

# CSI Techniques for Software Requirements and Analysis

**Thomas A. Bullinger**  
President  
ArchSynergy, Ltd.  
Victor, New York

**Sandeep Mitra**  
Associate Professor  
SUNY Brockport  
Brockport, New York

## Class ESC-441

### Abstract

Traditional requirements gathering techniques often result in hundreds of textual requirements. The nature of text-based requirements makes them difficult to organize and reconcile. This class presents software requirements and analysis techniques that result in a complete understanding of the behaviors and structure of the problem to be solved. The requirements and analysis process introduces the metaphor of a detective investigating a crime and gathering evidence. The observations and evidence are used to create a UML model of the problem.

### Introduction

From Sherlock Holmes to Inspector Jacques Clouseau, contemporary fiction dedicates an entire genre to the detective story. The detective story is of particular fascination for us as humans. Our fascination with the detective story appears rooted in our innate human curiosity. It leads us into investigations of the “truth” based on scant clues and fleeting evidence. As if the curiosity isn’t enough, the detective genre explores the range of human experience within the context of the mystery. For most of us, the contemporary version of the detective novel plays out in the hours of prime time television every evening.

The challenge of a detective mystery is, in part, the satisfaction (or at least the potential of satisfaction) of solving the mystery before the author of the story reveals the solution. There are few experiences more satisfying than figuring out the solution before the protagonist of the story.

Of interest in our context is the comparison of the activities of the detective and that of the software engineer engaged in gathering requirements. To understand the need for this comparison, we first refer to an ongoing study of software projects conducted by the Standish Group<sup>1</sup>. According to their ongoing CHAOS study, nearly 70% of software projects are cancelled, are late, or fail to meet the expectations of those who provide the funding. The CHAOS study attributes much of the failure rate to a lack of understanding of the problem, or poor requirements. To address this serious shortfall in most projects, we suggest that echoing

---

<sup>1</sup> <http://www.standishgroup.com>

# CSI Techniques for Software Requirements and Analysis

the activities of a detective solving a crime will go a long way toward providing a more rigorous and effective means of gathering and modeling requirements. To better understand our comparison, let's take a look at the activities of a typical detective and compare and contrast them to those of a software engineer gathering and modeling requirements.

## The Detective

A typical detective story follows a specific format or pattern. It begins with the commission of a crime, usually a murder or theft of large proportions. The details of the crime are rarely presented, and may be left completely out of the story. The lack of details allow the detective to reveal the particulars of the crime over the course of the telling.

Of interest to us are the activities of the detective as they go about solving a crime, and how their methods might be applied to the gathering and modeling of requirements for a software application. We suggest that the information gathering techniques used by a detective apply equally to both domains. While software requirements gathering remains a relatively new discipline in software engineering, the detective has been at work for hundreds, if not thousands of years refining their techniques of investigation and deduction.

The definition of the word detective is "a person, especially a police officer, who's occupation is to investigate and solve crimes."<sup>2</sup> The role of the detective is to find and interpret the evidence through deductive and inductive reasoning to ascertain the truth behind a crime.

The protagonist of any detective story, the detective, begins with a survey of the crime scene at which time the physical evidence is examined, tagged and bagged, and may be returned to a lab or morgue for further analysis. The suspects and witnesses, if any, are identified and interviewed by the detective in a first attempt to ascertain the details of the crime.

Information gathered at the initial crime scene invariably leads the detective to other people and places, each of which are visited in an attempt to understand the people involved in the crime, and their behavior relative to the crime.

Given the initial stories of the suspects and witnesses and the physical evidence of the crime scene, the detective weaves the information together into a self-consistent hypothesis of the crime and its commission. As the story progresses, it follows the detective through the gathering of additional evidence and testimony, and the logical interpretation of the

---

<sup>2</sup> From the Dictionary widget provided with Apple's OS X, version 10.4.

# CSI Techniques for Software Requirements and Analysis

information to prove or disprove the hypotheses of the crime. As the conclusion of the tale approaches, the detective has followed each lead to its logical conclusion and gathered all the available information. Based on his hypotheses, the detective creates a mental model of the crime detailing the behaviors of the persons involved. The model of the crime is turned over to the legal authorities for prosecution in a court of law.

## The Software Engineer

The definition of the word engineer is “a person who design, builds or maintains engines, machines, or public works.”<sup>3</sup> Adding software to the mix, we can define a software engineer as “a person who designs, builds or maintains software.”

The role of the detective and the role of the software engineer appear to have no relationship whatsoever. However, we suggest that the relationship between the detective and the software engineer is based not in *what* they each do, but rather in their methodology: *how* they go about doing it.

Despite what software tool vendors and traditional computer science texts may tell you, software development is NOT about the application of technology (.NET, J2EE, etc.), but rather about *solving an existing problem for the user of the application*. The job of the software engineer is to understand the problem to be solved, then automate a solution to this problem using the appropriate technology. A software application, and the computer it runs on, is nothing more than a tool to automate and streamline the activities of an existing problem. The recent success of Apple’s iPod highlights this philosophy.

MP3 music players have been on the market for several years, some more capable than others. Prior to the introduction of the iPod, MP3 players were designed around a technology: a compact storage medium and the ability to store and play music on that medium. While an interesting display of technology, the pre-iPod devices did not solve a problem: to play music in a portable manner. They did not understand the workflow of the user of the device. A typical MP3 device is plugged into a computer and the user rips and downloads songs to the device manually. Once the songs are on the device, a series of menus and options are navigated to play the music. On one product, the device actually plays a complex animation when turned on and off, forcing the user to wait while watching an insipid display of moving pixels for the 900th time. Products of this nature are more common than not, and don’t serve to solve a problem.

---

<sup>3</sup> From the Dictionary widget provided with Apple’s OS X, version 10.4.

# CSI Techniques for Software Requirements and Analysis

The folks at Apple looked at the potential for a portable music device and began their product development effort by understanding the problem to be solved. Apple developed a device that solves a specific problem for the user: the iPod plays music anywhere. Transferring music files to the iPod is transparent to the user: just plug the device into your computer and any music on your Mac or PC is automatically transferred to your iPod. Once transferred, the user interface on the iPod is easy to navigate to play songs in different ways. Apple even went so far as to pause the music should your headphone jack inadvertently pull out of the device! The result is a product that arrived on the scene late, yet still captures greater than 70% of the market for portable music players. A remarkable achievement in any business segment.

The message conveyed by this example is that the challenge of the software engineer is to NOT apply technology for the sake of technology, but to understand the problem to be solved first, then to apply the technology in an appropriate fashion to solve the problem.

Unfortunately, the example of the iPod is the exception for a software-based product. In fact, most software developed today fails to meet the needs of the target market of the product. This is certainly consistent with the market for portable music devices.

**We now suggest that the contemporary software developer commits a crime every time they attempt to foist an ill-considered application on an unsuspecting user.**

To counter the trend of technology-based software development, the metaphor of the detective is introduced to help refocus the efforts of the software engineer on the problem to be solved, and to ignore the application of technology until appropriate. The goal of the detective is to create an accurate model of the crime based on the evidence and testimony. The goal of the successful software engineer is to create an accurate model of the problem based on the evidence and testimony from the existing problem domain.

After the need for an application is established, the role of the software engineer is to first understand the problem to be solved. This is accomplished by visiting the scene where the proposed application will be used and observing the existing environment and processes. The information gained from the initial site visit leads to other domains and stakeholders, and each of the leads must be followed to ensure a complete, integrated solution. Based on the observations gathered from various scenes, the software engineer builds a model of the problem that describes the required behaviors within the context of the problem domain, and a model of the information consumed, created, or manipulated by the processes.

# CSI Techniques for Software Requirements and Analysis

A comparison of the two methodologies reveals significant overlap. While the activities are different, the intention behind each step remains consistent. The following table represents each step in the process and the activities of the detective and the software developer:

Activity	Detective	Software Engineer
Report the Crime	The detective assesses the crime scene to determine the scope and affect of the investigation.	The initial need is identified and a business case is prepared to ensure the problem is solvable.
Survey the Scene	Visit the scene and gather clues, evidence and testimony.	Visit the scene and gather clues, evidence and testimony.
Follow Leads	Follow leads to other scenes, witnesses and suspects ensuring a complete understanding of all the participants involved or affected by the crime.	Follow leads to other stakeholders, gathering information that considers the needs of everyone in the problem domain.
Create Hypotheses	Create a hypothetical model of the crime exploring how the information ties together into a consistent, cohesive model	Capture the behaviors of the current participants in the problem domain.
Build a Case	Create a case based on the information that is self-consistent and explains the evidence and observations gathered.	Create a model of the problem describing the complete behaviors, information, and structure of the problem domain.

A successful software engineer plays the role of the detective, particularly in the early stages of product development. The fundamental difference between the activities of the detective and the software engineer, is that the detective is investigating a crime that has already been committed. The detective has the luxury of dealing with evidence, witnesses and suspects *after* the commission of the crime. Our software engineer, on the other hand, can be considered to be investigating the “crime” before its commission.

Stated simply, the software engineer gathers information as to what the “crime” should be, then creates the “crime” to fit the evidence and testimony. As this is the reverse of the process

# CSI Techniques for Software Requirements and Analysis

employed by a traditional detective, we coin the title “The Reverse Detective” for our software requirements and analysis process.

Now that we better understand the value of leveraging the process of the detective, let’s take a look at the methods of the detective in greater detail.

## The Detective’s Process

In January 2000, the United States Department of Justice published a guideline for crime scene investigation<sup>4</sup>. The information in the guideline was prepared by a technical working group on crime scene investigation. The guideline outlines the procedures and practices recommended for the appropriate investigation of a crime scene.

The guidelines are broken into five sections, each addressing a portion of the crime scene response. The sections are:

- Arriving at the Scene: Initial Response / Prioritization of Efforts
- Preliminary Documentation and Evaluation of the Scene
- Processing the Scene
- Completing and Recording the Crime Scene Investigation
- Crime Scene Equipment

Each section addresses the principle, policy and procedure for the responders to the crime scene. The procedure is presented as a series of steps to take, or issues to be aware of. The process a detective follows as outlined in the DOJ white paper seeks to answer a series of questions related to the commission of the crime, while ensuring the integrity of the available evidence. As we consider the activities of the detective in the context of a software engineer gathering requirements and analyzing the problem, some of the issues outlined in the paper can be safely ignored. For example, emergency care of victims and the safety of the investigators are unlikely to be of concern to the person gathering requirements. Security of the evidence is similarly not an issue. Examining the remaining process elements, we can extract the attributes of the detective process that remain of interest to the software developer. The attributes of interest in the process used by the detective are:

- Comprehensive observation - No information is considered irrelevant.
- Well-defined scope - The crime scene boundaries are clearly defined with physical barriers.
- Everything is logged - A comprehensive log of observations and activities is maintained.

---

<sup>4</sup> Crime Scene Investigation: A Guide for Law Enforcement,

# CSI Techniques for Software Requirements and Analysis

- Physical evidence is gathered - Relevant artifacts related to the crime are gathered for later examination.
- Information is related - As disparate information is gathered by the detective, it is tied together into a cohesive model of the crime.
- A story is created - Observations and evidence are used to infer the behaviors of the participants of the crime.
- Chain of custody - The observations and evidence are traceable from their origin through the entire process to ensure the completeness and consistency of the case file.

Mapping the desired attributes above into a well-defined process for the detective, we suggest the following steps:

- Report the Crime - A perpetrator commits a crime at one or more locations, involving themselves and potentially others. Before, during or after the commission of the crime, physical evidence related to the crime is left behind. There are also potential witnesses to the crime. The crime is discovered and reported to the authorities.
- Survey the Scene - The detective, on learning of the crime (or being assigned to the case), visits the location where the crime occurred and determines the scope of the crime scene. He interviews potential witnesses capturing their stories for later review. He examines and records the physical evidence at the scene of the crime looking for clues related to the crime. The detective records the physical evidence and extracts the evidence for safekeeping or further analysis. He observes the environment of the crime, noting the context in which the crime occurred as supporting information for the subsequent development of hypotheses.
- Follow Leads - The information gathered at the scene may lead to other areas of investigation. Each of the leads are followed to their logical conclusion to gather as much information as possible. The pursuit of each lead may reveal further evidence, and may lead to other crime scenes, witnesses and related testimony. The additional information is added to the case file.
- Develop Hypotheses - During the scene investigation or the subsequent analysis of the information, the detective develops one or more hypotheses describing the commission of the crime. The detective suggests the behavior of the suspects and witnesses relative to the crime and links the evidence and the behavior into a plausible hypothesis of the commission of the crime consistent with the evidence and testimony. Using experience and intuition, he tests the hypothesis of the crime against the known facts and identifies evidence and testimony that does not fit the hypothesis, or cannot be proven. The detective, having discovered potential inconsistencies or holes in the hypothesis, then gathers further evidence to prove or disprove the hypothesis.
- Build the Case - When the detective is satisfied the hypotheses are justifiable, he creates a model of the crime. The model of the crime presents a self-consistent story that is used to prosecute the accused in a court of law. If the evidence is not sufficient or self-consistent, the case against the accused may be deferred until more information is available, or summarily closed.

# CSI Techniques for Software Requirements and Analysis

## The Software Engineer's Process

Like the detective, the software engineer has a proscribed methodology to follow while developing an application. Each developer has their own unique tricks and techniques, but there is a fundamental basis for everything the engineer must do. To facilitate an understanding of how the software engineer works, we start with an overview of one such methodology, the Rational Unified Process<sup>5</sup> (RUP). For our overview, we turn to Wikipedia<sup>6</sup>, the Internet-based public encyclopedia.

The overview of RUP provided by Wikipedia discusses a series of phases a software engineer must work through while developing an application. The RUP phases are:

- Inception,
- Elaboration,
- Construction, and
- Transition.

Each phase discusses the activities appropriate for that phase, the success criteria, and milestones. The following is extracted from the description found at the Wikipedia entry for RUP<sup>7</sup>

### ***The Inception Phase***

In this phase the business case, which includes the application context, success factors (expected revenue, market recognition, etc), and financial forecast is established. To complement the business case, a basic use case model, project plan, initial risk assessment and project description (the core project requirements, constraints and key features) are generated. After these are completed, the project is checked against the following criteria:

- Stakeholder concurrence on scope definition and cost/schedule estimates.
- Requirements understanding as evidenced by the fidelity of the primary use cases.
- Credibility of the cost/schedule estimates, priorities, risks, and development process.
- Depth and breadth of any architectural prototype that was developed.
- Actual expenditures versus planned expenditures.

---

<sup>5</sup> Booch, Jacobson, Rumbaugh. "The Unified Modeling Language User Guide", Addison-Wesley, 1999

<sup>6</sup> <http://en.wikipedia.org>

<sup>7</sup> [http://en.wikipedia.org/wiki/Rational\\_Unified\\_Process](http://en.wikipedia.org/wiki/Rational_Unified_Process)

# CSI Techniques for Software Requirements and Analysis

If the project does not pass this milestone, called the Lifecycle Objective Milestone, it can either be cancelled, or can repeat this phase after being redesigned to better meet the criteria.

## ***The Elaboration Phase***

The Elaboration phase is where the project starts to take shape. In this phase the problem domain analysis is performed and the architecture of the project gets its basic form.

This phase must pass another Lifecycle Architecture Milestone by meeting the following criteria:

- A use-case model in which the use-cases and the actors have been identified and most of the use-case descriptions are developed. The use-case model should be 80% complete.
- A description of the software architecture in a software system development process.
- An architecture prototype, which may be executable.
- A business case and risk list which are revised from previous phases.
- A development plan for the overall project.

If the project cannot pass this milestone, there is still time for it to be cancelled or redesigned. After leaving this phase, the project transitions into a high-risk operation where changes are much more difficult, expensive, and detrimental when made.

## ***The Construction Phase***

In this phase the main focus is the development of components and other features of the system. This is the phase when the bulk of the implementation takes place.

This phase produces the first external release of the software.

## ***The Transition Phase***

In the transition phase the product has moved from the development organization to the end user. The activities of this phase include: Training of the end users and maintainers, beta-testing the system to validate it against the end users expectations. The product is also checked against the quality level set in the Inception phase. If it does not meet this level, or the standards of the end-users, the entire cycle in this phase begins again.

The Rational Unified Process outline as described above covers the entire software development lifecycle at a high level of abstraction. For our purposes, we consider only the first portion of the development effort. The Wikipedia description does not include many

# CSI Techniques for Software Requirements and Analysis

details of the process, so we provide our own interpretation within the context of our discussion.

- Inception - Figure out the problem to be solved by the proposed application, and create a business case to justify the investment. The scope of the application is determined as are the initial required behaviors of the application.
- Elaboration - Dig into the details of the problem, and create a comprehensive model to demonstrate a complete understanding. Validate the model of the problem with the parties interested in the application.
- Construction - Choose a suitable technology and design a solution consistent with the problem model. Implement the design in the chosen technologies.
- Transition - Deliver the solution to the customer and integrate it into their existing workflow.

## A Pragmatic Process

The Rational Unified Process is an excellent starting point for a development process. However, for our purposes, a lesser degree of ceremony is desired. Specifically, we choose to translate the RUP process outline into an actionable process or *recipe* for developing software. A suitable process is prescriptive in nature, spelling out the steps in sufficient detail to ensure a novice can partake in the excitement of a well-ordered development process. Integrating the concepts and philosophy of a detective at work, we suggest the following steps for the initial requirements gathering and analysis process for developing software:

- Identify a need or a problem to be solved
- Survey the domain of the problem
- Follow leads to complete the survey
- Capture the required behaviors or workflow
- Model the problem to be solved

The process outlined above leaves much to the imagination. For example, the entire validation and verification of the work products has been ignored. Regardless, the steps above provide placeholders for our ensuing discussion. Let's describe each of the steps in more detail.

- Identify a Need - The process of developing software begins with identifying the need for a software application. The need may be recognized in a variety of ways, but can usually be traced back to the need to automate an existing workflow or solve an existing problem. As software is only a tool for automation, the source of information will always be an existing problem or workflow. Before any other time or effort can be invested, the need must be identified and described. Once the need is identified, the scope of the

# CSI Techniques for Software Requirements and Analysis

problem must be understood. The initial scope is determined by drafting a brief description of the problem. The scope is further clarified by finding each stakeholder and determining their interest, or stake in the application. Stakeholders include the users of the proposed application as well as the suppliers or receivers of information, raw or processed materials, regulatory agencies, or financiers. Given a well-defined scope for the problem to solve, a business case can be created to determine whether the development effort is worth the potential return on the investment.

- **Survey the Scene** - Once a need is identified and the initial scope determined, the problem domain is visited. The people and systems involved in the problem are observed and the behaviors associated with the problem domain captured. The work products, or artifacts of the problem workflow are gathered as samples.
- **Follow Leads** - The initial site survey provides much of the required information, but not all. Each stakeholder identified in the first steps must be interviewed or observed to ensure the proposed application fulfills their needs. Following leads may result in the required analysis of existing (legacy) systems to understand their role in the current workflow. At each subsequent scene, the people and systems are observed and artifacts sampled for later analysis.
- **Model the Problem** - At the completion of the initial survey of the various sites, and after gathering the existing set of artifacts from the problem domain, a model of the problem is built to represent a complete understanding of the problem to be solved. The model of the problem builds on the behaviors captured in the previous steps, and determines the structure of the components of the system (the architecture) and the behaviors of the identified components.

## Tying Everything Together

The process itself is only a starting point for the creation of a usable requirements and analysis model. Both the detective and the software engineer gather immense amounts of information, all of which must be sifted, sorted and categorized. Fortunately, UML provides a variety of artifacts for capturing the gathered information into a cohesive, self consistent model. For more details on building a traceable model for requirements, refer to the Embedded Systems Conference proceedings for my other paper and presentation, “Building a Traceable UML Model.”