Chapter 2
Introducing the Case Study

2.1 Overview

This chapter introduces the case study that will be used in subsequent chapters to illustrate some of the design principles in this book.[1] Very basically, the application is a multiuser software system with a database that is used to share information between users and intelligent tools that aim to help the user complete their work tasks more effectively. An informal context diagram is depicted in Fig. 2.1.

The system has software components that run on each user's workstation, and a shared distributed software "back-end" that makes it possible for intelligent third party tools to gather data from, and communicate with, multiple users in order to offer assistance with their task. It's this shared distributed software back-end that this case study will concentrate on, as it's the area where architectural complexity arises. It also illustrates many of the common quality issues that must be addressed by distributed, multiuser applications.

2.2 The ICDE System

The Information Capture and Dissemination Environment (ICDE) is part of a suite of software systems for providing intelligent assistance to professionals such as financial analysts, scientific researchers and intelligence analysts. To this end, ICDE automatically captures and stores data that records a range of actions performed by a user when operating a workstation. For example, when

[1]The case study project is based on an actual system that I worked on. Some creative license has been exploited to simplify the functional requirements, so that these don't overwhelm the reader with unnecessary detail. Also, the events, technical details and context described do not always conform to reality, as reality can be far too messy for illustration purposes.

I. Gorton, *Essential Software Architecture*,
DOI 10.1007/978-3-642-19176-3_2, © Springer-Verlag Berlin Heidelberg 2011

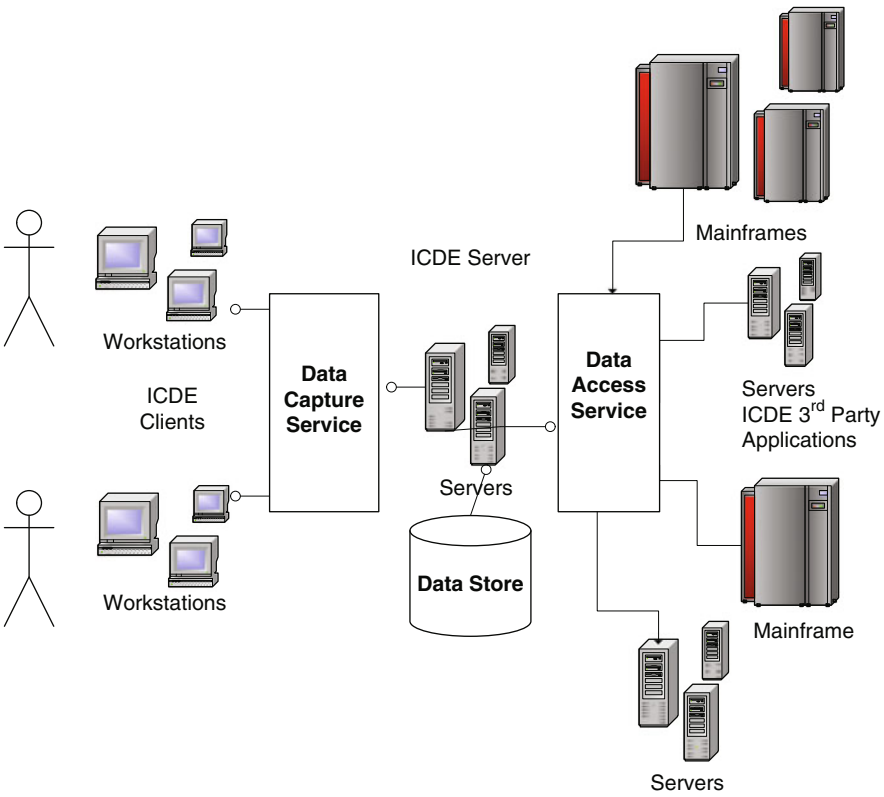

Fig. 2.1 ICDE context diagram

a user performs a Google search, the ICDE system will transparently store in a database:

- The search query string
- Copies of the web pages returned by Google that the user displays in their browser

This data can be subsequently retrieved from the ICDE database and used by third-party software tools that attempt to offer intelligent help to the user. These tools might interpret a sequence of user inputs, and try to find additional information to help the user with their current task. Other tools may crawl the links in the returned search results that the user does not click on, attempting to find potentially useful details that the user overlooks.

A use case diagram for the ICDE system is shown in Fig. 2.2. The three major use cases incorporate the capture of user actions, the querying of data from the data store, and the interaction of the third party tools with the user.

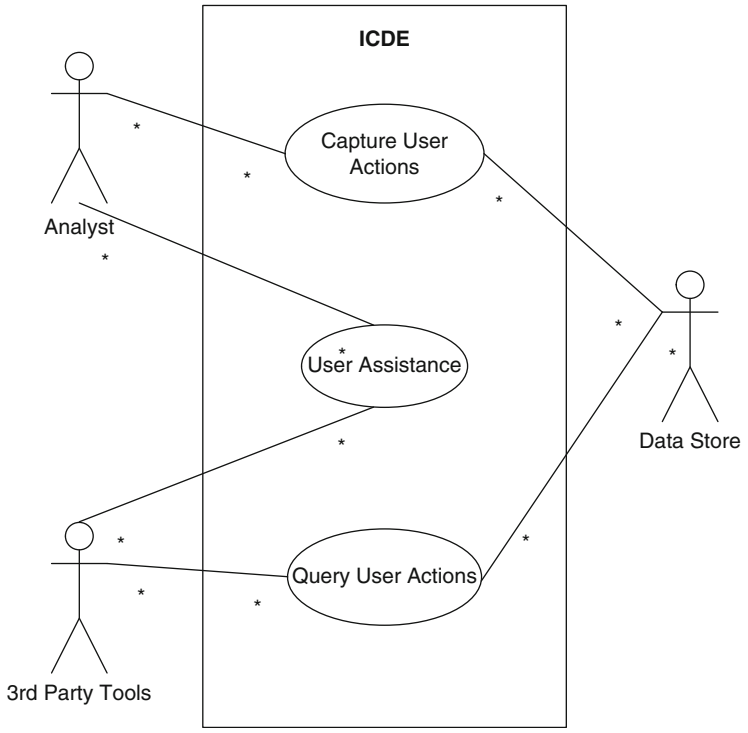

Fig. 2.2 ICDE system use cases

2.3 Project Context

Few real projects are green-field efforts, allowing the design team to start with a clean and mostly unconstrained piece of paper. The ICDE system certainly isn't one of these.

An initial production version (v1.0) of ICDE was implemented by a small development team. Their main aim was to implement the *Capture User Actions* use case. This created the client component that runs on each user workstation, and drove the design and implementation of the data store. This was important as the data store was an integral part of the rest of the system's functionality, and its design had to be suitable to support the high transaction rate that a large number of users could potentially generate.

ICDE v1.0 was only deployed in a small user trial involving a few users. This deployment successfully tested the client software functionality and demonstrated the concepts of data capture and storage. The design of v1.0 was based upon a simple two-tier architecture, with all components executing on the user's workstation. This design is shown as a UML component diagram in Fig. 2.3. The collection and analysis client components were written in Java and access the data store

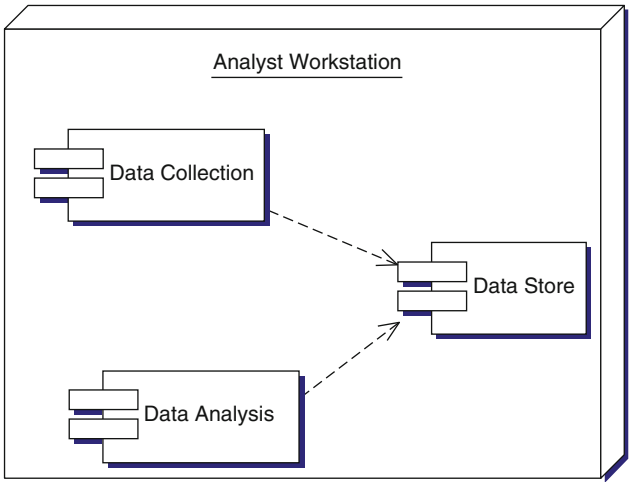

Fig. 2.3 ICDE Version 1.0 application architecture

(server) directly using the JDBC[2] API. The complete ICDE application executed on Microsoft Windows XP.

The role of each component is as follows:

- *Data Collection*: The collection component comprises a number of loosely coupled processes running on a client workstation that transparently track the user's relevant activities and store them in the *Data Store*. The captured events relate to Internet accesses, documents that are opened and browsed, edits made to documents, and some basic windowing information about when the user opens and closes applications on the desktop. Each event has numerous attributes associated with it, depending on event type. For example, a mouse double click has (x, y) coordinate attributes, and a window activation event has the associated application name as an attribute.
- *Data Store*: This component comprises a commercial-off-the-shelf (COTS) relational database. The relational database stores event information in various tables to capture the user activities, with timestamps added so that the order of events can be reconstructed. Large objects such as images on web pages and binary documents are stored as Binary Large Object Fields (BLOBS) using the native database facilities.
- *Data Analysis*: A graphical user interface (GUI) based tool supports a set of queries on the data store. This was useful for testing purposes, and to give the third party tool creators an initial look at the data that was being captured, and was hence available to them for analysis.

[2]Java Database Connectivity.

2.4 Business Goals

ICDE v2.0 had much more ambitious aims. Having proven that the system worked well in trial deployments, the project sponsors had two major business objectives for the next version. These were:

- Encourage third party tool developers to write applications for the ICDE system. For example, in finance, a third party developer might build a "stock advisor" that watches the stocks that an analyst is looking at in their browser and informs them of any events in the news that might affect the stock value.
- Promote the ICDE concept and tools to potential customers, in order to enhance their analytical working environment.

Clearly, both these objectives are focused on fostering a growing business around the ICDE technology, by creating an attractive market for third party tools and an advanced advisory environment for users in a range of application domains. Achieving these goals requires detailed technical and business plans to be drawn up and followed through. From a purely technical perspective, leaving out such activities as sales and marketing, the following major objectives were identified – see Table 2.1:

In order to attract third party tool developers, it is essential that the environment has a powerful and easy-to-use application programming interface (API) that could be accessed from any operating system platforms that a developer chooses to use. This would give tool developers flexibility in choosing their deployment platform, and make porting existing tools simpler. Surveys of existing tools also raised the issue that powerful analytical tools might require high-end cluster machines to run on. Hence they'd need the capability to communicate with ICDE deployments over local (and eventually wide) area networks.

Another survey of likely ICDE clients showed that potential user organizations had groups of 10–150 analysts. It was consequently important that the software could be easily scaled to support such numbers. There should also be no inherent design features that inhibit the technology from supporting larger deployments which may appear in the future.

Table 2.1 ICDE v2.0 business goals

Business goal	Supporting technical objective
Encourage third party tool developers	Simple and reliable programmatic access to data store for third party tools
	Heterogeneous (i.e., non-Windows) platform support for running third party tools
	Allow third party tools to communicate with ICDE users from a remote machine
Promote the ICDE concept to users	Scale the data collection and data store components to support up to 150 users at a single site
	Low-cost deployment for each ICDE user workstation

Equally important, to keep the base cost of a deployment as low as possible, expensive COTS technologies should be avoided wherever possible. This in turn will make the product more attractive in terms of price for clients.

2.5 Constraints

The technical objectives were ambitious, and would require a different architecture to support distributed data access and communications. For this reason, it was decided to concentrate efforts on this new architecture, and leave the client, including the GUI and data capture tools, stable. Changes would only be made to the client to enable it to communicate with the new data management and notification architecture that this project would design. For this reason, the client-side design is not dealt with in this case study.

A time horizon of 12 months was set for ICDE v2.0. An interim release after 6 months was planned to expose tool developers to the API, and allow them to develop their tools at the same time that ICDE v2.0 was being productized and enhanced.

As well as having a fixed schedule, the development budget was also fixed. This meant the development resources available would constrain the features that could be included in the v2.0 release. These budget constraints also influenced the possible implementation choices, given that the number of developers, their skills and time available was essentially fixed.

2.6 Summary

The ICDE application makes an interesting case study for a software architecture. It requires the architecture of an existing application to be extended and enhanced to create a platform for new features and capabilities. Time and budget constraints restrict the possible options. Certainly a redevelopment of the existing ICDE v1.0 client and data store is completely out of the question.

In Chap. 9, the design for the ICDE back-end will be elaborated and explained. The next few chapters aim to provide the necessary background knowledge in designing architectures to meet quality attributes, and exploiting technologies to make the creation of such systems tractable.