

## The Vanilla approach to building Reporting Systems

### Introduction

A typical Vanilla reporting system processes log files from multiple raw data sources and loads the processed data into a database against which a variety of customisable user reports can be run. Such systems provide managers within the business, and selected partners as appropriate, with timely access to key business information.

Vanilla's approach to building reporting systems is distinctive in the following ways:

### Focus on efficiency

We don't just build reporting systems, we build responsive reporting systems. We clean up and process the data before loading it into custom-designed databases against which queries can be run quickly. On top of this, we employ techniques such as query-caching to reduce the load on the database. Users aren't left hanging around for their reports to be generated.

### Focus on usability

We employ an easy-to-use web interface that anyone with access to the company intranet or extranet can get to grips with. We define the reports to be generated, and their associated form elements and output charts, in consultation with the client; and our flexible set of interface-building tools means that these reports can be changed easily, as and when necessary.

### Passionate about accuracy

Our seven years' experience of building reporting systems has taught us a lot about what can go wrong with them. For this reason, we know the right questions to ask when putting the system together, we build in warning systems for possible data errors (when data source formats change, for example), and we rigorously test and re-test our reports against the raw data.

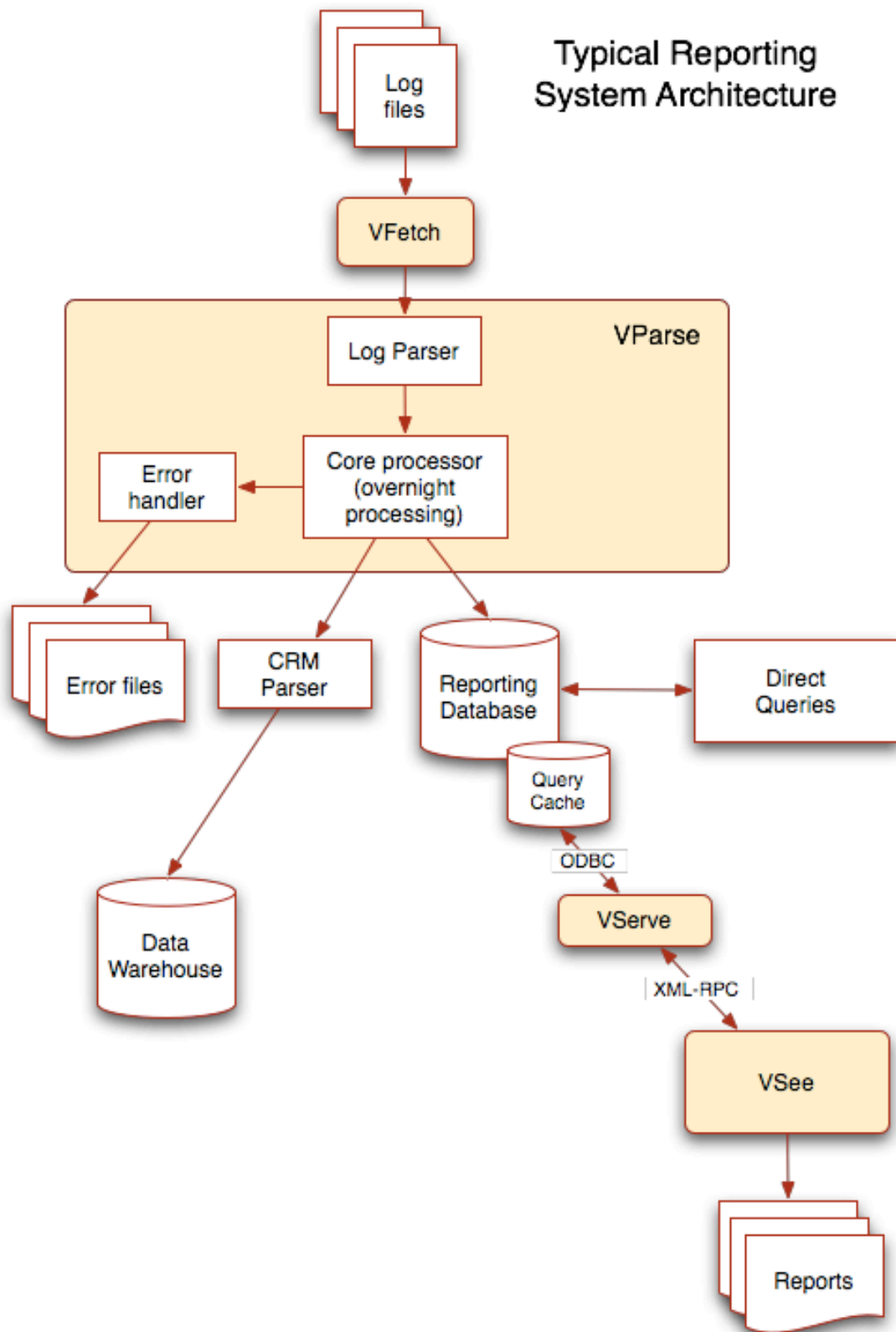
### Custom-built

For these reasons, we believe that the best way to build reporting solutions is to tailor them precisely to the client's reporting requirements. 'General purpose' reporting tools, in our experience, usually fail because they are inefficient, unusable, or inaccurate – or all three.

In simple terms, a typical Vanilla reporting system has four principal components, each of which has a distinct role in the reporting process:

- Vfetch - Log file retrieval and integrity validation
- VParse - Log file parsing and processing into a database
- VServe - Database query engine
- VSee - Customisable GUI for generating reports from the database

## Overall System Architecture



The overall reporting system installation is built using an n-tier architecture for maximum transparency of operation. Most commercial and open-source database

formats (including Oracle) are supported. The core processing software, custom-built from Vanilla's processing toolbox Vtools, is written in C. Log files are retrieved from their respective locations, parsed and pre-processed.

The processor is normally set up to run on a nightly schedule, with any user-defined actions or errors being read in from the configuration database prior to processing.

The processor populates the database with the processed data for that day, logging any errors that occur during processing.

When the user queries the database, he does so via a GUI that interfaces to the database via XML-RPC and ODBC through Vserve, the Vanilla XML-RPC query broker. The use of ODBC means that most commercial and open source database formats are supported. Popular report queries are cached in the database to save on processing time. Ad hoc queries can also be made direct to the database.

The optional CRM parsing module can populate a pre-existing CRM data warehouse with appropriately formulated data from the reporting engine.

### VFetch: Log file retrieval

VFetch is a tool for retrieving log files or other data sources for processing. In theory, this could be achieved by a simple shell script, but VFetch incorporates a number of value-adding features:

- Editable table of log file locations
- Configurable retrieval protocols (e.g. push / pull, FTP/SCP/RCP)
- Log file integrity checking (checks to see if the log file is of the expected size, was created on the expected date, etc.)
- Alerting – sends email or SMS if any problems occur with the retrieval

### VParse: Core Processing of the source data

Vanilla's reporting solutions are constructed from a set of pre-built tools (Vtools) according to a flexible set of design principles. This allows us to rapidly develop custom applications that capture from the source data precisely the information the client needs.

Vparse has three primary processing phases: Pre-processing, Parsing, and Output.

#### Pre-processing

Before the data is parsed, we have to do some pre-processing, which includes:

1. Filtering - discarding data that is not going to be reported upon
2. Concatenation - combining multiple data sources and sorting it
3. Translation - changing or transposing data elements.

## Parsing and processing

Once pre-processing is complete, the data is parsed - unpacked into its constituent elements - and stored in an in-memory database.

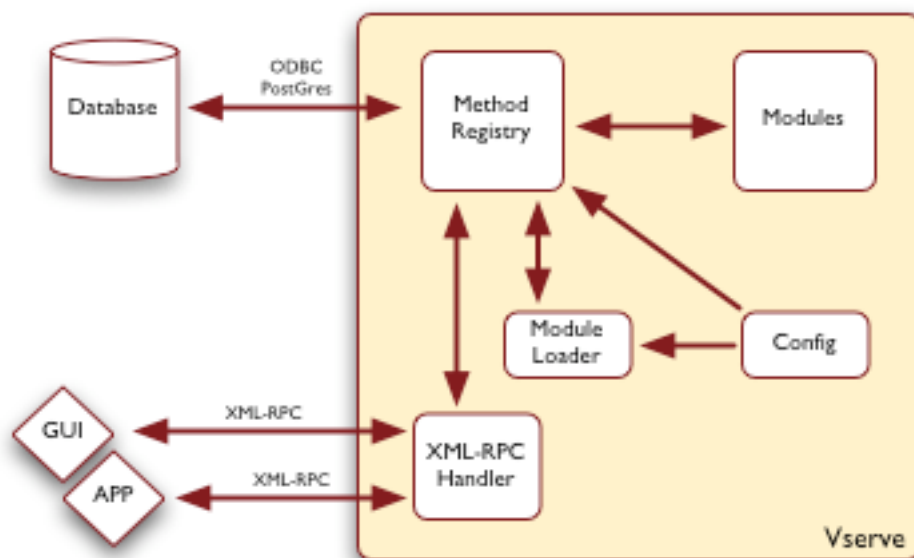
The practical upper limit on the size of this database is about 1 Gb, so at the pre-processing stage the log file is typically unbundled into smaller chunks, each containing only whole sessions.

Using an in-memory database means that the further processing required to generate the reporting databases can run very quickly.

## Output and Write to Reporting Databases

This operation writes the data from the parsing process to the database required by the client for reporting. The schema for this database is designed following consultation with the client about what reports are needed: this allows as much unnecessary data to be thrown away as possible, and keeps the database relatively compact.

## Vserve



VServe is the server interface between the loaded, configured database containing information from processed log files and the GUI or other application that will ultimately produce reports for users.

VServe is a modular system, which abstracts the database implementation away from the presentation of that data in reports or other applications. It allows new reporting datasets to be created quickly and easily without changing the configuration of the core database. It communicates with industry-standard XML-RPC for simplicity, and it performs query-caching, to ensure that common or repeat database queries are handled without delay.

The modules that VServe runs define the complete reporting application. Each module is made up of a number of methods. In the case of a typical reporting module, each of these methods takes inputs from the GUI (in most cases, a combination of user input and GUI-set defaults) and returns a data set, which is then presented back to the user through the GUI.

All of the datasets for the standard reports to be made available through the database are defined in VServe. As and when additional report datasets are required, they will be defined here.

## VSee

VSee is the reporting system's GUI. It handles the following tasks.

### Security and administration

Users log in with a username and password. New users can be added, and their permissions changed, by the administrator. Permissions can be used to restrict access to report types: so that, for example, a third party partner is only able to see reports on only their own data.

### Report presentation

The user can set up the parameters for a particular report and run that report off in HTML, Pdf or Excel format.

### Custom reporting

Vsee can be set up to produce reports against any methods that take as input a set of query parameters (entered through a web-based form, designed in Vsee using XML) to produce a dataset output (formatted as a graphical report in html or pdf).

### Corporate look and feel

The look and feel of the reports is set up through a style sheet system, which means that every report produced through the system will have a consistent look and feel, conforming to the company's brand identity guidelines.