic maps as providing a unique platform to express the semantic layer. It is interesting to note that just as we can transverse from the visual layer to the semantic layer, reversing the process by having the visualisation of the topic map as a starting point could potentially be used as the basis for making SDL an extremely extensible visual language.

The semantic layer creates a structural ontology for SDL. Hence the structural ontology also provides a template for which mapping operations to bind external resources (occurrences) with SDL meta-model entities. One such mapping operation is to turn static visual icons into dynamic ones, that is to give the icons the behavioural and functional nature of a widget, and which can then serve as a dynamic interface to control external resources. The icon-to-widget mappings for SDL tool support arise out of the need to support occurrences associated with a topic node at the tool level. Thus the semantic layer also provides a new perspective in looking at visual tool support from the modelled ontology.

8. Architecture and Implementation

Our SDL tools use an event-driven, loosely-coupled architecture as outlined in Figure 19. At left, SDL diagrams are represented as diagram data (or "views", diagram left) and a shared repository (or "model", diagram middle) following the Model-View-Controller paradigm. A set of extensible components (diagram right) are used to provide repository support, external tool integration, model compilation, execution engine, and a web service generator to make SDL designs accessible to other users. Brief descriptions of each of these components can also be seen (table right).



Component	*Description*
Handlers	The first component to handle diagram editor events and relay them to appropriate components when it is required.
Model Refinement Services	Consist of server services to reflect changes in the visual layer by creating/updating the metamodel layer.
Repository Service	Registrar of SDL service (data operations and techniques) metadata.
Service Factory	Provides a proxy object to enable communication with a remote service.
Specification Processor	Translate the underlying diagrammatic model into execution-ready specifications.
Diagram Execution	Executes provided specifications.
Service Generator	Generates diagrammatic specifications into inter-operable services (e.g. Axis hosted web services)

Figure 19: SDL tool architecture

(© 2007 IEEE. Reprinted, with permission, from [22])

Figure 20 shows how events are used to couple the components in our architecture. Changes to diagrams are sent as event notifications to an "event handler" for the diagram (1). This passes the event onto a model refinement service (2), which propagates the event to components subscribing to the diagram change type e.g. the Repository Service. The Repository Service translates SDL data into the dataset file format and updates this (3). A response event is generated by the service to indicate success or failure. This event is processed by the model refinement service (4) to determine if any updates to the shared SDL model are necessary. If so, these are applied to the model (5). Such changes may mean other SDL diagrams sharing the changed model data need updating (6). This event-based notification mechanism provides incremental multi-view consistency, persistent repository support, compilation of SDL models, an execution engine for diagrams, and an external tool integration platform.

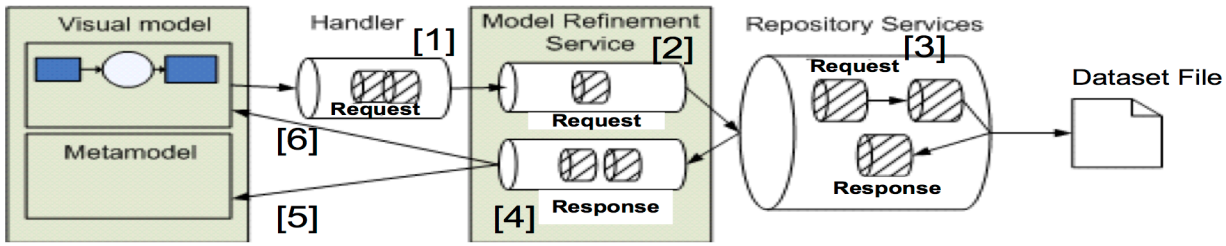


Figure 20. SDL toolset analysis pipe-line

(© 2007 IEEE. Reprinted, with permission, from [22])

We used our Pounamu meta-tool [46] to implement our SDL multiple-view design tools. All of the SDL diagrams in this paper were generated from screen dumps from these editors. Pounamu allows multi-view, multi-notation diagramming tools to be quickly specified using meta-model, view type, shape definer and event definer meta-tools. Its support for "on the fly" changes to tools allowed us to readily experiment with notational elements and diagram types and modify them at low cost. SDL was specified as a canonical Pounamu meta-model and a set of view types, one for each different SDL visual notation. Each view type (diagramming specification) has its own set of shapes, connectors and editing constraints. The SDL diagram editors are realised by Pounamu interpreting the SDL tool specifications to provide multi-view and multi-user diagram editors with a shared model. The Pounamu model produced by the SDL diagramming tools is used to provide a data-oriented integration platform to external statistical analysis tools. The Pounamu SDL shared model is "walked" by Pounamu "event handlers" developed specifically for each external tool, transforming it into a format understood by that tool and the tool is invoked via service factory components. Handlers also provide presentation and control integration for external tools allowing results to be displayed in the SDL tool.

Figure 21 shows a more detailed set of components making up the SDLTool architecture. When using SDLTool, a meta-tool specification is opened by Pounamu (a) and SDLTool is instantiated. Statisticians build SDL models in SDLTool, having diagram and model information stored locally (b). SDLTool uses an event-driven (c) generator to turn technique diagrams into scripts to drive external 3rd party statistical analysis tools (d). SDLTool may invoke these

scripts directly with example data sets, allowing testing of a technique implementation (e). The technique web service generator produces a web service providing an interface to the generated scripts for the 3rd party tool analysis and a WSDL specification for this interface (f). The generated technique web services are deployed (g) for remote discovery and invocation. 3rd party client applications may (h) discover and invoke (i) the generated technique web service to make use of its statistical survey technique implementation (j). Datasets are passed via the web service to the 3rd party statistical analysis packages (k), processed, and results returned (l) to the invoking 3rd party applications.

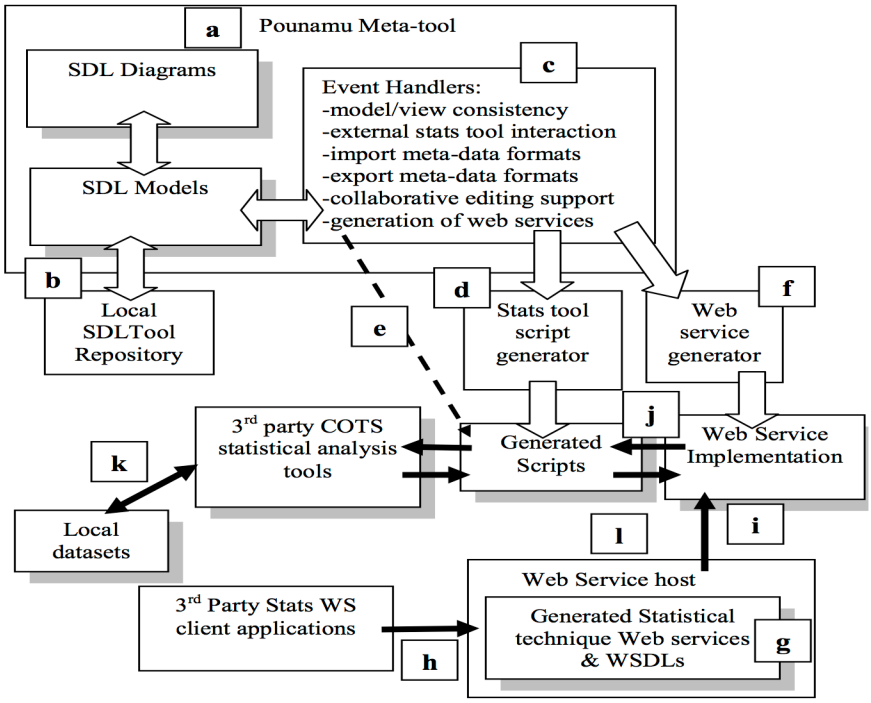


Figure 21. SDLTool architecture (© 2007 IEEE. Reprinted, with permission, from [22]).

The script generator takes a survey technique diagram and generates one or more scripts for 3rd party statistical analysis packages like R. These are "technique implementations" that given suitable input datasets will clean, select, analyse and produce an output datum or dataset for the specified technique algorithms. The web service implementation generator produces a Java implementation of the web service that is used to receive datasets as XML messages, transform them into local data files in appropriate 3rd party tool proprietary data formats, invoke the 3rd party statistical analysis package with appropriate generated technique script implementation, and transform any result data from the 3rd party tool's proprietary format and back into an XML message.

9. Evaluation

Recall our key research question from Section 2:

RQ1: Can visual notations with appropriate software tool support be used by end users, i.e. statisticians and other survey developers, to facilitate statistical survey design and implementation?

To help answer this question we undertook two cognitive walkthrough-based [17] evaluations, primarily targeted at the "design" aspect of RQ1, and a third evaluation where we designed and built several statistical surveying techniques with SDLTool and generated web services to provide access to these, targeted at the "implementation" aspect of RQ1. We evaluated the services by using them within other statistical surveys. For the two cognitive walkthrough studies the theoretical testing basis for the evaluation was the cognitive dimensions framework [16]. In the following, italics are used in the text body to highlight specific cognitive dimensions.

For the initial cognitive walkthrough we used a doctoral student in statistics as our test subject. The aim here was to provide a preliminary answer to RQ1 with a near final version of toolset to help shape a final iteration of the language and tool design and to act as a pilot to help design the subsequent evaluations. This user was an expert end user of statistical survey designs and techniques.

The second cognitive walkthrough evaluation provided a more formal usability assessment of SDLTool using eight test subjects, some very knowledgeable in statistics and some with basic statistical surveying and analysis knowledge.

9.1 Pilot cognitive walkthrough

Methodology

In the initial cognitive walkthrough study we observed a doctoral student in statistics making use of SDL and SDLTool to design a moderately complex statistical survey. We chose to use such a test subject as they are very knowledgeable about existing statistical processes and tools, are an expert user of the range of features of SDL and SDLTool, and could give us detailed and expert feedback during each step of the walkthrough on their thinking and tasks. We used an earlier version of SDL and SDLTool than described in this paper for this cognitive walkthrough, with findings from this leading to several visual language and tool enhancements.

Our test plan for examining SDL using such a cognitive walkthrough approach comprised the following elements:

• Overview of SDL - the test subject was briefly introduced to SDL and working examples explained to observe SDL in action.

• Users Tasks - a list of tasks to be performed by the test subject was given. As the cognitive walkthrough approach focuses on user–oriented solution finding, the tasks were intended to give the test subject opportunities for a self-initiated exploratory path to complete the tasks. Thus the tasks attempted to simulate the cognitive context of a survey researcher in practice rather than imposing fine-grained questions. Three tasks were given to the test subject, all designed to model real-world problems. The first was to request the subject to design a survey diagram for a large-scale UK government sponsored labour force survey [34]. In the second task the subject was given a survey data diagram and requested to explain it and comment on the information represented. The third task involved the subject designing a survey of his choice from scratch.

• User Awareness- a well-designed visual language should give users the sense of self-awareness. In other words, users should be able to tell whether they on the right track in terms of meeting final goals during the course of the using SDL. Insufficient user awareness can especially impact the usability of the diagrams, which are affected greatly by local changes, as late changes could imply a significant overhaul. Therefore the evaluation of SDL looked into not only the final results but also user awareness throughout the testing session.

Results

The test subject successfully composed a survey diagram, shown in Figure 22 (left) after a brief introduction and with no interventions. The subject quickly identified survey contexts and added the key survey attributes to each context. The subject found the notation intuitive and easy to use. Only one small fault was identified in the diagram produced. The attribute in the bottom right corner should have been directly connected to the *Subjects* context rather than to the other attribute as it is not an extension of that attribute.

For Task 2, our subject was initially given the survey data diagram of Figure 22 (right) This used an early version of the notation where sampling operations were linked back to the original data structure. From a data type perspective this makes sense, but was found to be confusing by the test subject. This led to the revision shown in Fig. 5, where operations on a data structure results in a new data structure, which should be less confusing for non programmers. However, we have retained the self-reference type representation as a convenient shorthand for complex diagrams with the knowledge that this abstraction requires significant learning.

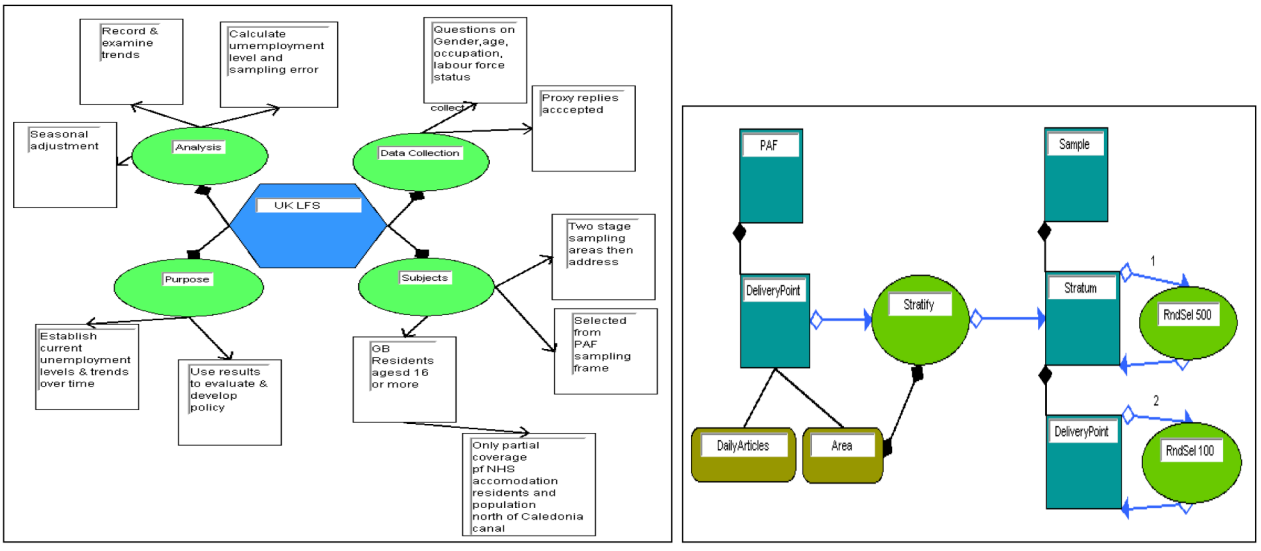


Figure 22. (Left) Survey diagram generated by test subject and (right) Initial survey data diagram given to subject

(© 2005 IEEE. Reprinted, with permission, from [23])

After the first two tasks our subject was asked to derive a survey design from scratch. No specific guidelines were set. The subject chose an analytical survey design based on a harvest estimation survey that he regarded as typical of analytical agricultural surveys. The core operational details of the survey were discussed and immediately those details described in SDL diagrams. The survey process included many iterative decision-making components, but our subject found these to be readily represented in SDL. Figure 23 shows the survey process diagram created during the test session. The diagram demonstrates the advantages of SDL in easily turning large amounts of survey design information into a comprehensible visual model that is both clear and has explicit semantics.

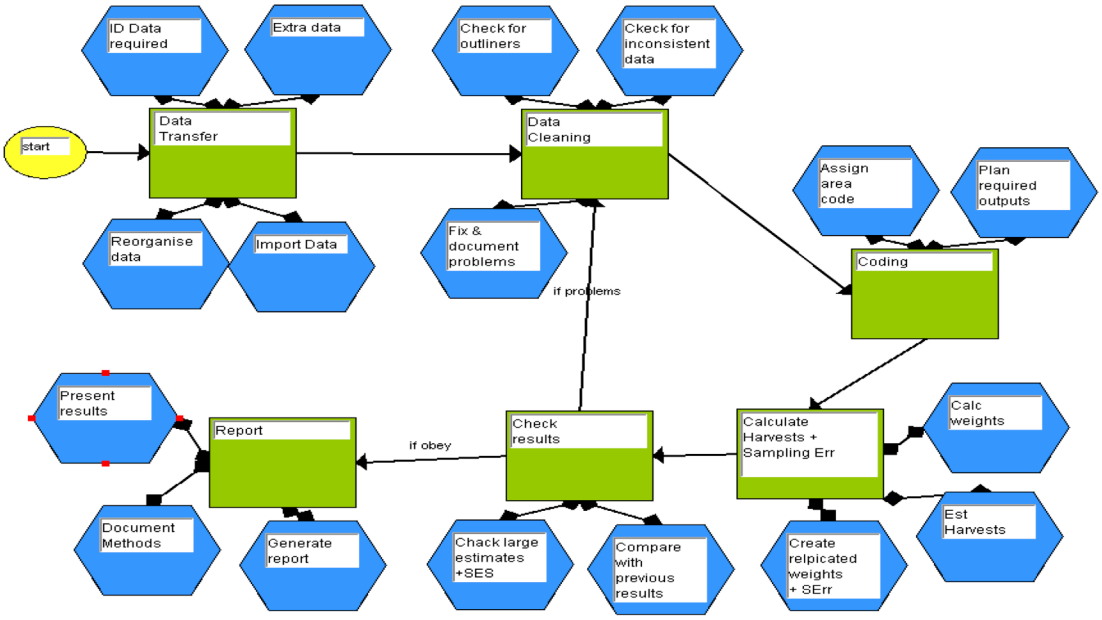


Figure 23. Survey process diagram generated by test subject for harvest estimation survey

(© 2005 IEEE. Reprinted, with permission, from [23])

Interpretation

The test subject readily and capably applied SDL for modelling the survey in a short space of time and was able to follow rules for association and visual symbols, not repeating the earlier error of misplacing survey attributes supporting an affirmative answer to RQ1. SDL provided a compact description of the survey design describing the activities, including events or items that are required in the execution of the survey process, and associations and revealed the overall purpose and structure of the survey. SDL has a number of limitations identified by the test subject. All of these originated from diagram layout concerns. One example is the survey process diagram's tendency to spread out over a large space. It can be more than an aesthetic problem as the spread-out look may slow down a user's comprehension by presenting more visual components than a single visual scan can perceive. One quick solution to resolve the visibility problems is to use multiple views, with each view elaborating on only a limited number of contexts. However, this introduces *hidden dependencies* and *juxtaposition* issues. Another would be making the processes elidable and to build an intelligent layout algorithm to shape the entire diagram to optimise visibility. Future work with Pounamu, used to build SDLTool, has included some improved reusable layout support, motivated by SDL and other DSVL layout problems.

9.2 Formal User Evaluation

To investigate RQ1 from a target end user's perspective, we carried out a further study with 3rd Year Statistics student participants. The testing session incorporated a cognitive walkthrough approach to gather qualitative data [17], together with a survey instrument to obtain user perceptions. Task completion data were also gathered.

Methodology

Test subjects were recruited from final year undergraduate statistics students at the University of Auckland. They were chosen out of a potential candidate pool resulting from recommendations from a tutor in our Department of Statistics.

All test subjects were invited to attend an introductory meeting that was designed to convey some of SDL's core concepts and to provide a basic tutorial on SDL and SDLTool.

A wide range of user activities were conducted in each testing session: pre- and post-demonstration interviews; diagram comprehension exercises; survey technique implementations using provided software tools; and comparative evaluations of SDL and existing statistical survey support tools. Two types of testing outcome were compiled: user-completed questionnaires including user perceptions, opinions and satisfaction regarding the SDL tools and their own performance; and investigator recorded performance evaluations, including task correctness, mistakes, and time to complete given tasks. We ranked performance of each task and subtask using a combination of these measures.

The key tasks asked of subjects were:

- Task 1 – SDL and SDLTool preview and tutorial (instructor led);
- Task 2 – SDL diagram comprehension: survey diagram, survey task diagram, survey data diagram, survey technique diagram and mapping of SDL diagram components to external data and services;
- Task 3 – SDL diagram and service construction: survey diagram, task diagram, survey technique diagrams; web service generation and testing

The questions asked of subjects were:

- Is it easy to model the survey with the tool?
- What does the tool do well in your opinion?
- Are there any notations that should be made clearer for the user? How? (e.g. more user interaction behaviours, colour codes)
- Is there anything the tool does not let you do that you would like to?
- What other information of views of information should the tool display?
- Are there any improvements you would like to see in the tool?
- Would you like to use the notations if you had a large survey project to do? Why/Why not?
- Are there any concepts/ideas in the SDL diagrams that are difficult to comprehend?
- Is there anything the diagram does not let you understand that you would like to?
- What other information or views of information should the diagram convey?
- Are there any improvements you would like to see in the diagram?
- Any general comments you have on the concepts of the tool and notations.

The introductory SDL diagrams used in this experiment were based on the 2001 New Zealand Crime Victims Survey to simulate real-life survey communication problems. Test subjects were asked to explain the semantics of presented diagrams and to give feedback on their effectiveness, expressiveness, usefulness and usability. Activities utilizing the software tools were to implement survey techniques to produce solutions for given Crime Victims Survey scenarios. The scenarios were designed to permit user-initiated actions and task execution.

Results

Eight test subjects, all final year undergraduate statistics students at the University of Auckland, participated in the user testing. All were new to the concept of a visual environment for statistical surveys but all had good working knowledge of statistical packages such as R and SAS, survey theory and design in academic or commercial settings. Three participants were competent programmers of general purpose high-level languages such as Java and C

Figure 24 shows the results of Task 2 – diagram comprehension. Performance of each task was ranked by a combination of user-reported feedback and experimenter (the first author) observation. Figure 25 shows the results of Task 3 – diagram and service construction. Performance of each task was ranked by a combination of user-reported feedback and experimenter (the first author) observation.

Task 2 - Diagram comprehension

#	Survey Diagram	Task	Data	Technique	mappings	Average	Rounded
1	2	3	2	2	2	2.2	2
2	3	3	3	3	3	3	3
3	3	2	3	3	2	2.6	3
4	1	1	2	3	1	1.6	2
5	3	3	3	3	2	2.8	3
6	1	1	2	3	1	1.6	1
7	3	3	3	3	3	3	3
8	3	2	3	3	3	2.8	3
Average by diagram	2.375	2.25	2.625	2.875	2.125		

0=Failed, 1=Average, 2=Good, 3=Excellent

Survey Diagram	Survey Diagram
Task	Survey Task Diagram
Data	Survey Data Diagram
Technique	Survey Technique Diagram
mappings	Diagram mappings and SDL tools

Figure 24: Task 2 results summarised.

Task 3 – Diagram Construction

Participant #	Task1	Task 2	Task 3	Task 4	Average	Rounded
1	1	1	1	0	0.75	1
2	3	3	3	3	3	3
3	3	3	3	3	3	3
4	2	2	2	1	1.75	2
5	2	2	2	1	1.75	2
6	0	1	0	0	0.25	0
7	3	3	3	2	2.75	3
8	2	2	2	1	1.75	2
Task Average	2	2.125	2	1.375	1.875	2

0=Failed, 1=Average, 2=Good, 3=Excellent

Figure 25: Task 3 results summarised.

Figure 26 summarises the results of analysing user responses to the survey questionnaire and the task completion data. The former are summarised in the first two categories where qualitative responses were analysed and categorised into positive or negative opinions. The latter are summarised in the last two categories, which were categorised according to how complete the solutions provided were against pre-set criteria. Combined these provide general quantitative indications of the usability and efficacy of SDL and the SDLTool.

Category	*Results*
General tool usability	Positive 87.5% Negative 12.5%
Overall notation usability	Positive 75% Negative 25% (half these were only partial negative)
Diagram comprehension (user performance)	Excellent 62.5% Good 25% Average 12.5% Incomplete 0%
Task completion (user performance)	Excellent 37.5% Good 37.5% Average 12.5% Incomplete 12.5%

Figure 26: Results of user survey

Even though the test subjects were new to the concept of using visual language for statistical surveys, the subjects were able to understand and use diagrammatic notations to express various aspects of the survey process and compose statistical techniques to solve test scenarios. The learning curves of the subjects varied but all of them were able to comprehend testing diagrams to the level required for their tasks.

Interpretation

Key findings from the detailed survey comments and observations include the following:

1. SDL accentuates and integrates the multiple aspects of a survey that are often not addressed by existing practice. This makes SDL based solutions more user-oriented. Users interacted with visual notations not just to formulate numerical computations but also to convey actual operational semantics behind those activities. The test subjects responded that this approach would scale well to communicate large-scale survey projects.

2. Participants reported that the visual modeling approach and inter-diagram mappings helped users to think of a survey project as set of tasks within the context of the whole survey process and individual survey constructs to be conceptualized as reusable components.

3. The effect of the visual approach on comprehension performance of both high and low level details of the survey process varied. Each diagram drew different responses but with overwhelmingly positive overall feedback. Survey diagrams were received well by the test subjects as being easy-to-use, expressive and a time-effective alternative to conventional documentation. Two participants viewed survey task diagrams as too radical a departure from existing practices, but the majority stated that the diagrams visualized a valuable aspect of the process. It was interesting to note that the three participants who had previous experience in highly specialized statistical research roles were the most enthusiastic about the concept of task models. Their professional or academic backgrounds ranged from financial modelling to accounting. The tacit existence of prevalent patterns in those specialized fields seemed to be largely responsible for their reactions. Both survey technique and data diagrams were received favourably. They shared strengths and weaknesses as they look alike and convey information at similar levels. Their most notable strength was in capturing an integrated view of sampling, statistical metadata and techniques using an intuitive dataflow metaphor. Negative feedback mostly originated from a lack of *secondary notation* support.

4. As visual language novices, it was not a trivial task for participants to visualize inter-diagram relationships and harmonize disparate diagrams into a single unified model. Even though test subjects did very well in comprehending and utilizing individual diagrams they expressed some difficulty in mapping all the diagrams together when designing the survey process due to *hidden dependencies*. This problem is analogous to a novice UML user's difficulty in merging all UML diagrams mentally together to form a unified view and resulted in some cases in the introduction of inconsistencies. One promising solution suggested by the test subjects to remedy this design issue was the creation of a visual layer to dynamically illustrate how the underlying model is formed by contributing diagrams and the relationships amongst them. Future development of SDL will explore the feasibility of this approach.

5. All five diagrams allowed all graphical entities to be viewed and manipulated in a single view pane at the individual level. This may have addressed a significant portion of *visibility* criteria but without automatic layout management to beautify graphical constructs, visual distractions accumulated as the size of a diagram grew. A diagram layout management component (e.g. automated tree layout management) would alleviate the visual distraction level of large-scale diagrams.

6. One important aspect of SDL is its ability to have a survey represented in multiple diagrams and an SDL diagram's graphical entity may have several sub-level layers that edify lower level information associated with the graphical entity. This design feature of SDL could contribute to potential *juxtaposability* problems, which were commented on by participants. For example a survey data diagram's data entity may have up to 3 sub layers associated with the entity. The presence of the associations is visually noted by changes in the entity's boundary line shape and colour. An ad-hoc remedy such as putting lower-level information on one top-level layer might address this issue but would require a heavy trade off in a diagram's level of visual clutter. *Juxtaposability* could be enhanced by offering tool level enhancements. For instance more fluid navigation between diagram layers allows user's working memory to be minimally disrupted by changing scenes. Another tool-level approach to remedy the trade-off could be interactive 3D visualisation of SDL diagram layers. A 3D environment may provide better a visual perspective for users that side-by-side presentation in 2D with the same screen dimensions cannot offer.

7. Participants responded favourably to the degree of freedom afforded by the lack of *premature commitment* and low *viscosity* the SDLTool offers in designing survey processes and the broad range of diagram types supported. No existing specialized survey tools have functionalities corresponding to survey and task diagrams drawing favourable comment from participants. Participants also noted that, unlike existing tools, they were not restricted to a sequential batch mode or an interactive mode, which tends to require *premature commitment* and also high attention investment [4] to articulate a technique that needs to be frequently modified. Existing tools emphasise result oriented step-by-step batch

operations or UI based pre-packaged procedures. SDLTool's target emphasis is not exclusively result oriented. Its diagrammatic notations aim to capture the whole semantics embodied in a statistical technique thus allowing a type of expressiveness that is intuitive to the user. The *low viscosity* of the data flow metaphor utilized in survey data and technique diagrams provided users with design-time freedom to change input and output data flows and the dynamic mapping of a graphical entity to a physical dataset or statistical technique meant data flows could be routed to multiple techniques in a variety of ways to easily form many variations of the initial design model. All these positive aspects made SDL tools successful in providing visual cues for the whole process while also providing direct manipulation interfaces for a wide range of more detailed operations.

8. Developing and implementing statistical techniques in many popular statistical computing packages entails the translation of symbolic mathematical statements into tool specific languages. When operational requirements are embedded into the survey process, users need to switch back and forth between two vastly different modes of mental operation. If implementations in both mental modes are *inconsistent* or demand *hard mental operations*, the whole survey process can induce harmful side effects such as *a high abstraction barrier*, and *viscosity*. In the visual environment supported by the prototype tool, the test subjects were expected to know correct usage of statistical techniques but they were separated from low-level representations of the techniques that only exist in the forms of service components. Technique constructions are metaphoric abstractions that reside in a single domain. Thus a technique composition in actuality is a model building exercise that is entirely independent from underlying platforms. Although these enhancemets of the prototype tool brought a new level of usability to the test subjects, which they responded favourably to, some instances where the model-level coupling would work against performance efficiency were also found. For example the current SDLTool does not allow users to do code level modifications; expert participants who were comfortable with raw code modifications in R's interactive mode felt they would outperform the SDL-based approach when the statistical techniques, which are provided as a service to SDL tools, are themselves subject to frequent changes. Therefore we will consider possible tradeoffs in implementing a dual-mode access to mapped services in the future development of SDLTool. The example illustrates how established practices may unexpectedly deteriorate the quality of a solution founded on theoretically presumed user patterns.

10. One of the design principles behind SDL was to minimise the set of graphical entities in a diagram (low *diffuseness*). This design decision was principally made to lower the accessibility of the barrier to accommodate a diverse pool of users. The approach was justified by positive user responses regarding the usability of diagrammatic notations and SDL tools. However the presumption that less equals better usability was not the case across all user experiences. As the participants became more immersed in the visual environment, their demand for more direct visual control of data flow grew in scope and functionality.

Our prototype tool allows users to navigate to a sub layer to modify parameters of a mapped service via a dynamically generated UI but once they leave the layer they are no longer able to edit the mapped service either by indirect or direct manipulation and the values of the parameters are hidden behind a graphical icon. Direct control of the mapped service is delegated to the sub layer primarily to reduce users' memory load and to provide an exact contextual perspective for each visual layer as discussed in one of the design principles of SDL. One of the interesting observations during the testing sessions was that more competent users requested the ability to manipulate mapped services within the top-level diagram without navigating to appropriate sub level diagrams. Further post session discussions revealed that the user request is reminiscent of shortcuts. Shortcuts provide a secondary access to application functionalities in a typical GUI design for users who wish to bypass GUI based interactions to save time and effort. Likewise, visual shortcuts should improve user performance by offering time-saving alternatives. A visual shortcut approach will be investigated in conjunction with better elision support in our future work.

Overall, both the quantitative and qualitative analyses support an affirmative answer to RQ1.

9.3 Statistical Service Generation

To further evaluate the "implementation" dimension of RQ1, we evaluated the web service generation support of SDLTool by using it to develop a range of statistical survey designs and technique implementations for various moderately-sized statistical problems.

Methodology

Our exemplar survey technique web service generation problems came from the New Zealand Crime Victimisation Survey and a large existing survey, Predictor of One-Year Development Status in Low Birth Weight Infants [38]. Each of these surveys makes use of a set of tasks, processes, meta-data, data and composed statistical techniques. Generated implementations of survey techniques are a set of scripts, sometimes very complicated, used to drive one or more 3rd party statistical analysis tools. These generated statistical technique implementations are made available to remote, 3rd party applications via generated web service interfaces. Generated WSDL specifications for the web service permit advertising the service via a registry and discovery and invocation of the technique implementation. Generated web

service code supports data translation from remote clients into the target 3rd party tool input format, and from the output format of the 3rd party tools into XML for return.

Surveys rarely become obsolete in the way software does. Thus revisiting existing surveys is common practice for survey developers and users. SDL diagrams can be used to aid review and restructuring of existing surveys by highlighting core semantics of the surveys. We also used SDLTool to model and modify the Predictor of One-Year Development Status in Low Birth Weight Infants [38] case study. This allowed us to investigate a conversion procedure to turn existing survey descriptions and techniques into SDL diagrams and some reusable survey technique web services.

Results

We successfully reused the generated technique web services from within SDLTool, from .NET-implemented remote clients for the statistical survey service, and from scripts used by 3rd party applications. The current tool only turns survey technique diagrams into executable web services, as they are the aspect of survey process design most amenable to code generation. Each technique diagram represents an independent, stateless and reusable set of atomic statistical techniques within the context of a single activity without the orchestration overheads that are inherent in behavioural diagram types such as UML activity diagrams and SDL process diagrams. Generated statistical technique implementations manifested as web services are currently deployed to an Axis server with a single implementation invoking a single 3rd party statistical analysis tool script. Some of these technique implementations are very heavy-weight and simultaneous use of such services by multiple clients is not feasible. A more scalable deployment architecture is required for these technique implementation web services. This includes the ability to deploy multiple services to different web service hosts; the ability to deploy a technique web service and associated generated scripts and 3rd party tools as a unit; and the ability to performance test and engineer the deployed services.

The synthesized statistical technique implementation web services produce proprietary WSDL message formats rather than utilising any standard protocol for encoding statistical data and decoded results data. This means 3rd party client applications using the generated technique web services must be engineered with knowledge of these protocols. With emerging meta-data standards for statistical analysis tools these generated web services should at least conform to an XML-based data representation scheme for such techniques. Additionally, as web service-based technique implementations become more widely accepted in this domain, new protocols to invoke such technique implementation services are likely to emerge. Again our generated web services should conform to such standards to make them compatible with potential 3rd party client applications.

In our experiment with remodelling and reusing the Predictor of One-Year Development Status in Low Birth Weight Infants survey, results were positive. The survey design was readily able to be modelled and some initial design procedures developed to assist survey researchers to convert existing textual survey designs into SDL. We managed to package for reuse several statistical techniques in the survey as web services for reuse by not only a reimplementation of this survey but by other survey tools as well.

9.4 Strengths, Limitations and Lessons Learned

Our three evaluations of SDL and SDLTool have identified their key strengths as being:

- SDL and SDLTool supports all aspects of survey design and development, some better and more completely than others.
- Many aspects of statistic survey design and development that are currently ad-hocly described and carried out are both formalised and structured, especially the inter-relationship of different aspects of survey design.
- The distinct visual representations used for each SDL diagram type are generally accessible and found to be useful by participants in our studies
- The generated web services embodying survey techniques are reusable from SDLTool-based survey designs and by a range of third party survey data analysis tools

Weaknesses that we have identified include:

- While each SDL diagram type is limited in notational elements and semantics, the combination forms a large and relatively complex visual language suite, resulting in a moderate learning curve, similar to that of UML does for software modelling.
- Inter-process dependencies are not particularly well expressed and SDL still has significant hidden dependency issues, a problem common with many DSVLs.
- The underlying SDL meta-model and composite model formed by the multiple SDL diagrams constructed in SDLTool is hidden from the user; making these composite elements and properties accessible may assist in accuracy, completeness and overall survey design understanding.

- Using SDL within SDLTool requires knowledge of the abstractions and languages but tool support to advise users of mistakes or to suggest improvements to constructs and approaches is currently limited.
- Process diagrams are documentation-oriented rather than executable; supporting the latter would be helpful to assist in further automation of survey implementation.
- While the web services generated are reusable, only limited off the shelf technique implementation tools are currently supported.
- Limited and fairly generic support for version control, reusable patterns for various diagram types, diffing and merging, check-in/checkout etc are provided via Pounamu. While these are quite powerful, tailoring many of them more to SDLTool user's needs would make the tool both easier to use and more effective.

Overall, we judge that use of SDL and its supporting environment, SDLTool support an affirmative answer to RQ1 (Can visual notations with appropriate software tool support be used by end users, i.e. statisticians and other survey developers, to facilitate statistical survey design and implementation?).

- We have been able to provide designers with explicit, domain-specific visual language (DSVL) representations of the overall statistical survey process, many common tasks, and statistical survey data and techniques. Novel concepts such as our survey diagrams and task diagrams could usefully be adopted to describe complex statistical surveys with or without using SDLTool itself.
- Using the DSVLs hat we developed for SDL does appear to provide both an effective and efficient approach to survey design. It is unclear how much benefit this is to expert designers, though our experts were positive about the more tangible representations of aspects of survey design in SDL.
- For novice and intermediate survey designers, SDL and SDLTool provide enhancements over existing batch-oriented, low-level tools. Our web service generation support is relatively novel compared to most existing platforms. Experts are generally comfortable with using existing tools but see benefits for documenting surveys that lack suitable current notations, and for teaching statistical survey design.

We have adopted a range of approaches to the development and evaluation of SDL and SDLTool that we hope other DSVL developers will find useful. We briefly summarise the key approaches we adopted and lessons we have learned from their application in this research project.

- Multiple techniques for requirements gathering and validation: as we were not experts in the target domain of this research, we found that we had to adopt a range of information gathering approaches, and multiple ways of validating the information gathered. As described in Section 4, this included corpus analysis using natural language extraction; user-centric design, visual language design principles and noticing the similarity of statistical survey design process structures to software process structures. On this project we found all of these provided useful guidance. On other DSVL projects, we may have more expertise or may find other techniques helpful e.g. more formal comparative analysis to existing approaches, brainstorming solutions with focus groups etc.
- Principles for DSVL notational and meta-model design: this work was carried out guided by an early CD evaluation of an early SDL design, using Burnett et al's visual language design guidelines, using general usability principles, and was constrained by our meta-tool platform, Pounamu's, capabilities, in terms of visual language rendering and editing support. The latter, one might argue, should not overly constrain DSVL design, but in our experience if one wants good tool support for the DSVL, one must work within the limitations of the eventual implementation platform capabilities. Were we design SDL now, we would also inform SDL design using Moody's Physics of Notations framework [32], which we have successfully used on several of our more recent DSVL tool research projects.
- We used evaluations often and early not only in this work but many other of our DSVL tool development projects. We also used a range of evaluations – end user, heuristic (in this case, based on the CD framework), and walkthrough. We would have used comparative analysis of SDLTool to existing survey design tools – however existing tools lack a great many of SDL and SDLTool's capabilities (e.g. survey overview, task model, process model and even high-level data and technique representations) making such comparisons trite.

9.5 Future work

Several areas of future work have already been identified, specifically better representation of inter-process dependencies, design critics to catch errors such as the attribute attachment error [1], and multiple view/elision support for large diagrams [27]. In addition, we see considerable scope for providing back end integration with other statistical survey tools so that our SDL environment can be used to not only design a statistical survey, but also implement and control it. Pounamu has an increasingly sophisticated set of integration mechanisms, including RMI and web services based APIs, code generation and import, together with collaborative work support that allows multiple designers to use Pounamu generated tools collaboratively. These can be leveraged to integrate Pounamu with other statistical packages. However, we also see the opportunity to provide a more generic framework for integration using a meta-data based approach for specifying legacy tool capabilities. We are exploring this in current work. We plan to add executable

statistical process diagrams, user scripting via the R language for more powerful analysis capabilities for expert users, and repositories for both statistical data and reusable process models. Further enhancements include secondary notation support for annotative diagrams, visual presentation of underlying statistical survey process enactment and data analysis state, improved visual presentation of mapping forms, and automatic layout of some diagrams.

10. Summary

We have described SDL, a set of visual notations for specifying statistical surveys, and its support environment, SDLTool, with modelling support, generation and execution support for complex statistical survey implementations, and generation of reusable web services encapsulating survey techniques. SDL aims to provide a similar modelling framework for statistical survey design as UML does for software design. SDL is comprised of five diagrammatic types, two of which has multiple levels of abstraction. These together represent both the data and process-oriented components of a statistical survey design. SDLTool supports modelling with SDL and integration of rich statistical meta-data, data sets, 3rd party technique implementations for analysis, and 3rd party visualisation tools. Statistical survey technique implementations are made available to 3rd party client applications via generated web services, which are then deployed and advertised for discovery and invocation. These web services are synthesized from the domain-specific visual language models in SDLTool. A cognitive walkthrough with a subject expert was undertaken to evaluate the usability of SDL. In addition, we undertook a larger cognitive dimensions informed usability study of SDLTool with eight undergraduate statistics students. We have also generated web service interfaces to several complex statistical survey techniques defined and tested in SDLTool. We have used these web services from remote clients to send data to the technique's web service, have this processed by the 3rd party analysis tools used to realise the technique's implementation, and return data to the client for further analysis and result visualisation. Service and statistical survey documentation is also generated by SDLTool to aid understanding and reuse. Combined these evaluations provided support for the efficacy of our approach.

Acknowledgements

The authors gratefully acknowledge the assistance of James Reilly as statistical consultant, our panel of user evaluators, and Nianping Zhu for implementing the Pounamu meta tool. Support from the Foundation for Research, Science and Technology for parts of this research is gratefully acknowledged.

References

[1] Ali, N., Hosking, J.G., Huh, J. and Grundy, J.C. Template-based Critic Authoring for Domain-Specific Visual Language Tools, *2009 IEEE Symposium on Visual Languages and Human-Centric Computing*, Cornwallis, Oregon, USA, Sept 20-24 2009.

[2] Biemer, P.P. and Lyberg, L.E. *Introduction to survey quality*, Wiley Inter-Science 2003, Chapter 2.

[3] Biezunski, M Topic Maps at a glance, *XML EUROPE '99*, Granada, Spain, 26-30 April 1999.

[4] Blackwell, A.F., "First steps in programming: a rationale for attention investment models," *IEEE 2002 Symposia on Human Centric Computing Languages and Environments,* Arlington, Virginia, USA, September 3 – 6, 2002.

[5] Bottoni, P., Costabile M. F., Levialdi, S. , Matera, M., Mussio, P., Principled Design of Visual Languages for Interaction, *IEEE Symposium on Visual Languages,* Seattle, Washington, USA, Sept 10-14 2000, pp. 45-52.

[6] Burnett, M., Baker, M., Bohus, C., Carlson, P., Yang, S., van Zee, P., Scaling Up Visual Programming Languages *IEEE Computer*, March 1995, 45-54.

[7] Chambers, J.M. and Ryan, B.F., The ASA Statistical Computing Section: A History, *The American Statistician, 4 (2)*, May 1990, pp 87-89

[8] CSPro, http://www.census.gov/ipc/www/cspro/

[9] DDI, The Data Documentation Initiative, http://www.icpsr.umich.edu/DDI/

[10] Danado, J., & Paternò, F. Puzzle: a visual-based environment for end user development in touch-based mobile phones. *Human-Centered Software Engineering*, Springer Berlin Heidelberg, 2012.

[11] Daniel, F., Soi, S., Tranquillini, S., Casati, F., Heng, C., & Yan, L. From people to services to UI: distributed orchestration of user interfaces. *Business Process Management*, Springer 2010, pp. 310-326.

[12] Dodero, J. M., del Val, Á. M., & Torres, J. An extensible approach to visually editing adaptive learning activities and designs based on services, *Journal of Visual Languages & Computing*, *21*(6), 2010, 332-346.

[13] Fogli, D., & Provenza, L. P. End-user development of e-government services through meta-modeling. In *End-User Development*, Springer, 2011, pp. 107-122.

[14] Gillman, D. and Appel, A. The Statistical Metadata Repository: an electronic catalog of survey descriptions at the U.S. census bureau, *IASSIST Quarterly*, Summer 1997.

[15] Garg, P. and Jazayeri, M. Process-Centred Software Engineering Environments, *Software Process,* Wiley, pp25-49, 1996

[16] Green, T.R.G and Petre, M, Usability analysis of visual programming environments: a 'cognitive dimensions' framework, *Journal of Visual Languages and Computing*, 7 (2), June 1996, pp.131-174.

[17] Green, T. R. G., Burnett, M. M., A Ko, J., Rothermel, K. J., Cook, C. R., and Schonfeld, J., Using the Cognitive Walkthrough to Improve the Design of a Visual Programming Experiment *IEEE Symposium on Visual Languages*, Seattle, Washington, Sept. 2000, 172-179.
[18] Ihaka, R. and Gentleman, R. R: A Language for Data Analysis and Graphics, *Computational and Graphical Statistics*,5 (3), 1996, pp. 299-314.
[19] Jenkins, S. G. The Triple-S survey interchange standard http://www.triple-s.org/sssasc96.htm
[20] Kugler, H., Larjo, A., & Harel, D. (2010). Biocharts: a visual formalism for complex biological systems, *Journal of The Royal Society Interface*, *7*(48), 1015-1024.
[21] Kim, C. H. *Visual Language and Environment for Statistical Surveys*, MSc Thesis, Department of Computer Science, University of Auckland, February 2006.
[22] Kim, C. H., Hosking, J., Grundy, J., Model Driven Design and Implementation of Statistical Surveys, *40th Annual Hawaii International Conference on System Sciences*, Hawaii, 3-6 Jan 2007.
[23] Kim, C., Hosking, J., and Grundy, J., A Suite of Visual Languages for Statistical Survey Specification, *2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, Dallas, Texas, USA, 20-24 Sept 2005, 19-26.
[24] Knuplesch, D., Reichert, M., Ly, L. T., Kumar, A., & Rinderle-Ma, S. Visual modeling of business process compliance rules with the support of multiple perspectives. *Conceptual Modeling,* Springer, 2013, pp. 106-120.
[25] Lanzenberger, M., Sampson, J., & Rester, M. Ontology Visualization: Tools and Techniques for Visual Representation of Semi-Structured Meta-Data, *Journal of Universal Computer Science, 16* (7), 2010, 1036-1054.
[26] Lieberman, H., Paternò, F., and Wulf, V., *End user development*. Human-Computer Interaction Series, Vol. 9. Springer, 2006.
[27] Li, L. Hosking, J.G. and Grundy, J.C. Visual Modelling of Complex Business Processes with Trees, Overlays and Distortion-Based Displays, *2007 IEEE Symposium on Visual Languages and Human-Centric Computing*, USA, Sept 23-27 2007, IEEE CS Press.
[28] Liu, H. and Lieberman, H. Toward a Programmatic Semantics of Natural Language, *2004 IEEE Symposium on Visual Languages and Human-Centric Computing*, Rome, Italy, September 26-29, 2004.
[29] Madansky A., On Biblical Censuses, *Journal of Official Statistics*, 2 (4), 1986. pp. 561-569
[30] Mellor, S.J., Scott, K., Uhl, A., Weise, D. *MDA Distilled: Principles of Model-Driven Architecture*, Addison-Wesley, 2004.
[31] MetaNet, 2003 *Metanet Project Meeting Final Report*, Chapter 1, Samos, May 2003, http://data-archive.ac.uk/media/1689/METANET_proceedings_finalreport.pdf
[32] Moody, Daniel. The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering, *IEEE Transactions on Software Engineering,* 35 (6), 2009, pp. 756-779.
[33] NZCS 2003, New Zealand National Survey of Crime Victims in 2001, New Zealand Ministry of Justice
[34] Office for National Statistics (ONS) UK , *UK Labour Force Survey Statistical Outputs Group*, http://www.statistics.gov.uk/STATBASE/Source.asp
[35] Olenski, J. Global Standard for Harmonization of Social Statistics, *Expert Group Meeting on Setting the Scope of Social Statistics United Nations Statistics Division in collaboration with the Siena Group on Social Statistics,* New York, 6-9 May 2003, https://unstats.un.org/unsd/demographic/meetings/egm/Socialstat_0503/docs/no_10.pdf.
[36] OMG, *OMG Unified Modelling Language,* http://www.uml.org/
[37] Object Management Group, *What is OMG-UML and Why is it important?* html://www.omg.org/new/pr97/umlprimer.html
[38] Pederson, D., Evans, B., Chance, G., Bento, A. and Fox, A. Predictor of One-Year Development Status in Low Birth Weight Infants, *Journal of Developmental and Behavioural Paediatrics*, 9 (5), Oct 1988, p287-92.
[39] Pepper, S., The TAO of Topic Maps, *Ontopia AS*, 2002, http://www.ontopia.net/topicmaps/materials/tao.htmlSAS Institute Inc. http://www.sas.com
[40] SPSS, SPSS Inc. *SPSS statistical software,* http://www.spss.com
[41] SPSS Inc., *SurveyCraft*, http://www.spss.com/surveycraft/
[42] Statistics Netherlands, *Blaise*, http://www.cbs.nl
[43] Sutcliffe, A. *Domain Theory: Patterns for Knowledge and Software Reuse,* Lawrence Erlbaum Associates, Inc. Mahwah, NJ, USA, 2002
[44] Unhelkar, B., and Henderson-Sellers, B., Modelling Spaces and the UML, 2004 *Information Resources Management Association Conference*, New Orleans, 2004
[45] Wajid, U., Namoun, A., & Mehandjiev, N. (2011). Alternative representations for end user composition of service-based systems. *End-User Development* (pp. 53-66). Springer Berlin Heidelberg.
[46] Young, F.W. & Bann, C.M. ViSta: *A Visual Statistics System Statistical Computing Environments for Social Research*, Sage Publications, Inc., 1997, 207-235
[47] Zhu, N., Grundy, J.C. and Hosking, J.G., Pounamu: a meta-tool for multi-view visual language environment construction, 2004 IEEE Symposium on Visual Languages and Human-Centric Computing, Rome, Italy, 25-29 Sept 2004, pp. 254-256.

Captions

Figure 1. Basic statistical survey process.

Figure 2. An example SDL Survey Diagram for the 2001 New Zealand Crime Victimisation Survey.

Figure 3. SDLTool usage overview.

Figure 4: Relationships between SDL diagrams

Figure 5. (Left) Survey diagram notation and (right) example usage.

Figure 6. (a) Basic survey task diagram notation; (b) OR grouping; (c) iterative tasks.

Figure 7. Basic crime survey task decompositions.

Figure 8. (Left) Survey data diagram notation and (right) example of usage in Crime survey.

Figure 9. Two Crime survey data diagram examples.

Figure 10. Statistical technique diagram notation.

Figure 11. (left) Simple chi-square technique and (2) survey technique diagram for discriminant test.

Figure 12. Survey process diagram notation.

Figure 13. Survey process diagram for the crime victimisation survey.

Figure 14. Examples of NZ Crime Survey SDL diagrams.

Figure 15: Binding statistical resources to SDL elements

Figure 16: Examples of execution of statistical technique diagrams including results visualization.

Figure 17. Generating and using a statistical survey technique implementation and its web service interface.

Figure 18: (left) Inter diagram mappings and (right) partial topic-map representation of SDL.

Figure 19: SDL tool architecture

Figure 20: SDL toolset analysis pipe-line

Figure 21. SDLTool architecture.

Figure 22. (Left) Survey diagram generated by test subject and (right) Initial survey data diagram given to subject.

Figure 23. Survey process diagram generated by test subject for harvest estimation survey.

Figure 24: Task 2 results summarised.

Figure 25: Task 3 results summarised.

Figure 26: Results of user survey