ISSN 1744-1986

Technical Report N° 2010/ 18

Heuristics and Software Design—A Case Study

Darrel Ince

12 November, 2010

Department of Computing
Faculty of Mathematics, Computing and Technology
The Open University

Walton Hall, Milton Keynes, MK7 6AA
United Kingdom

http://computing.open.ac.uk

Heuristics and Software Design—A Case Study

Darrel Ince
Dept of Computing, Open University
Walton Hall, Milton Keynes
MK7 6AA
d.c.ince@open.ac.uk

Abstract

This article describes a case study in which a software system costing hundreds of thousands of pounds was redesigned and re-implemented using a series of heuristics due to the Hungarian mathematician George Pólya. The heuristics are first outlined by applying some of the more important ones to a small programming problem. Then the application of the heuristics applied to the re-design of the system is detailed. Because of the attention to abstraction the resulting system is able to be used in a variety of applications—not just the one that it was envisaged for.

Keywords: Abstraction, System Design, System Requirements, System Failure, Human-centered Computing, Simplicity.

1 Introduction

Gregor Pólya was an international mathematician who worked in a wide number of areas including number theory, probability, statistics, geometry and combinatorics [21]. He is, however, remembered primarily for his seminal work on problem solving 'How to Solve it' [20]. It detailed a number of heuristics that can be used by mathematicians in order to solve problems in both the pure and applied branches of the subject. A number of his heuristics are detailed in Table 1. Pólya's influence on mathematical reasoning has been immense; for example [24]

> The mathematician best known for his conceptualization of mathematics as problem solving, and for his work in making problem solving the focus of mathematics instruction, is Pólya. Indeed, the edifice of problem solving work erected in the past two decades stands largely on the foundations of his work.

His ideas are clearly reflected in major works in mathematical thinking such as [12] and [29]. The aim of this article is to show how a number of his heuristics have relevance to software development as well as mathematics.

2 An Example

The triangle problem is a famous problem that has been used by software testing researchers for the last 40 years. It was initially posed by Glenford Myers [18].

Heuristic	*Informal description*	*Mathematical concept*
Analogy	Find an analogous problem	Map
Generalization	Find a more general problem to be solved	Abstraction
Induction	Develop a general solution from concrete examples	Induction
Specialisation	Develop a specific solution from concrete examples	Concretisation
Auxiliary problem	Solve another problem that helps?	Lemmas and theorems
Decomposition	Split the problem into small problems	Establishment of sub-theorems
Problem variation	Vary the problem	Solution searching
Auxiliary elements	Add something	Extension
Auxiliary elements	Remove something	Constriction

Table 1: Some Pólya heuristics

The problem is

> Process three integers and, first, determine whether they represent the sides of a triangle. If they do, discover whether the triangle is isosceles (two sides equal), equilateral (three sides equal) or scalene (no sides equal).

A first attempt at the triangle categorization part of the problem might involve a number of `if` statements. This is the solution usually presented in the testing literature. As an example of a case study for structural (branch) testing it is excellent. As an example of software design it lacks elegance. In order to aim for a simpler solution it is worth generalizing. One potential generalization is to pose the problem for an n-sided figure and constrict the problem.

> Process n integers that represent the sides of an n-dimensional figure if they do discover whether the figure has n sides equal, $n-1$ sides equal... no sides equal.

For the sake of illustration it is worth constricting the problem a little by specifying that there is only one set of duplicate integers; for example the collection

$$1, 2, 3, 88, 4, 2, 99, 88, 4, 23, 88, 2$$

would be disallowed.

A Java program that solved this n-sided problem is shown below

```
int n=// Set the value of the number of sides here
int  j, count=0;
int[] sides=// Set the values of the array here
for(int i=0;i<n;i++){
    j=i+1;
    while(j<n){
```

```
            if (sides[i]==sides[j])
              count++;
            j++;
        }
        if (count>0) break;
    }
```

All it does is to process each element of the array `sides` containing the sides and as soon as an equality has been found in the inner loop it quits[1]. On termination the number of equal sides will be one more than `count`.

This is quite a complicated solution; it does however, have the advantage it solves the problem for any n-sided figure. If you want to make it more efficient then all that is needed is to specialize the problem by giving each of the sides of the triangle variable names (`a`, `b` and `c`) and comparing each for equality. This is shown below

```
if(a==b) count++;
if(b==c) count++;
if(c==a) count++;
```

where if `count` is 0 a scalene triangle is found, if it is 1 then an isosceles triangle is found and if it is 3 then an equilateral triangle has been discovered.

Can we go further? One of the features of the 'general' solution is that it is not quite general: it is grounded in a geometrical interpretation. Perhaps looking at the problem as something more abstract will produce something more elegant. A re-specification is shown below

> Process n objects and categorize whether n objects are equal, $n-1$ objects are equal ... or no objects are equal.

Here objects are specified. Provided that these objects have an `equals` method associated with them, then solving this problem will be feasible. The problem has now been varied, but still has the core functionality: that of categorizing equality of a set of objects.

A set solution suggests itself. One of the features of a set is that when items are added to it then any duplicates found via the `equals` method associated with the objects are not added. The Java code for this very general solution is shown below, `noOfInt` is the number of objects added while `arr` is an array containing the objects.

```
for(int j=0;j<noOfInt;j++){
   container.add(arr[j]);
}
```

When the code has completed the number of duplicates is given by

```
noOfInt-container.size()+1
```

where `size` is the `cardinality` of the set `container`[2].

[1] For purists the alternative to the `break` statement is to have a Boolean that is set after the inner loop has completed which is used to exit the outer loop.

[2] The result 1 indicates no matches as with the previous code.

3 The Triangle Problem and the Pólya Heurstics

The solutions detailed previously all employ a number of the Pólya heuristics. First a concrete problem grounded in geometry was generalized to a more abstract problem in geometry—the transformation from a 3-sided figure problem to an n-sided figure problem. The problem was augmented by adding a condition to the problem before it was transformed to the n-sided figure problem—the restriction to only *one* collection of duplicate elements. The solution to the n-sided figure was then specialized back to a three-sided figure problem in order to derive a simple solution to the original posed problem. The problem was then reposed in terms of a collection of objects—here it was made more abstract than the previous n-sided abstraction. This final abstraction gave rise to a solution that was general, both in terms of the number of objects that could be processed, but also the type of objects.

Clearly this is a toy problem. It is presented here as an example of the Pólya heuristics in action. The key question is whether they can be used in a larger context. The aim of the remainder of this article is to explore this question.

4 The Integrated Children's System

The Integrated Children's system is a set of processes that the previous British Labour government imposed on the social work profession. They were supported by a number of computer packages. A key event in the development of ICS was the death of a young black child, Victoria Climbié. She died at the hands of her guardians. Her death occurred while she was under the care of the Haringey Social Services Department. She was burned by cigarettes, scalded, kicked, assaulted with heavy objects such as wooden coat hangers, abused and beaten; in the final months of her life she slept in a bath within a bin bag containing her own excrement and urine and was fed while in the bag.

4.1 The Climbié inquiry

The inquiry that was set up as a result of the death of Victoria Climbié by the British government was an extensive investigation of social work practice in the United Kingdom. The vast majority of the recommendations concerned the overhaul of social work practice and established a set of tightly regulated processes that had to be adhered to by British social workers who were involved in children's work. These processes were to be supported by computer systems which had, at their heart, a database that stored the details of the interaction between a social worker, a child and others such as the child's family, the local pædiatric department of a hospital and the local police-based, child protection unit.

The response from social work academics to the Climbié report was prescient, for example Stanley [28] predicted that

> The danger is that tighter structures for scrutinizing and monitoring child care work will result in a focus on procedure rather than increasing practitioners' capacity to engage with families; more boxes will be ticked but children will still go unheard.

Rustin [23] also criticized the bureaucratic and procedure-oriented nature of public service delivery that the Climbié inquiry exemplified. Reder and Duncan [22] predicted that that there would be a risk that all that would happen was the emergence of more procedures and processes.

4.2 The development of the ICS Software Packages

The Department of Children, schools and Families[3] developed a number of documents that formed the system specification. These included a high quality, top-level requirements document; minimum compliance criteria that an ICS system would have to pass; a set of exemplars, these were documents containing a large number of forms which were mandated for any ICS implementation; a graphics/text-based process model 202 pages in length and a data model 187 pages in length.

Versions of ICS were implemented by a number of systems developers based on requirements documents provided by the DCSF[4]. In *technical terms* ICS can be regarded as a success: 80% of the implementations were delivered to time, the remaining 20% were delivered shortly after and the only technical errors were those that could be expected during the initial deployment of a system. The system developers cannot be faulted for the systems they produced: they successfully implemented the requirements. It is now widely recognized that the specification was wrong.

4.3 ICS in use

A good starting point in examining the success or otherwise of the ICS is the evidence that UNISON, a trade union that represents many staff in local government made to a second Laming Inquiry which arose from the appalling death of another child known in the British press as Baby P. It claimed that

> ...problems appear to be fundamental, widespread and consistent enough to call into question whether the ICS is fit for purpose.
>
> The reports we have received show that the direct impact of the system is to delay, frustrate and disrupt the flow of work in busy and over-stretched teams.

The problems of ICS IT fall into a number of categories[5].

Technical problems These included slow page loads, slow data saves and time-out problems. These were due to implementing a heavy duty industrial system on local government computer systems[6].

[3]This was the branch of British government responsible for children; after the Conservative election victory in 2010 it was renamed as The Department for Children.

[4]There was a carrot and stick approach from the DCSF in that the system was mandated, but that funding was given to local authorities that wished to buy or develop support packages. The only local authority that did not conform was the Royal Borough of Kensington and Chelsea who developed their own system; this won a British Computer Society prize.

[5]It is worth re-stressing that these problems were not implementation problems but with the government specification that reflected a massively over-complex view of what children's social work entailed.

[6]British local government has traditionally spent much less on computer hardware and software than commercial companies.

Workflow problems These included an inability to use the system outside the office, process design issues, poor system design and overall problems with workflow.

Problems associated with the chunking of data These included an inability to have multiple windows open at all times, problems with multiple records, atomization of data and the number of clicks needed to achieve a task.

Task-related problems Screen design issues, form design issues and unwieldy form design

There were also a number of academic critiques. Broadhurst *et al* [3] found that social workers carrying out initial assessment of children were confronted with major problems; Shaw *et al* [25] examined ICS from the point of view of statistical reporting, the experience of social workers and the implications of ICS for social work practice in terms of the time used by social workers in interacting with the system; they found major problems in all these areas; and Wastell, White and Broadhurst [31], in the most critical assessment of the ICS, pointed out that it represented a chiasmus in that an IT-supported process intended to aid social work had actually shackled it.

The DCSF commissioned a report into the functioning of the ICS in a number of pilot authorities by the University of York [2, 25, 26, 27]. It came to the conclusion that the ICS system was seriously flawed. A worrying statistic that arose from the York evaluation was the large amount of time that social workers spent using a computer—something like 65% to 80% of their time—leading to their skimping on important tasks such as visiting a child at risk and their family. This extra commitment to bureaucracy did not result in a major increase in accuracy in reporting; indeed there were substantial problems in statistical reporting [25].

The DCSF *eventually* agreed there were problems: in June 2009 they stopped mandating ICS as a child care system[7]. At the time of writing English local authorities no longer have to adhere to the mandates of ICS, but have to cope with complex expensive systems that implement the ICS mandates[8].

4.4 The nature of ICS

ICS was specified as a system that addressed three needs:

- The needs of front-line social worker. It should have provided facilities that enabled the development of a chronicle of the interaction between the social worker and others: the child, their family, their school, community health staff etc.

[7] At the time of writing (Nov. 2010) it is still mandated in Wales.

[8] For example Hartlepool Borough Council entered into a £700 000 contact over a five year period for their ICS system. The author is currently a member of the government's review of social work—the Munro Review—and visited the social work department of a small local authority that has expended over £200 000 per year on advisory staff who help social workers understand and use their ICS computer system—the equivalent of the staff costs of about 5–7 social workers.

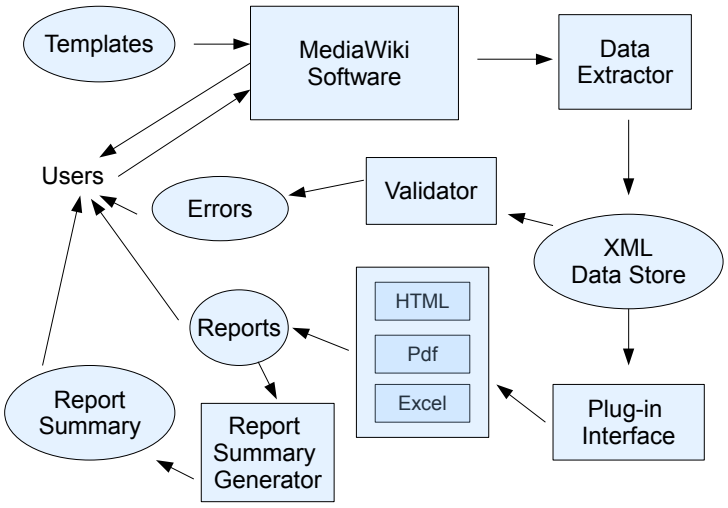

Figure 1: The Architecture of a Wiki-based Chronicling System

- Local management. The management of a children's department require a number of reports to carry out their functions, for example reports on the progress of cases and on the current workload of social workers.
- Government, via reports such as the Children in Need Census, require details of the number of children in need and their current state, for example the number of children in the care of a local authority and the number who are the subject of action in a family court.

The core problem with the ICS and its associated computer packages was that reporting dominated. In the spectrum bounded by front-line utility and government reporting it lies somewhere between local government and national government reporting. The result of this was that social workers found themselves using a computer for the majority of their time—acting as data entry clerks—rather than carrying out their essential functions[9]. The narrative process was relegated to a minor aspect of work. The ICS lead to the narrative being atomized into small chunks effectively reducing the story of child in need to a series of fragments that were not related to each other. This is not just a feature of ICS, but has occurred in other government-initiated, human-centred systems [32]; however, in ICS it is a very serious problem.

[9]One of the problems was that free text was confined to text boxes a few hundred characters in length and social workers spent hours trying to edit their notes to fit into the space. This was a consequence of the use of relational databases.

5 The Redesigned System

5.1 Analysis

The analysis for the requirements specification was carried out by applying a latent error protocol that is used by the Boeing Corporation for discovering problems during aircraft maintenance to the report of the Victoria Climbié inquiry [11]. One of the problems with the response to such reports is that surface errors tend to have been addressed and little attempt was made to dig deeper into latent errors [15]. Munro has articulated the problems and a solution based on a systems approach [16].

> Solutions take the form of trying to control erratic practitioners: psychological pressure to achieve higher standards, increasing formalization and guidelines to reduce the scope of individual fallibility, and stricter management surveillance. The inquiry into the death of Victoria Climbié fits this model. However, thirty years of such inquiries have not lead to the expected improvement in professional practice. Indeed the Climbié report describes several agencies operating at a very low level, and failing to implement the most basic elements of good practice. A similar history of failure in engineering has led to the development of a systems approach.

Three examples of the approach are::

1. There was an occasion when a referral was made to Brent Council on June 21 1999; documents for this went missing; there was also confusion with regard to a previous referral and indeed a question whether there was in fact a referral. The latent error here was clear: that an inadequate filing system had been set up and a proper set of processes were not employed. The solution is simple: a proper filing system, even one based on a word-processor together with adequate procedures for depositing documents to a central server.

2. Victoria Climbié was also under the care of Haringey Council. There was a handbook provided by Haringey which outlined a number of agencies roles in the child protection process. This was a guide which provided information about inter-agency working. Staff involved with Victoria had never heard of the handbook. The latent error here was the lack of visibility of the handbook. The solution again was simple: manual processes that made staff familiar with any documentation and the display of the documentation either as help facilities or as accompanying text on any forms that have to be filled out.

3. An important handwritten FAX concerning Victoria's medical condition sent by the Central Middlesex Hospital was delayed by ten days before reaching the relevant social worker after it was received by Haringey and was hard to decipher. The latent error here was the primitive means of communication used and the lack of an adequate communication channel. The solution was again simple but perhaps partially not available at the time: the use of email as a communication medium with manual procedures that ensured that the email was received by the right person and logged.

While the stress in the above bullet points is that of latent error, it is worth saying that *some* individual performance was below that expected of a professional as described by the quote from Munro in Section 5.1; however, this would have been picked up much earlier if adequate computer support and managerial processes were extant in the various agencies that were involved in the care of Victoria.

The impression gained from a latent error analysis of the Victoria Climbié inquiry report is, *in the main*, not of staff who were incompetent, lazy or uncaring but of staff from all the agencies, not just social service departments, who were often overwhelmed with work, carrying out multiple roles—some of them contradictory—and not supported by a proper set of procedures and a system that could have been implemented using a word processor [7].

An interesting outcome from the latent error analysis was that the requirements specification that was constructed was very close to the specification that was initially developed by the DCFS; this was then considerably expanded to support the process model defined by the DCSF and which gave rise to a 600 page document that effectively shackled British childrens social workers.

5.2 The Eventual Architecture

In order to discuss the architecture and use of the system some vocabulary is needed. It is general in that it is designed for a collection of professionals each of whom are responsible for a collection of client. Each client is associated with an event. Typical professionals would include a children's social worker, probation officer or doctor; typical clients would be a child, an offender and a patient; and typical events would be a referral from a someone such as a community health worker, the release of an offender from some institution or a medical consultation

The architecture of a Wiki-based system for chronicling is shown as Figure 1. At the heart of the system is *MediaWiki* [1]. This is the software system that implements the online encyclopædia *Wikipedia*. One of the features of the work that is described here is that it was comparatively easy to develop the system because *MediaWiki* had a large amount of required functionality embedded within it or could be easily added[10]. There are three examples of this.

First, a number of ICS packages relegated narrative text to a minor supporting role compared with the priority that structured data had. The *MediaWiki* package makes free text a first class citizen and, as will be demonstrated later can implement structured text.

Second, the Climbié report detailed a number of occasions where versions of documents were lost and arguments about who actually filled in the forms. *MediaWiki* has an excellent version control system that allows tracking to any change to a document.

Third, it is exceptionally easy to tag entities with categories; this is, of course, a consequence of the prominence of taxonomies in *Wikipedia*. This means that children can, for example, be tagged as subject to a family court proceedings, being fostered or subject to an s47 enquiry[11] where a children's department

[10]This was the equivalent of using a set in the triangle problem.

[11]Section 47 of the British Children's Act 1989 places a duty on Local Authorities to make enquiries into the circumstances of children considered to be at risk of 'significant harm' and, where these enquiries indicate the need, to undertake a full investigation into the childs

think harm will be occasioned to a child.

A professional interacts with the system by adding events to the collection of events associated with a client (each professional is associated with a web page that lists all the clients). Each collection of events is stored as a *Wikipedia*-like article. Each event is associated with a collection of structured data and free text describing the event. For example when a client is referred to a department the associated structured data would include the name of the referee, their address and other contact information, while the free text would contain details of their referral.

When a professional wants to add an event they copy a *MediaWiki* template from a web page where it is stored and then add it to the chronicle of events associated with a client. The template contains data names and space for the data values.

The second most important component is a data extractor. This issues a PHP command to the *MediaWiki* software and gathers in the list of professionals, the clients that they are responsible for and the article that holds a sequence of events each of which contains structured and unstructured data. The data is written to a store in an XML format.

The third component is a validator whose thread is periodically woken to examine the data store and flag errors in structured data. These errors are written to a web page and can be corrected by a user.

The fourth component of the system is a plug-in interface that allows access to the stored XML source. Three visitor patterns [5] provide programming facilities for the generation of *MS Excel* spreadsheets, the generation of tables in pdf format and the generation of web pages.

This facility allows reports to be developed very easily, for example a report that generated a hyper-linked pdf document with the hyperlinks pointing at both professionals, clients and individuals events in a client's chronicle took 165 lines of Java most of it straight line code except for a 17 line method for simple text justification which nested two loops and a small loop which was embedded in a conditional statement and extracted out event attributes from XML source.

The fifth component is an index generator that periodically examines the folder containing reports and updates a web page that provides summary report data: the name of the report and its hyper-link and the format of the report.

The code is general in that it implements the model of professionals being responsible for clients who are associated with a sequence of events. The only place in the system where the application would be betrayed is in a short list of system constants implemented as a Java interface. These contain data such as the name of the professional XML tag and the regular expressions used for data extraction.

In passing it is worth saying that all the components of the system were developed using either free or free and open source software. A list is detailed. in Table 2.

6 Heuristics and the Development of the System

In this section I deliberately change the style of writing from the academic to the personal in order to provide a chronicle of a number of the design decisions that

circumstances.

Software	*Use*
NetBeans	Java IDE used for development
Apache and PHP	Web site hosting
MediaWiki	Storage of chronicles
pdflatex	Generation of pdf-based reports
Apache HSSF	Generation of Excel files
Apache SAX API	Generation of reports

Table 2: Software used

were made. The Pólya heuristic that was employed at each stage is italicized.

Many of the decisions were unconscious ones but can be mapped to the Pólya heuristics (these are highlighted in italics in the remainder of this section). However, the decisions based on moving up a level of abstraction were a conscious application of one of the heuristics.

The first part of the development involved examining specific examples of the system to be developed, I examined some of the specifications of the childrens system and then designed two very primitive versions of the system that could support ICS functionality(*Induction*). I was unhappy with both of these solutions as they were too concrete and too tied to the ICS model of social work, any changes would have been difficult to carry out. The two solutions (one a word-processed solution, the other a wiki-based solution) however, had commonality: each involved a textual database which was indexed by keywords and each reflected the relationship:

```
Children's social worker -> Children -> Sequence of events
```

The next stage was to look at packaged solutions to the problem. The key to developing an improved system for childrens social work was in placing narrative to the forefront of functionality. The author is currently serving as a member of the Munro review of childrens social work and as part of the work has interviewed around 16 social workers in England and Wales in order to discover what they required from a future replacement of the ICS system. Without exception they asked for facilities whereby they could construct a narrative of their interaction with a child, their family and other agencies such as the local community health department, this was also a theme running through the research that had been carried out on the ICS system; it is also a common thread running through the perception of human-centred computer systems by users [32]. This suggested a solution that was centred on documentation management systems. These provide facilities for metadata initiation and maintenance and the capture, indexing, storage and retrieval of documents written in standard formats such as those found with MS Word (*Analogy*). This would have made an adequate solution; however, many documentation management systems are over-specified with respect to the narrative forming functionality required for children's social work.

At this stage I was still thinking of a system which addressed the needs of children's social workers, but a conversation with a friend who is a director of social services in Wales made me think in more general terms. She told me that although children's social work was important there was the danger that adult social work would be ignored and become the Cinderella of social

services. It was clear that if you looked at social work generally there was a commonality in that it involved social workers interacting with a client and other agencies and documenting important events that occurred such as an initial referral, a case conference and a request for some service internal to a social work department or externally such as a service from a community health department (*Generalisation*).

So the next stage was to think of a solution in terms of social work generally and the model of professional interacting with a client who is associated with a time-ordered sequence of events became central. During the year in which this research was carried out I suffered a detachment of my retina and required a number of operations. I attended a number of consultations and during a conversation with the surgeon it struck me that the general model of interaction between myself and him or one of his staff fitted the same pattern (professional=doctor, client=patient and sequential series of events: initial diagnosis, *in-vitro* operation, eye examination with ophthalmoscope, laser surgery and final discharge. I therefore decided to design a system that could be used generally by doctors, social workers, probation officers and teachers, for example (*Generalization*).

A number of studies of mathematicians have dispelled the idea that there is a linear progression from say the conjecture of a theorem to the proof of that theorem that features such as ambiguity, contradiction and paradox are employed within the search of a solution space [4]. Furthermore, that the solution space is not bounded by the branch of mathematics that is addressed, for example a graph theorist may delve into category theory or algebraic topology in order to reuse results in that area to support their progression to a final result. This is embodied in the Pólya heuristic (*analogy*). This was employed in the next stage where I revisited the Wiki approach.

One of my initial designs of a chronicling system employed Wiki software as a device to hold narrative and structured data associated with an event. During the latter stages of this project a model of social work was being proposed in which multi-disciplinary teams work very closely with a child, their family and other agencies; one aspect of this is the facilitation of sharing data and information and the group development and critiquing of childrens work (for a case study see [17]). This lead me back to the wiki approach and it was clear that the interaction between the writers of an article in *Wikipedia* is roughly the same as the interaction between social workers in an interdisciplinary team: there is the initial development of an article (chronology) and the subsequent addition of further detail (addition of events to chronology) and the commenting and critiquing of article content (development of a rationale for changes within the *MediaWiki* talk page) (*Auxiliary Problem* and *Generalisation*).

The system as developed can be specialized by the addition of application-specific templates and the development of simple reporting code that accesses the XML database for a variety of applications ranging from medicine to school reporting. (*Specialisation*). The plug-in architecture that has been developed allows the attachment of specific events and the development of reports as requirements change or to differentiate the management structures within different children's departments (*Problem Variation* and *Decomposition*).

The description of the design decisions shows vertical movement over the levels of abstraction, the consideration of auxiliary problems (designs / implementations) such as Wikis and word processors and the use of analogy. Table 3

Heuristic	*Informal description*	*Mathematical concept*
Analogy	Find an analogous design	Map
Generalization	Develop a more general design	Abstraction
Induction	Develop a general design from concrete examples	Induction
Specialisation	Develop a specific design from concrete examples	Concretisation
Auxiliary problem	Seek another design that helps	Lemmas and theorems.
Decomposition	Split the design into small problems	Establishment of sub-theorems
Problem variation	Vary the design	Solution searching
Auxiliary elements	Add something to the design	Extension
Auxiliary elements	Remove something from the design	Constriction

Table 3: Some Pólya heuristics associated with system design

is a rewriting of Table 1 with the focus on design rather than on problem solving

7 Summary

The core of this article concerns the application of a specific collection of heuristics associated with the development of mathematical artifacts. However, it is worth pointing out some smaller novelties. The first is the use of latent error analysis to develop a requirements specification. This is a fruitful area for further research, for example there is a considerable literature on error and patient safety [30] and the use of the latent error technique detailed here provides evidence of its utility in, for example, developing a chronicling system for a medical discipline such as ophthalmology, oncology or neurology.

The second novelty is that of employing an existing large piece of software for the vast majority of the functions of a system. Employing *MediaWiki* enabled a system to be developed that consisted only of just under a thousand lines of Java code. It is a concrete example of the maxim that you should look around for similar designs and implementations before starting *ab initio* on the development of a system.

The key outcome however is that the Pólya heuristics provide a framework within which the designer can navigate a design solution space that enables the traversal of levels of abstraction (upwards, downwards and sideways) to suggest designs that have much more utility than a design that is initially developed from a more concrete specification. I would also contend that a focus on high-level abstractions and employing the *analogy* heuristic enabled the *MediaWiki* system to be identified.

The content of this article is most relevant to the topic of system frameworks and the reuse agenda that it chimes in with. Frameworks have arisen from work on design patterns or micro-architectures detailed in [5]. A framework being a software scaffolding containing hooks into which software components can be inserted. Where the work reported here differs is that it is not reuse in the

small through components, but was achieved via interaction with a system that which was not originally designed as a placeholder for components, but in which communication is via an API that reads and writes data to a textual database of *Wikipedia* topics. There are of course some parts of the system that are inspired by frameworks and design patterns, for example the visitor pattern [5] is used to provide a hook that enables pdf, Latex and *MS Excel* spreadsheet reports to be produced. However, the essence is of reuse in the large.

This article is not about architecture; it is about how the processes that contribute towards the development of an architecture can be informed by a specific set of heuristics that are embedded in the thought processes used by mathematicians. There has been no shortage of articles about architecture, reuse and frameworks since the publication of [5], for example [9] and [19]; indeed, the decrease in the number of articles since 2005 indicates that perhaps little research remains. There is, however, a shortage of work on heuristics; there are two exceptions. First, Jackson's seminal work on requirements analysis and problem frames is inspired by the work of Pólya [8]; also Michalewicz and Fogel's book on optimization [14] draws heavily on Pólya's heuristics in describing relatively small algorithms for optimization[12].

When I look back on the thought processes, intuitions and events that lead to the design and development of the chronicling system the major conclusion reached is how difficult it would be to teach the heuristics. The development of the chronicling system arose from a reading of [5] in the late nineties, a reading of a paper on public sector failure while recovering from a series of eye operations [6] which posited the fact that public sector computer failures would reduce if simple systems were considered, a conversation with an eye surgeon, a reading of a book on *MediaWiki* [1], a conversation with a director of social services, a number of conversations with children's social workers and my use of 'How to Solve it' at a mathematics summer school. Perhaps the greatest challenge that this article poses is that of the teaching of these heuristics—particularly those associated with abstraction—to an emerging generation of software engineers [10].

References

[1] Barrett, D. *MediaWiki*. O'Reilly Media, 2009.

[2] Bell, M., Shaw, I., Sinclair, I., Soper, P., and Rafferty, J. An evaluation of the practice, process and consequences of the ICS in councils with social services responsibilities. Report, 2007.

[3] Broadhust, K., Wastell, D., White, S., Hall, C., Peckover, S., Thompson, K., Pithouse, A., and Davey, D. Performing 'Initial Assessment' Identifying the Latent Conditions for Error at the Front-Door of Local Authority Children's Services. *British Journal of Social Work* (2009 Forthcoming).

[4] Byers, W. *How Mathematicians Think*. Princeton University Press, 2007.

[12] As an aside it is also worth saying that Microsoft used to give all their new programmers a copy of 'How to Solve it' [13].

[5] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison Wesley, 1994.

[6] Goldfinch, S. Pessimism, Computer Failure, and Information Systems Development in the Public Sector. *Public Administration Review* (September/October 2007), 917–929.

[7] Ince, D., and Griffiths, A. A Chronicling System for Children's Social Work:Learning from the ICS Failure. *British Journal of Social Work* (2011 Forthcoming).

[8] Jackson, M. A. *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*. Addison-Wesley ACM Press,, 1995.

[9] Johnson, R. E. Frameworks = Components + Patterns. *Communications of the ACM 40*, 10 (Oct 1997), 39–42.

[10] Kramer, J. Is Abstraction the Key to Computing? *Communications of the ACM 50*, 4 (2007), 36–42.

[11] Laming, W. *The Victoria Climbié Inquiry*. Dept. of Health, 2003.

[12] Maddox, R. *A Transition to Abstract Mathematics: Learning Mathematical Thinking and Writing*. Academic press, 2008.

[13] McConell, S. *Code Complete: A Practical Handbook of Software Construction*. Microsoft Press, 2004.

[14] Michalewicz, Z., and Fogel, D. B. *How to Solve It: Modern Heuristics*. Springer-Verlag, 2000.

[15] Munro, E. Improving Practice : Child Protection as a Systems Problem. *Children and Youth Services Review 27*, 4 (2005), 375–391.

[16] Munro, E. A Systems Approach to Investigating Child Abuse Deaths. *British Journal of Social Work 35*, 3 (2005), 531–546.

[17] Munro, E. *The Munro Review of Child Protection*. Dept. for Education, 2010.

[18] Myers, G. *The Art of Software Testing*. Wiley, 1971.

[19] Parsons, D., Rashid, A., Telea, A., and Speck, A. An Architectural Pattern for Designing Component-based Application Frameworks. *Software Practice and Experience 36* (2006), 157–190.

[20] Pólya, G. *How to Solve It*. Penguin, 1990.

[21] Pólya, G., and Alexanderson, G. *The Random Walks of George Pólya*. Mathematical Association of America, 2001.

[22] Reder, P., and Duncan, S. Making the Most of the Victoria Climbié Inquiry. *Child Abuse Review 13* (2004), 95–114.

[23] Rustin, M. Learning from the Victoria Climbié Inquiry. *Journal of Social Work Practice 18*, 1 (2004), 9–18.

[24] Schoenfeld, A. H. Learning to Think Mathematically: Problem Solving, Metacognition, and Sense-making in Mathematics. In *Handbook for Research on Mathematics Teaching and Learning*, D. Grouws, Ed. MacMillan, 1992, pp. 334–370.

[25] Shaw, I., Bell, M., Sinclair, I., Sloper, P., Mitchell, W., Dyson, P., Clayden, J., and Rafferty, J. An Exemplary Scheme? An Evaluation of the Integrated Children's System. *British Journal of Social Work 39* (2009), 613–626.

[26] Shaw, I., and Clayden, J. Technology, Evidence and Professional Practice: Reflections on the Integrated Children's System. *Journal of Children's Services 4*, 4 (2009), 15–27.

[27] Shaw, I., Morris, K., and Edwards, A. Technology, Social Services and Organizational Innovation or How Great Expectations in London and Cardiff are Dashed in Lowestoft and Cymtyrch. *Journal of Social Work Practice 23*, 4 (2009), 383–400.

[28] Stanley, N. A Year on from the Climbié Inquiry. *Child Abuse Review 13* (2004), 75–79.

[29] Sternberg, R., and Ben-Zeev, T. *The Nature of Mathematical Thinking*. Lawrence Erlbaum, 2009.

[30] Vincent, C. *Patient Safety*. Wiley-Blackwell, 2010.

[31] Wastell, D., White, S., Broadhurst, K., Hall, C., and Peckover, S. The Chiasmus of Design: Paradoxical Outcomes in the E-government Reform of UK Children's Services. In *IFIP Advances in Information and Communication Technology* (2009), G. Dhillon, B. Stahl, and R. Baskerville, Eds., Springer Boston, pp. 257–272.

[32] White, S., Hall, C., and Peckover, S. The Descriptive Tyranny of the Common Assessment Framework: Technologies of Categorization and Professional Practice in Child Welfare. *British Journal of Social Work 39*, 7 (2009).