Modelling and Refinement of Forensic Data Acquisition Specifications

Benjamin Aziz

School of Computing
University of Portsmouth
Portsmouth PO1 3HE, United Kingdom
Telephone:+4402392842265
Fax:+4402392842525
Email: benjamin.aziz@port.ac.uk

Abstract

This paper defines a model of a special type of digital forensics tools, known as data acquisition tools, using the formal refinement language Event-B. The complexity and criticality of many types of computer and Cyber crime nowadays combined with improper or incorrect use of digital forensic tools calls for more robust and reliable specifications of the functionality of digital forensics applications. As a minimum, the evidence produced by such tools must meet the minimum admissibility standards the legal system requires, in general implying that it must be generated from reliable and robust tools. Despite the fact that some research and effort has been spent on the validation of digital forensics tools by means of testing, the verification of such tools and the formal specification of their expected behaviour remains largely underresearched. The goal of this work is to provide a formal specification against which implementations of data acquisition procedures can be analysed.

Keywords:
Computer Forensics, Disk Data Acquisition, Formal Specifications, Event-B Method, Formal Refinement

1. Introduction

Digital forensics tools are becoming increasingly of a critical nature due to the complexity of attacks on digital assets and the sophisticated role that computer and Cyber systems play in modern day crime. As a result, there is

continuous need in the law enforcement community to ensure the high quality of generated evidence and acceptable reliability levels for forensic tools used in digital crime investigations, particularly when such investigations are global and/or carry significant importance Friedberg (2012). As a result, it is important to understand properties of digital forensic tools, in particular, where correctness, accuracy and completeness of such tools is vital to the course of justice and the discovering of facts. This view is supported by research in recent years in the area of digital forensics modelling Carrier and Spafford (2004); Ciardhuáin (2004); Beebe and Clark (2005); Ieong (2006); Cohen (2009); Casey and Rose (2010), where the need for the development of more robust and rigorous scientific methods is highlighted in the area of digital forensics in Garfinkel et al. (2009).

The National Institute of Standards and Technology (NIST) project on the Computer Forensic Tool Testing NIST (http://www.cftt.nist.gov/) aims at raising the assurance of computer forensic tools by providing informal definitions of the various computer forensic tools and the requirements underlying such tools. These requirements are then used for the development of functional specifications, test procedures, criteria, sets and hardware. In this paper, we take this assurance process to another level where the functional specifications and some of the properties of the computer forensic tools are formally defined and verified using a well-established framework based on the Event-B method Abrial (2010). According to Eoghan Casey Casey (2011), such formalisation "encourages a complete, rigorous investigation, en-sures proper evidence handling and reduces the chance of mistakes created by preconceived theories, time pressures and other potential pitfalls.".

The Event-B method facilitates the modelling of system specifications based on a combination of set-theoretic and action semantics Mosses (1986); Watt (1987). The top-level abstract model is then refined by adding more detail and by following the rules of refinement Abrial et al. (2005) until the desireable level of refinement is reached. In this paper, the abstract model for a data acquisition tool is first defined and then refined by adding more detail that distinguishes between accessible and inaccessible data in the acquired source, and then by including contructs for preserving the integrity of the acquired data based on hash functions. Throughout this refinement, the focus of the work is on capturing some of the main requirements on data acquisition tools as stated by NIST NIST (2004), in particular requirements related to the accuracy and completeness of such tools. The result that the work shows is that though completeness is possible to express generally, accuracy is not.

As a result, we conclude that any implementations of NIST's specification of a data acquisition tool must deal with accuracy in a delicate manner, paying attention to the accessibility property of the acquired data.

The rest of the paper is structured as follows. In Section 2 we discuss related work. In Section 3, we give a brief introduction to the Event-B method and language. In Section 4, we give an overview of NIST's main requirements for a data acquisition tool. In Section 5, we define the first abstract model of a data acquisition tool along with its completeness property. In Section 6, this model is refined by distinguishing between accessible, hidden and inaccessible data in the digital source. We show here that accuracy is possible to define. In Section 7, we further refine the specification to include the concept of hash functions and defined based on these the data integrity requireements of the tool. Finally, we conclude the paper in Section 8 and discuss future research directions.

2. Related Work

The application of formal modelling and analysis techniques to digital forensics is by no means a new idea, though it has been under-researched in many aspects. In Gladyshev and Enbacka (2007), the B method Abrial (1996) is used for developing incosistency checks and verifying the correctness of digital evidence. The B method has also been used to formally specify and refine write blocker systems in Enbacka and Laibinis (2005); Enbacka (2007) based on NIST's informal definitions of these systems in NIST (2003) and provide formal definitions of the properties of these systems. Our work here follows on the footsteps of Enbacka and Laibinis (2005) by adopting similar approach for a different type of digital forensic tools.

In Leigland and Krings (2004), the authors prpose a formal model for analysing and constructing digital forensic procedures. The model is based on set theory and incorporates attacks on systems. In Stephenson (2003), the author uses coloured Petri Nets to model root cause analyses of digital incidents (i.e. digital post mortems). In Gladyshev (2005), finite state machines are used as a defense tool to exploit weaknesses in claimed evidence in computer investigations. The approach is applied to a case of blackmail investigations, where finite state machines are used to demonstrate alternative scenarios to the claimed incident. More recently, James et al. (2009) compute the intersection of the various states in a finite automata to reconstruct events and evidence related to a specific crime incident. Earlier, in Carrier

(2006), Carrier defines a model of hypothesis-based digital forensics based on finite state machines. The model captures the concept of *computer history* and consequently, formalises evidence based on this concept.

In Rekhis and Boudriga (2005, 2010), the authors developed a logic-based model, called $S\text{-}TLA^{+}$, capable of describing complex investigations and generate evidence under different levels of abstraction. The model is also capable of expressing anti-forensic attacks and provides the machinery to detect such attacks based on the analysis of their action traces. Recently, this model was extended in Rekhis and Boudriga (2012) to include a theory of hierachical visibility providing better verification framework of anti-forensic attacks. In Mazza et al. (2011); Métayer et al. (2011), the authors propose a formal framework for specifying and reasoning about decentralised logs, and define an analysis that can generate both precise and approximate evidence of past events.

There are some frameworks and methodologies that propose a testing approach to the validation of digital forensics tools, including among others NIST (http://www.cftt.nist.gov/); Beckett and Slay (2007); Guo et al. (2009); Shamala and Azizah (2012). Nonetheless, formal verification and analysis of such tools remains an area of research laregely unexplored, towards which this paper aims to contribute.

3. Event-B

Event-B Abrial (2010) is an extension of Abrial's B method Abrial (1996) for modelling distributed systems. This section presents a brief overview of Event-B. Modularity is central to the Event-B method and this is achieved by structuring specifications and development into *machines*. Machines are essentially abstract data types with states, representing an abstract model of a system. An Evetn-B machine can be refined and implemented. The correctness of the machines and the refinements can be validated by proof obligations. Invariants and other predicates are given in first order predicate calculus and set theory. The underlying logic is untyped.

In Event-B, machines are defined in a context, which has a unique name and is identified by the keyword `CONTEXT`. It includes the following elements: `SETS` defines the sets to be used in the model; `CONSTANTS` declares the constants in the model; and finally, `AXIOMS` defines some restrictions for the sets and includes typing constraints for the constants in terms set membership. When a context is refined, it `EXTENDS` its related abstract context.

An Event-B machine is introduced by the `MACHINE` keyword, it has a unique name. A machine `SEES` a particular context, which means that it is able to access any sets or constants declared in that context. The machine also includes the following elements. `VARIABLES` represents the variables (state) of the model. `INVARIANT` describes the invariant properties of the variables defined in the clause `VARIABLES`. Typing information and general properties are described in this clause. These properties shall remain true in the whole model and in further refinements. Invariants need to be preserved by the initialisation and events clauses. `INITIALISATION` allows to give initial values to the variables of the system.

`EVENTS` cause the state to change by updating the values of the variables as defined by the *generalised substitution* of the event. Events are guarded by a *condition*, which when satisfied implies that the event is permitted to execute by applying its generalised substitution in the current state of the machine.

Event-B also incorporates a refinement methodology, which can be used by software architects to incrementally develop a model of a system starting from the initial most abstract specification and following gradually through layers of detail until the model is close to the implementation. A machine `REFINES` its abstract parent from which it was refined. This means that some of the events in the refined machine may be refinements of their parent events, in which case this is indicated by using the keyword `extends`.

In Event-B, an event is defined by the following syntax:

EVENT e *WHEN* G *THEN* S *END*

where G is the guard, expressed as a first-order logical formula in the state variables, and S is any number of generalised substitutions, defined as:

$$S ::= \quad x := E(v) \quad | \quad x := z : |P(z) \quad | \quad S \parallel S'$$

The deterministic substitution, $x := E(v)$, assigns to variable x the value of expression $E(v)$, defined over set of state variables v. In a non-deterministic substitution, $x := z : |P(z)$, it is possible to choose non-deterministically local variables, z, that will render the predicate $P(z)$ true. If this is the case, then the substitution, $x := z$, can be applied, otherwise nothing happens. Finally, substitutions can be composed in parallel, $S \parallel S'$.

For a comprehensive description of the Event-B language, its semantics

and its associated toolkit, Rodin, we refer the reader to other references such as Abrial (2010); Abrial et al. (2010); Abrial and Hallerstede (2007); Métayer et al. (2005).

4. Disk Data Acquisition Specifications

Forensic data acquisition is a process that involves the identification of a *digital source*, such as a hard disk, a memory card or any other form of media and data storage, and the copying of the identified data to some accessible *destination object*, such as an *image file*, a *clone* or a *bit-stream duplicate*, performed in a complete and accurate manner. Hence, *completeness* and *accuracy* are the two most important features that any data acquisition tool must demonstrate, in order for the tool to be considered of a forensic standard of quality.

In practice, there is an abundance of disk imaging tools such as dd for Linux and Clonezilla, as well as the more general computer forensic toolkits that implement disk imaging, for example, the Slueth Kit, Encase and the FTK Imager. The NIST Digital Data Acquisition Tool Specification NIST (2004) is a document that outlines the main requirements expected of such implementations of the data acquisition functionality. These requirements are highlighted as *mandatory* and *optional* features of such tools. Here, we focus on most of the main mandatory requirements, particularly paying attention to those that deal with the completeness and accuracy of the tool, and consider a couple of optional ones that are of interest to the integrity of the acquired data. NIST also provides a specification of the plans and assertions for testing digital data acquisition tools NIST (2005), but this is currently outside the scope of this work.

The NIST document mentions a number of mandatory requirements related to disk data acquisition tools.

- **DI-RM-01**: The tool shall be able to acquire a digital source using each access interface visible to the tool.
- **DI-RM-02**: The tool shall be able to create either a clone of a digital source, or an image of a digital source, or provide the capability for the user to select and then create either a clone or an image of a digital source.

- **DI-RM-03**: The tool shall operate in at least one execution environment and shall be able to acquire digital sources in each execution environment.
- **DI-RM-04**: The tool shall completely acquire all visible data sectors from the digital source.
- **DI-RM-05**: The tool shall completely acquire all hidden data sectors from the digital source.
- **DI-RM-06**: All data sectors acquired by the tool from the digital source shall be accurately acquired.
- **DI-RM-07**: If there are unresolved errors reading from a digital source then the tool shall notify the user of the error type and the error location.
- **DI-RM-08**: If there are unresolved errors reading from a digital source then the tool shall use a benign fill in the destination object in place of the inaccessible data.

Requirement **DI-RM-01** states the general functionality of a disk data acquisition tool. Requirement **DI-RM-02** relates to the two general methods of data acquisition; imaging and cloning. In our model, we do not deal with this distinction, nor with the distinction in the tool's execution environment as stated in requirement **DI-RM-03**. The completeness property of the tool is demonstrated by requirements **DI-RM-04** and **DI-RM-05**, which also related to the two types of data the tool is expected to deal with, i.e. hidden and visible data. Accuracy is expressed in **DI-RM-06**, and finally, requirements **DI-RM-07** and **DI-RM-08** are related to the case of errors occuring during the acquisition process. Interestingly, **DI-RM-08** hints at a third type of data that the tool should deal with, namely, inaccessible data. We consider in our model later only hidden data that are accessible.

In addition to the above features, we also tackle in our model a couple of interesting optional features:

- **DI-RO-16**: If the tool offers block hash logging and block hash logging is selected then the tool shall log correct hashes for blocks of the requested size from the digital source.

- **DI-RO-18**: If the tool offers acquisition of a digital source that is unprotected by a write block tool or device then an unprotected source shall not be modified during the acquisition process.

The first requirement is related to the integrity of the acquisition, whereas the second requirement protects the source against any writing.

5. The Abstract Model: General Definition of Data Acquisition

The abstract model consists of an abstract context and an abstract specification of the disk data acquisition tool as depicted in Figure 1.

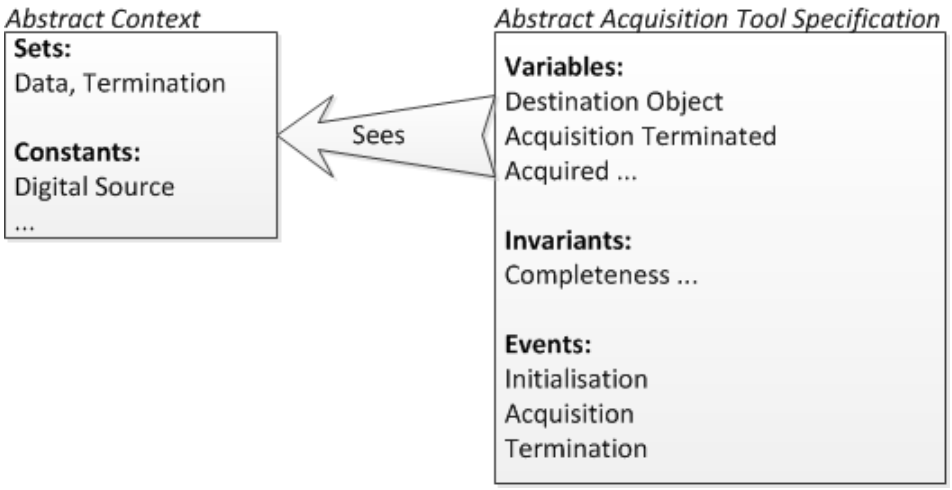

Figure 1: A Representation of the Abstract Context and Machine of the Disk Data Acquisition Tool.

The specification of this model is shown in Figure 2. The context defines two types; *Data* representing the set of all of possible data, and *Termination*, representing a binary *yes*/*no* value to indicate whether or not some process has terminated. We call the source of acquisition *DigitalSource* as per the NIST terminology, and for simplicity we assume it consists of three data elements, *sourceData1*, *sourceData2* and *sourceData3*, which can be thought of as data in the digital source. Finally, *null* represents a bad data element that does not belong to the digital source. The context also includes various axioms on the above two data types and their member elements.

CONTEXT AbstractContext
SETS

Data The set of all possible data
Termination A binary set denoting whether acquisition has terminated or not

CONSTANTS

DigitalSource The source of digital data being imaged
yes
no
null
*sourceData*1
*sourceData*2
*sourceData*3

AXIOMS

axm3 : $Data \neq \varnothing$
axm1 : $DigitalSource \subseteq Data$
axm2 : $Termination = \{yes, no\}$
axm6 : $yes \neq no$
axm16 : $null \in (Data \setminus DigitalSource)$
axm17 : $sourceData1 \in DigitalSource$
axm18 : $sourceData2 \in DigitalSource$
axm19 : $sourceData3 \in DigitalSource$
axm20 : $sourceData1 \neq sourceData2$
axm21 : $sourceData1 \neq sourceData3$
axm22 : $sourceData3 \neq sourceData2$

END

MACHINE AbstractAcquisitionToolSpecification

SEES AbstractContext

VARIABLES

DestinationObject The destination object where the data is placed
AcquisitionTerminated Variable denoting the termination of the acquisition process
Acquired Variable to express whether an element in the source has been acquired or not

Figure 2: The Abstract Context and Specification of the Disk Data Acquisition Tool.

INVARIANTS

DestinationObjectType : $DestinationObject \subseteq Data$
AcquisitionTerminatedType : $AcquisitionTerminated \in Termination$
AcquiredType : $Acquired \in DigitalSource \rightarrow Termination$
Completeness : $\forall x \cdot (x \in DigitalSource \wedge AcquisitionTerminated = yes) \Rightarrow Acquired(x) = yes$

EVENTS

Initialisation

begin

act1 : $DestinationObject := \varnothing$
Destination object is empty initially
act3 : $AcquisitionTerminated := no$
Initially, acquisition process has not terminated
act5 : $Acquired : |(dom(Acquired') = DigitalSource) \wedge (ran(Acquired') = \{no\})$
Initially, every element in the source object is not acquired yet

end

Event *Acquisition* $\widehat{=}$

any

$sourceDataElement$ Pick some element in the source object
f

where

grd3 : $AcquisitionTerminated = no$
Check if acquisition has terminated or not
grd2 : $Acquired(sourceDataElement) = no$
such that the element has not been acquired yet
grd4 : $f \in DigitalSource \rightarrow Data$

then

act1 : $DestinationObject := DestinationObject \cup \{f(sourceDataElement)\}$
The source element is added to the destination after applying the f behaviour function

act2 : $Acquired := Acquired \lhd\!\!+ \{(sourceDataElement \mapsto yes)\}$
Mark source element as having been acquired

end

Event *Termination* $\widehat{=}$

when

grd1 : $\forall x \cdot x \in DigitalSource \Rightarrow Acquired(x) = yes$
grd2 : $AcquisitionTerminated = no$

then

act1 : $AcquisitionTerminated := yes$

end

END

Figure 2: The Abstract Context and Specification of the Disk Data Acquisition Tool (continued).

The machine *AbstractAcquisitionToolSpecification* represents the first most abstract specification of a disk data acquisition tool, as per NIST's definition. The machine "sees" the above defined context. In addition to the initialisation event, the specification machine introduces two events expressing the main functionality of a disk acquisition tool: an *Acquisition* and a *Termination* event. Additionally, the machine defines three variables; *DestinaitonObject* representing the destination object to which the disk data will be copied, a binary-valued counter *AcquisitionTerminated* to indicate whether the acquisition process has terminated or not, and *Acquired*, which is an overloaded function expressing whether a data element in the digital source has been acquired or not.

The acquisition event will update the destination object with a value corresponding to a new data element, *sourceDataElement*, selected from the digital source. The value to which it is mapped depends on an abstract function f, which is used to model "how" the source element is copied to the destination. At this level, f is kept abstract and hence can express any possible behaviour, as long as it is deterministic behaviour (i.e., f is a function and not a relation). This is necessary since this behaviour will differ in the case of copying accessible versus copying inaccessible data as we shall explain in more detail in the first refinement of this specification later. Therefore, at this level, the acquisition is denoted as the function application $f(sourceDataElement)$, where the type of f is chosen non-deterministicly. This implies that it is possible here to give any clear definition of what accuracy means as per NIST's definition of accuracy. More specifically, the model is abstract enough to include both **DI-RM-06** and **DI-RM-08**, therefore, not permitting accuracy to be expressed here.

On the other hand, since we log all the acquisition steps using the *Acquired* variable, it is possible to state the completeness property of the model as follows as per requirements **DI-RM-04** and **DI-RM-05**:

$$\forall x \cdot (x \in DigitalSource \land AcquisitionTerminated = yes) \Rightarrow (Acquired(x) = yes)$$

which means that when the acquisition process has terminated, "every" element x in the digital source will have a *yes* value in *Acquired* (i.e. will have been acquired). This result indicates that while it is possible to have a general definition of completeness, accuracy cannot be defined generally due to the conflicting requirements of **DI-RM-06** and **DI-RM-08** and therefore

must have a definition that is customised for both accessible and inaccessible data, as we show next in the first refinement of this abstract model.

6. First Refinement: Data Visibility and Accessibility

The first refinement represents the addition of more detail to the initial abstract context and machine of the previous section. The refined context, as shown in Figure 3, is now able to distinguish between the hidden, visible and inaccessible visible data on a digital source. It also introduces a "benign" data element necessary for expressing the requirements of **DI-RM-08**, when acquisition fails in copying data to the destination. Associated with this benign element is a *benignfill* function, which is used to express such failure behaviour. The specification of this refinement is shown in Figure 4.

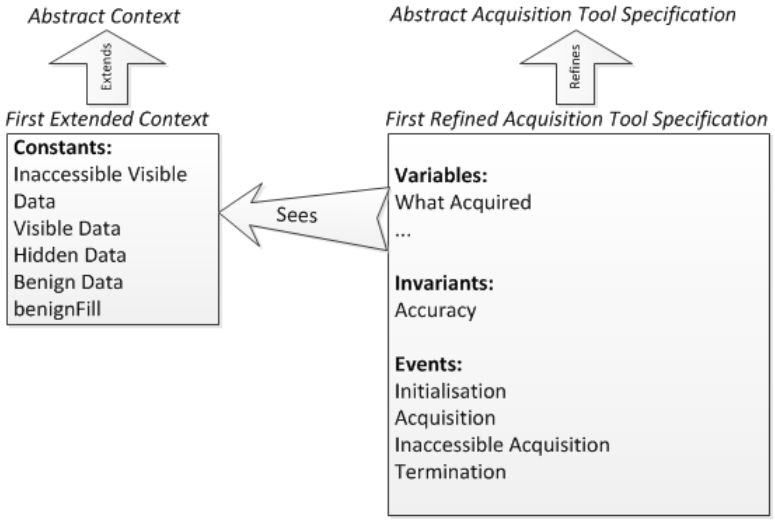

Figure 3: A Representation of the First Extended Context and Refined Machine of the Disk Data Acquisition Tool.

On the other hand, the first refinement machine introduces an additional event called *InaccessibleAcquisition*, which is an extension of the *Acquisition* abstract event. This event picks an inaccessible data element and fills its destination with a benign data value, using the *benignfill* function. This translation is logged and it is done for every inaccessible data element in the digital source.

CONTEXT FirstExtendedContext
EXTENDS AbstractContext
CONSTANTS

$InaccessibleVisibleData$
$VisibleData$
$HiddenData$
$benignData$
$benignfill$

AXIOMS

axm3 : $InaccessibleVisibleData \subseteq DigitalSource$
axm4 : $VisibleData \subseteq DigitalSource$
axm5 : $HiddenData \subseteq DigitalSource$
axm6 : $InaccessibleVisibleData \cap VisibleData = \varnothing$
axm7 : $HiddenData \cap VisibleData = \varnothing$
axm8 : $InaccessibleVisibleData \cap HiddenData = \varnothing$
axm9 : $DigitalSource = HiddenData \cup VisibleData \cup InaccessibleVisibleData$
axm14 : $VisibleData \neq \varnothing$
axm15 : $InaccessibleVisibleData \neq \varnothing$
axm16 : $HiddenData \neq \varnothing$
axm10 : $benignData \subseteq Data$
axm11 : $benignData \cap DigitalSource = \varnothing$
axm12 : $benignfill \in DigitalSource \rightarrow benignData$
axm17 : $null \notin benignData$

END

MACHINE FirstRefinedAcquisitionToolSpecification
REFINES AbstractAcquisitionToolSpecification
SEES FirstExtendedContext
VARIABLES

DestinationObject The destination object where the data is placed
AcquisitionTerminated Variable denoting the termination of the acquisition process
Acquired Variable to express whether an element in the source has been acquired or not
WhatAcquired Variable denoting what value the source element has been mapped to in the destination

INVARIANTS

Accuracy : $\forall x \cdot (x \in DigitalSource \wedge AcquisitionTerminated = yes) \Rightarrow$
$(WhatAcquired(x) = (\lambda y \cdot y \in DigitalSource | y)(x)) \vee$
$(x \in InaccessibleVisibleData \wedge (WhatAcquired(x) = benignfill(x)))$

Figure 4: The First Refinement of the Context and Specification of the Disk Data Acquisition Tool.

EVENTS

Initialisation

extended

begin

act1 : $DestinationObject := \varnothing$
Destination object is empty initially

act3 : $AcquisitionTerminated := no$
Initially, acquisition process has not terminated

act5 : $Acquired : |(dom(Acquired') = DigitalSource) \land (ran(Acquired') = \{no\})$
Initially, every element in the source object is not acquired yet

act6 : $WhatAcquired : |(dom(WhatAcquired') = DigitalSource) \land (ran(WhatAcquired') = \{null\})$
Initially, source elements have no acquired values in the destination

end

Event *Acquisition* $\widehat{=}$

extends *Acquisition*

any

$sourceDataElement$ Pick some element in the source object
f

where

grd3 : $AcquisitionTerminated = no$
Check if acquisition has terminated or not

grd2 : $Acquired(sourceDataElement) = no$
such that the element has not been acquired yet

grd4 : $f \in DigitalSource \rightarrow Data$

grd5 : $sourceDataElement \notin InaccessibleVisibleData$

grd6 : $f = (\lambda x \cdot x \in DigitalSource | x)$

then

act1 : $DestinationObject := DestinationObject \cup \{f(sourceDataElement)\}$
The source element is added to the destination after applying the f behaviour function

act2 : $Acquired := Acquired \lhd\!\!+ \{(sourceDataElement \mapsto yes)\}$
Mark source element as having been acquired

act3 : $WhatAcquired := WhatAcquired \lhd\!\!+ \{(sourceDataElement \mapsto f(sourceDataElement))\}$
Log the mapped value

end

Event *Termination* $\widehat{=}$

extends *Termination*

when

grd1 : $\forall x \cdot x \in DigitalSource \Rightarrow Acquired(x) = yes$

grd2 : $AcquisitionTerminated = no$

then

act1 : $AcquisitionTerminated := yes$

end

Figure 4: The First Refinement of the Context and Specification of the Disk Data Acquisition Tool (Continued).

Event $InaccessibleAcquisition \mathrel{\widehat{=}}$

extends $Acquisition$

any

$sourceDataElement$ Pick some element in the source object

f

where

grd3 : $AcquisitionTerminated = no$
Check if acquisition has terminated or not

grd2 : $Acquired(sourceDataElement) = no$
such that the element has not been acquired yet

grd4 : $f \in DigitalSource \rightarrow Data$

grd1 : $sourceDataElement \in InaccessibleVisibleData$

grd5 : $f = benignfill$

then

act1 : $DestinationObject := DestinationObject \cup \{f(sourceDataElement)\}$
The source element is added to the destination after applying the f behaviour function

act2 : $Acquired := Acquired \Leftarrow \{(sourceDataElement \mapsto yes)\}$
Mark source element as having been acquired

act3 : $WhatAcquired := WhatAcquired \Leftarrow \{(sourceDataElement \mapsto f(sourceDataElement))\}$
Log the mapped value

end

END

Figure 4: The First Refinement of the Context and Specification of the Disk Data Acquisition Tool (Continued).

By contrast, the original *Acquisition* event is now refined to be able to deal with all the other types of accessible data on the source. These include both visible and hidden data. We now arrive at the interesting part of this model, which is the definition of accuracy:

$$\forall x \cdot (x \in DigitalSource \wedge AcquisitionTerminated = yes) \Rightarrow$$
$$(WhatAcquired(x) = (\lambda y \cdot y \in DigitalSource | y)(x)) \vee$$
$$(x \in InaccessibleVisibleData \wedge (WhatAcquired(x) = benignfill(x)))$$

This definition states that once acquisition has terminated, then what has been acquired, in terms of data elements in the digital source, is either the same data obtained by applying the identity function (first part of the right-side of the logical implication), or a benign representation of that data obtained by applying the *benignfill* function (second part of the right-side of the implication). This definition is suitable for both cases of accessible and inaccessible data, this is thanks to the presence of suffecient detail in the refined model. It does also highlight the need to treat accuracy more delicately than one would expect due to this differntiation between accessible and inaccessible data in NIST's requirements.

Our model above considers the visibility and accessibility aspects of the acquired data, which is a common property of most digital storage media with configurable sectors. Hidden sectors are abstractions of Device Configuration Overlays (DCOs) and Host Protected Area (HPAs) on a digital source. Data resident in such areas are not visible to an application and cannot be read (therefore cannot be acquired). In the case of logical block addressing on a hard disk for example, the presence of hidden sectors on the disk will result in the setting of the MAX LBA ADDRESS variable to a value less than that of the ACTUAL MAX LBA variable. The difference between the two is the hidden area. Ideally, a data acquisition tool should reconfigure the digital source drive such that the drive allows access to the hidden sectors by resetting the MAX LBA ADDRESS to the ACTUAL MAX LBA. This will allow access to the entire drive and remove any existing hidden areas. Nonetheless, this introduces a (legal) side issue about as to what to do after the acquisition: whether to reset the MAX LBA ADDRESS variable to its original value (i.e. re-introduce the hidden sectors) or not, since this reconfiguration may be considered as an alteration of evidence. Therefore, in our model above, we consider the general case where hidden areas are allowed to exist to avoid such legal issues and assume that the specific case where

no hidden areas exist to be simply equivalent to a machine with no axiom $HiddenData \neq \varnothing$.

As mentioned in the introduction, the model only considers accessible hidden data. A further refinement of this machine would be to consider the case where hidden data has also an inaccessible part. This can easily be included by adding an additional data set *InAccessibleHiddenData* to refer to the inaccessible element of the hidden sectors.

7. Second Refinement: Additional Features

The second and final refinement of the data acquisition model is aimed at expressing the optional requirements of **DI-RO-16** and **DI-RO-18** in NIST's specification. This refinement introduces a new context, as shown in Figure 5, which includes the definition of a hash function as well as an axiom (axm2) on its uniqueness property. This is then captured in the refined machine by a *BlockHash*, which maps every subset of the source data to its hash value.

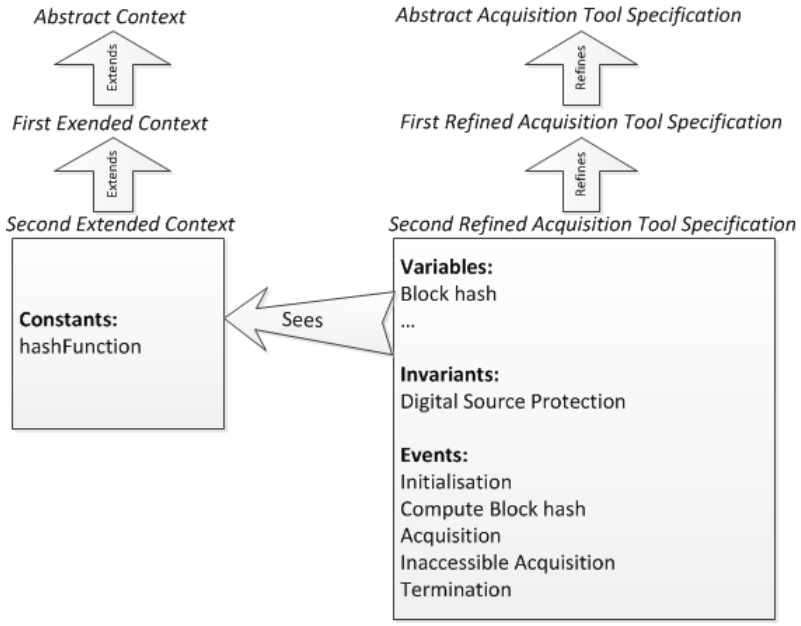

Figure 5: A Representation of the Second Extended Context and Refined Machine of the Disk Data Acquisition Tool.

Specification of this second refined machine and context is shown in Figure 6. A new event is introduced, *ComputeBlockHash*, which is run before the acquisition takes place. This is necessary in order to preserve the integrity of the source data before even the acquisition has commenced. Once this is done, the rest of the second refinement machine behaves similarly to the previous first refinement machine. The inclusion of the hashing functionality so far implements **DI-RO-16** of NIST's optional requirements. It is worth pointing out here that the current level of abstraction of this machine adopts the idealistic definition of hash functions, i.e. that they should be collision free. However, in a different refinement path where different machines geared towards error-prone implementations of hashing (with some probabilistic estimates of their collision rates), one would be able to further express other qualitative aspects of data acquisition tools that would be based on the quality of the hashing algorithm implemented by the tool. Again, we consider this level of detail to be out of the scope of the paper but an interesting direction for future research.

Finally, **DI-RO-18** is maintained by declaring the *DigitalSource* being acquired throughout the three refinement levels as a *constant* in the context of the machine. This implies that any *correct* implementation of the tool based on the model set out here (at any level of abstraction) will not be able to (and should not) modify the *DigitalSource* constant, thereby enforcing **DI-RO-18**. Another alternative for ensuring that the source is not modified during the acquisition process is comparing the block hash value (i.e. $BlockHash(x)$) computed at the beginning of the process with the actual hash value of the source (i.e. $hashFun(x)$). This can be expressed as the following invariant:

$$\forall x \cdot x \in \mathbb{P}(DigitalSource) \Rightarrow$$
$$((StartAcquisition = yes) \Rightarrow (hashFun(x) = BlockHash(x))$$

Note that this invariant holds from the point in time when the acquisition process starts onward. The fact that $BlockHash(x)$ remains unchanged throughout the machine lifecycle is reflected in its equality with its value computed by the event *ComputeBlockHash*, otherwise, any such changes would violate the equality $hashFun(x) = BlockHash(x)$. This requirement is interesting in that it implements the comparison of the value of the digital source pre- and post-acquisition using hash functions. Another choice would be to copy the pre-acquisition value to a reference copy and then compare this to the post-

CONTEXT SecondExtendedContext
EXTENDS FirstExtendedContext
CONSTANTS

$hashFun$

AXIOMS

axm1 : $hashFun \in \mathbb{P}(DigitalSource) \rightarrow \mathbb{N}_1$
axm2 : $\forall x, y \cdot (x \in dom(hashFun) \land y \in dom(hashFun)) \Rightarrow (hashFun(x) = hashFun(y) \Rightarrow (x = y))$

END

MACHINE SecondRefinedAcquisitionToolSpecification
REFINES FirstRefinedAcquisitionToolSpecification
SEES SecondExtendedContext
VARIABLES

$DestinationObject$ The destination object where the data is placed
$AcquisitionTerminated$ Variable denoting the termination of the acquisition process
$Acquired$ Variable to express whether an element in the source has been acquired or not
$WhatAcquired$ Variable denoting what value the source element has been mapped to
$StartAcquisition$ Variable to control the start of the acquisition process
$BlockHash$ Variable that maps every subset of the Digital Source data to its hash value

INVARIANTS

StartAcquisitionType : $StartAcquisition \in Termination$
BlockHashType : $BlockHash \in \mathbb{P}(DigitalSource) \rightarrow \mathbb{N}$

EVENTS
Initialisation
extended
begin

act1 : $DestinationObject := \varnothing$
Destination object is empty initially
act3 : $AcquisitionTerminated := no$
Initially, acquisition process has not terminated
act5 : $Acquired : |(dom(Acquired') = DigitalSource) \land (ran(Acquired') = \{no\})$
Initially, every element in the source object is not acquired yet
act6 : $WhatAcquired : |(dom(WhatAcquired') = DigitalSource) \land (ran(WhatAcquired') = \{null\})$
Initially, source elements have no acquired values in the destination
act7 : $BlockHash : |\forall x \cdot x \in \mathbb{P}(DigitalSource) \Rightarrow BlockHash'(x) = 0$
act8 : $StartAcquisition := no$

end

Figure 6: The Second Refinement of the Context and Specification of the Disk Data Acquisition Tool.

Event *ComputeBlockHash* $\widehat{=}$

when

grd1 : $StartAcquisition = no$
grd2 : $AcquisitionTerminated = no$

then

act2 : $BlockHash := \{x \mapsto y | x \in \mathbb{P}(DigitalSource) \land y = hashFun(x)\}$
act1 : $StartAcquisition := yes$

end

Event *Acquisition* $\widehat{=}$

extends *Acquisition*

any

sourceDataElement Pick some element in the source object
f

where

grd3 : $AcquisitionTerminated = no$
Check if acquisition has terminated or not
grd2 : $Acquired(sourceDataElement) = no$
such that the element has not been acquired yet
grd4 : $f \in DigitalSource \rightarrow Data$
grd5 : $sourceDataElement \notin InaccessibleVisibleData$
grd6 : $f = (\lambda x \cdot x \in DigitalSource | x)$
grd7 : $StartAcquisition = yes$

then

act1 : $DestinationObject := DestinationObject \cup \{f(sourceDataElement)\}$
The source element is added to the destination after applying the f behaviour function

act2 : $Acquired := Acquired \Leftarrow \{(sourceDataElement \mapsto yes)\}$
Mark source element as having been acquired
act3 : $WhatAcquired := WhatAcquired \Leftarrow \{(sourceDataElement \mapsto f(sourceDataElement))\}$
Log the mapped value

end

Event *Termination* $\widehat{=}$

extends *Termination*

when

grd1 : $\forall x \cdot x \in DigitalSource \Rightarrow Acquired(x) = yes$
grd2 : $AcquisitionTerminated = no$
grd3 : $StartAcquisition = yes$

then

act1 : $AcquisitionTerminated := yes$

end

Figure 6: The Second Refinement of the Context and Specification of the Disk Data Acquisition Tool (Continued).

Event $InaccessibleAcquisition \mathrel{\widehat{=}}$
extends $InaccessibleAcquisition$

any

$sourceDataElement$ Pick some element in the source object
f

where

grd3 : $AcquisitionTerminated = no$
Check if acquisition has terminated or not
grd2 : $Acquired(sourceDataElement) = no$
such that the element has not been acquired yet
grd4 : $f \in DigitalSource \rightarrow Data$
grd1 : $sourceDataElement \in InaccessibleVisibleData$
grd5 : $f = benignfill$
grd6 : $StartAcquisition = yes$

then

act1 : $DestinationObject := DestinationObject \cup \{f(sourceDataElement)\}$
The source element is added to the destination after applying the f behaviour function

act2 : $Acquired := Acquired \lhd\!\!+ \{(sourceDataElement \mapsto yes)\}$
Mark source element as having been acquired

act3 : $WhatAcquired := WhatAcquired \lhd\!\!+ \{(sourceDataElement \mapsto f(sourceDataElement))\}$
Log the mapped value

end

END

Figure 6: The Second Refinement of the Context and Specification of the Disk Data Acquisition Tool (Continued).

acquisition value of the source. However, this choice renders the definition of the invariant (and consequently the requirement) cyclic. Instead, the current definition simply reduces the correctness of the requirement to the correctness of the hash function adopted. The invariant, however, is based on the ideal view that hash functions (in this case *hashFun*) are collision-free.

8. Conclusion and Future Work

This paper presented a formal specification based on Event-B of the data acquisition functionality of digital forensics tools. The specification defined three levels of abstraction; the first level is the most abstract and does not distinguish in the accessibility of acquired data, the second includes a clear distinction between accessible, non-accessible and hidden data. The third level includes detail about the integrity of the acquisition process.

One of the advantages of using formal methods techniques is the ability to express good properties of the specification as proof obligations. Discharging proof obligations guarantees model consistency throughout the refinement process. In our case, the successful discharging of the proof obligations performed with the help of the Rodin tool, for the accuracy and completeness properties, revealed that accuracy, unlike completeness, is not a general property that can be specified, reasoned on and talked about in a uniform manner. The validity of the accuracy property is closely coupled with the accessibility property of the acquired data, and can hold a different meaning depending on whether the acquired data is accessible or not.

The refinement methodology (where Event-B is an example of) allows detail to be included in the model to as much precision as needed by the system and its context. Therefore the above three levels of abstraction are by no means an exhaustive definition of how data acquisition is performed with real tools. For example, one important aspect of this acquisition that we do not consider here is the type of the interface access used by the digital source, such as whether this is BIOS or Direct. This differece in the interface type may cause different data acquisition tools to interpret the number of sectors in the digital source differently, hence impacting the definition of the machine variable *DigitalSource* in our model. Further refinement machines could take this additional detail into consideration.

There are several other directions for future research based on the results of this paper. First, it is important to extend the existing model to deal with the rest of the NIST requirements on data acquisition NIST (2004), as

this may reveal further interesting results. For example, additional detail related to the types of digital source interfaces and the presence or not of hidden sectors and their accessibility have all the potential for studying other interesting aspects and properties of data acquisition tools by introducing specialised refinements that will deal with this level of detail. For the scope of this paper, we consider these as interesting areas of future work.

Similar to NIST's testing plans NIST (2005), one of the main other advantages of adopting an automated formal framework such as Event-B is that it is also possible to generate test cases based on the specifications describing the modelled system (see for example Malk et al. (2009)). This will provide the possibility in the future for generating test cases for digital forensics tools based on robust methods. Rodin also provides a plug-in called MBT (Model-Based Testing) for the generation of execution paths, counter paths and test suites for the model. This means that the testing methodology can be more rigorous than for example the testing plan of NIST NIST (2005). The test suite generation provides automatic algorithms for different strategies including minimum size of test suite, minimum number of executed events, minimum length of longest execution path, maximum distribution quality and balanced cases between other path lengths and minimum longest path's length. The MBT tool can also provide a counter example involving any number of the events specified in the model.

Finally, we plan to consider other digital forensics tools covered by the NIST testing project, such as deleted file recovery and storage media preparation.

9. Acknowledgements

The author would like to thank the anonymous reviewers for their time and effort in providing valuable feedback and constructive comments to the initial versions of this paper. Many thanks to Geoff Hamilton for early comments on this work.

References

Abrial JR. The B Book. Cambridge University Press, 1996.

Abrial JR. Modeling in Event-B: System and Software Design. Cambridge University Press, 2010.

Abrial JR, Butler M, Hallerstede S, Hoang TS, Mehta F, Voisin L. Rodin: an open toolset for modelling and reasoning in Event-B. STTT 2010;12(6):447–66.

Abrial JR, Cansell D, Méry D. Refinement and reachability in event$_{\mathrm{b}}$. In: ZB. Springer; volume 3455 of *Lecture Notes in Computer Science*; 2005. p. 222–41.

Abrial JR, Hallerstede S. Refinement, Decomposition, and Instantiation of Discrete Models: Application to Event-B. Fundamenta Informaticae 2007;77(1-2):1–28.

Beckett J, Slay J. Digital forensics: Validation and verification in a dynamic work environment. In: Proceedings of the 40th Annual Hawaii International Conference on System Sciences. IEEE Computer Society; HICSS '07; 2007. p. 266a–.

Beebe N, Clark JG. A hierarchical, objectives-based framework for the digital investigations process. Digital Investigation 2005;2(2):147–67.

Carrier B. A Hypothesis-Based Approach to Digital Forensic. Ph.D. thesis; Purdue University; 2006.

Carrier BD, Spafford EH. An event-based digital forensic investigation framework. In: Proceedings of the 4th Digital Forensic Research Workshop. DFRWS'04; 2004. .

Casey E. Digital Evidence and Computer Crime Forensic Science, Computers and the Internet 3rd Ed. Elsevier, 2011.

Casey E, Rose C. Forensic Discovery: Handbook of Digital Forensics and Investigation. Academic Press, 2010.

Ciardhuáin SO. An extended model of cybercrime investigations. IJDE 2004;3(1).

Cohen F. Digital Forensic Evidence Examination. Fred Cohen & Associates, 2009.

Enbacka A. Formal methods based approaches to digital forensics. Master's thesis; Åbo Akademi University; 2007.

Enbacka A, Laibinis L. Formal specification and refinement of a write blocker system for digital forensics. 2005.

Friedberg S. Report of digital forensic analysis in: Paul d. ceglia v. mark elliot zuckerberg, individually, and facebook, inc. 2012.

Garfinkel S, Farrell P, Roussev V, Dinolt G. Bringing science to digital forensics with standardized forensic corpora. Digital Investigation 2009;6:2–11.

Gladyshev P. Finite state machine analysis of a blackmail investigation. IJDE 2005;4(1).

Gladyshev P, Enbacka A. Rigorous development of automated inconsistency checks for digital evidence using the b method. IJDE 2007;6(2).

Guo Y, Slay J, Beckett J. Validation and verification of computer forensic software tools-searching function. Digit Investig 2009;6:S12–22.

Ieong RSC. Forza - digital forensics investigation framework that incorporate legal issues. Digital Investigation 2006;3(Supplement-1):29–36.

James J, Gladyshev P, Abdullah MT, Zhu Y. Analysis of evidence using formal event reconstruction. In: ICDF2C. Springer; volume 31 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*; 2009. p. 85–98.

Leigland R, Krings AW. A formalization of digital forensics. IJDE 2004;3(2).

Malk QA, Lilius J, Laibinis L. Scenario-based test case generation using event-b models. In: Proceedings of the 2009 First International Conference on Advances in System Testing and Validation Lifecycle. Washington, DC, USA: IEEE Computer Society; VALID '09; 2009. p. 31–7.

Mazza E, Potet ML, Le Métayer D. A formal framework for specifying and analyzing logs as electronic evidence. In: Proceedings of the 13th Brazilian conference on Formal methods: foundations and applications. Berlin, Heidelberg: Springer-Verlag; SBMF'10; 2011. p. 194–209.

Métayer C, Abrial JR, Voisin L. Event-B Language. Rodin Deliverable D3.2; 2005.

Métayer DL, Maarek M, Mazza E, Potet ML, Frénot S, Tong VVT, Craipeau N, Hardouin R. Liability issues in software engineering: the use of formal methods to reduce legal uncertainties. Commun ACM 2011;54(4):99–106.

Mosses PD. Action semantics. In: ADT. 1986. .

NIST . Software Write Block Tool Specification and Test Plan (v3.0). Technical Report; NIST; 2003.

NIST . Digital Data Acquisition Tool Specification (v4.0). Technical Report; NIST; 2004.

NIST . Digital Data Acquisition Tool Test Assertions and Test Plan (v1.0). Technical Report; NIST; 2005.

NIST . Computer forensics tool testing (cftt) project web site. http://www.cftt.nist.gov/.

Rekhis S, Boudriga N. A formal logic-based language and an automated verification tool for computer forensic investigation. In: Proceedings of the 2005 ACM symposium on Applied computing. New York, NY, USA: ACM; SAC '05; 2005. p. 287–91.

Rekhis S, Boudriga N. Formal digital investigation of anti-forensic attacks. In: Proceedings of the 2010 Fifth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering. Washington, DC, USA: IEEE Computer Society; SADFE '10; 2010. p. 33–44.

Rekhis S, Boudriga N. A hierarchical visibility theory for formal digital investigation of anti-forensic attacks. Computers & Security 2012;31(8):967–82.

Shamala P, Azizah AM. Digital Computer Forensic: Validation and Verification for Disk Imaging: A Comprehensive Validation and Verification (V&V) Disk Imaging Model for Court of Law Admissibility. LAP LAMBERT Academic Publishing, 2012.

Stephenson P. Modeling of post-incident root cause analysis. IJDE 2003;2(2).

Watt DA. An action semantics of standard ml. In: MFPS. Springer; volume 298 of *Lecture Notes in Computer Science*; 1987. p. 572–98.