

# Team 51

## Final Presentation

Owen Kahle  
Lucas Marshall  
Paul Mawhirter  
Ben Ridgway

# Technical Review

# Customer motivation and vision

The current situation motivating the customer:

Our customer wishes to centralize data from multiple heterogeneous databases into one intuitive graphical interface.

# Customer motivation and vision

The solution we needed to develop for the customer:

A simple way for analysts to quickly view large amounts of data from a database, without intimate knowledge of database structure.

# Customer motivation and vision

The product / solution will fit into the customer's environment:

We developed an easy to use desktop application which is independent of any particular database type, connection, and/or schema.

# Example Use Case

- Customer has large database and wishes to have his many Data Analysts go through it looking for specific details, yet he only has one employee the XML Programmer who has any knowledge of the database structure and relations.

# Example Use Case cont'd

## Use GIMpeD

- He has his XML Programmer (the employee who knows the DB) write up the XML configuration file.
- The numerous Data Analysts can now load up GIMpeD and the XML configuration file. They now have access to the DB, and it is laid out in an intuitive iTunes like format.

# Example Use Case summary

- The numerous Data Analysts in the customers employment are now searching through the DB with the apparent skill and proficiency as the more senior XML Programmer through the use of GIMpeD.

# Requirements (revised)

- Ability to connect to multiple database backends. Data source should not impact interface
- Must run in Linux and Windows
- All relations and data sources are defined in a single XML document
- The GUI must respond in real-time. Any bottlenecks can only be caused by the database

# Requirements (continued)

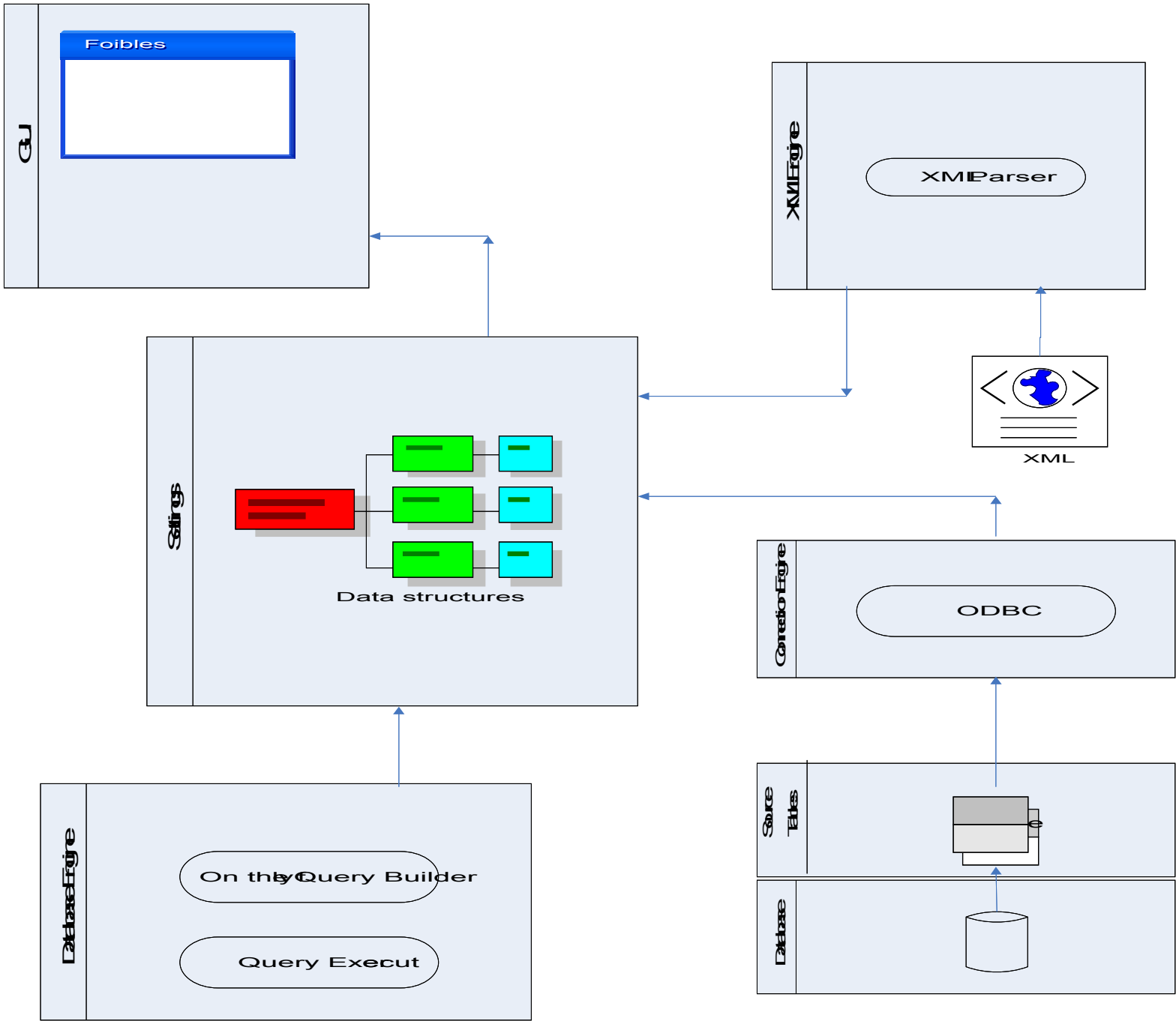
- The GUI must be able to be used by somebody who doesn't understand the database structure.
- The XML specification must be simple to create for someone with some database experience
- System must respond with meaningful error messages
- The program must support large data sets

# Technical Background

- The XML configuration file will allow the XML programmer who has an in-depth understanding of the database to configure GIMpeD such that a novice can easily sort through the data without regard to the underlying structure or relations in the database.

# Technical Background cont'd

- ODBC connection strings. This is used to specify the connection to the database in the XML configuration file. Refer to [www.connectionstrings.com](http://www.connectionstrings.com) for specifics.



# High Level Design Approach cont'd

- Graphical User Interface: This section is responsible for taking input from the user via point and click on various items and forwarding these selections to the Database Engine for query building. The GUI is also responsible for formatting, and displaying the data in accordance with input from Settings.

# High Level Design Approach cont'd

2. XML Engine: This process parses the XML configuration file into useful data for the GIMpeD application. It also verifies that the XML configuration file follows proper format as specified in the XML document type definition (dtd). The dtd allows a strong enforcement of file format and syntax on the XML configuration file with very little effort.

# High Level Design Approach cont'd

3. Settings: is the data structure heart of GIMpeD. All data passes into and through Settings before it goes anywhere else.

# High Level Design Approach cont'd

4. Database Engine: contains the functionality to create queries embodied in the on the fly Query Builder, and to execute the queries on the database. Query creation is based on the XML configuration file specifications, and the selections made from the Option lists. Both sets of data as specified earlier are acquired through the Settings.

# High Level Design Approach cont'd

5. Connection Engine: establishes a connection to the database through use of the Open Database Connectivity (ODBC) method. This makes it possible to access any data from any source regardless of which DBMS is in use. The only two requirements for making a new DBMS work are that it supports standard SQL statements and an appropriate ODBC driver is installed.

# High Level Design Approach cont'd

6. The database is the data source to be viewed. The required information for its structure is specified in the XML configuration file. As of now the database can be local or network based. Again, connectivity is managed through ODBC drivers. Assuming proper ODBC operation, connectivity to GIMpeD is transparent.

How did we do?

# Missing Features

- Concurrent multi-database support
- Multiple data type support (IP addresses)
- Some GUI refinement
- Tabs (multiple simultaneous XML specs)
- Multi-threading

# Well Implemented Features

- XML Parser
- Gimped.Settings namespace/data struct
- GUI layout and usability
- Modular design

# Issues

- Mono is not production ready
- Linux ODBC can be hard to set up
- Hex viewer not always accurate
- Graphical issues in Mono

# Suggestions for Product

## Biggest Limitations:

- Constructing relationships over separate database types
- Handling and filtering large amounts of data (ie petabyte-size databases)

## Further Development:

- Scalable, multi-node server database engine w/ cross server relationship support.

# Team Performance

# Team Collaboration and Values

- **Communication** - Assure that all participants are involved and well informed.
  - We did this well during our regular Tue/Thu meetings
  - We did not do as well when it came time to inform others about what we did individually

# Team Collaboration and Values

- **Dedication** - Meet regularly and complete tasks on schedule.
  - We meet regularly...and then some
  - While most tasks were completed on schedule, not all were and that affected what we were able to accomplish in the given time frame

# Team Collaboration and Values

- **Reliability** - We will follow through with our commitments with integrity.
  - We gave all commitments an honest effort, however due to unforeseen challenges and time constraints we were not able to deliver all of our commitments.

# Team Collaboration and Values

- **Organization** - Follow close to defined processes and documentation.
  - We followed our programming guidelines rather strictly
  - We constantly redefined our objectives/goals for each week and completed tasks based on priority

# Team Collaboration and Values

- **Quality** - Develop a product we can all be proud of.
  - In order to deliver a usable product we decided against a hurried rush to add more and more features without being sure as to the reliability of our product.

# Team Collaboration and Values

- **Cooperation** - Work as a group to overcome all obstacles.
  - We did this best in our meetings
  - We learned how use team members to solve problems we could not solve alone

# Project Risk

## (Programming Learning Curve)

Risk: Some of the members on our team are not fluent in the languages and technologies that we will have to use for this project.

Solution: We obtained a library of "For Dummies" books to rapidly teach group members who are not skilled in C#.

# Project Risk (Stress)

Risk: Our ability to remain calm as a group greatly benefits our ability to get work done.

Solution: We will also be mindful of deadlines and not procrastinate. We will listen to classical music during all meetings and coding sessions.

# Project Risk (Personal Illness)

Risk: Personal Illness, injury or family emergency is another risk we need to consider.

Solution: We are constantly in contact with each other so we can cover for another team member incase he contracts the plague.

# Project Risk

(Long distance communication)

Risk: 2,653 miles of separation between our sponsors and us.

Solution: Frequent email communication with sponsors.

# Schedule Performance

- Release 1
  - We completed most deliverables
  - Had to redo the schedule for all release dates, due to design flaw pointed out by our sponsors
- Release 2
  - Completed all deliverables minus on-the-fly query builder

# Schedule Performance

- Beta Release

- Our release was not really tested as we did not have a working release until the night before
- It took another 2 weeks to get everything working as desired or with reasonable limits due to time constraints

# Schedule Performance

- Final Release
  - Our final release has the core functionality we planned for
  - Limits as to all the features we had hoped to implement

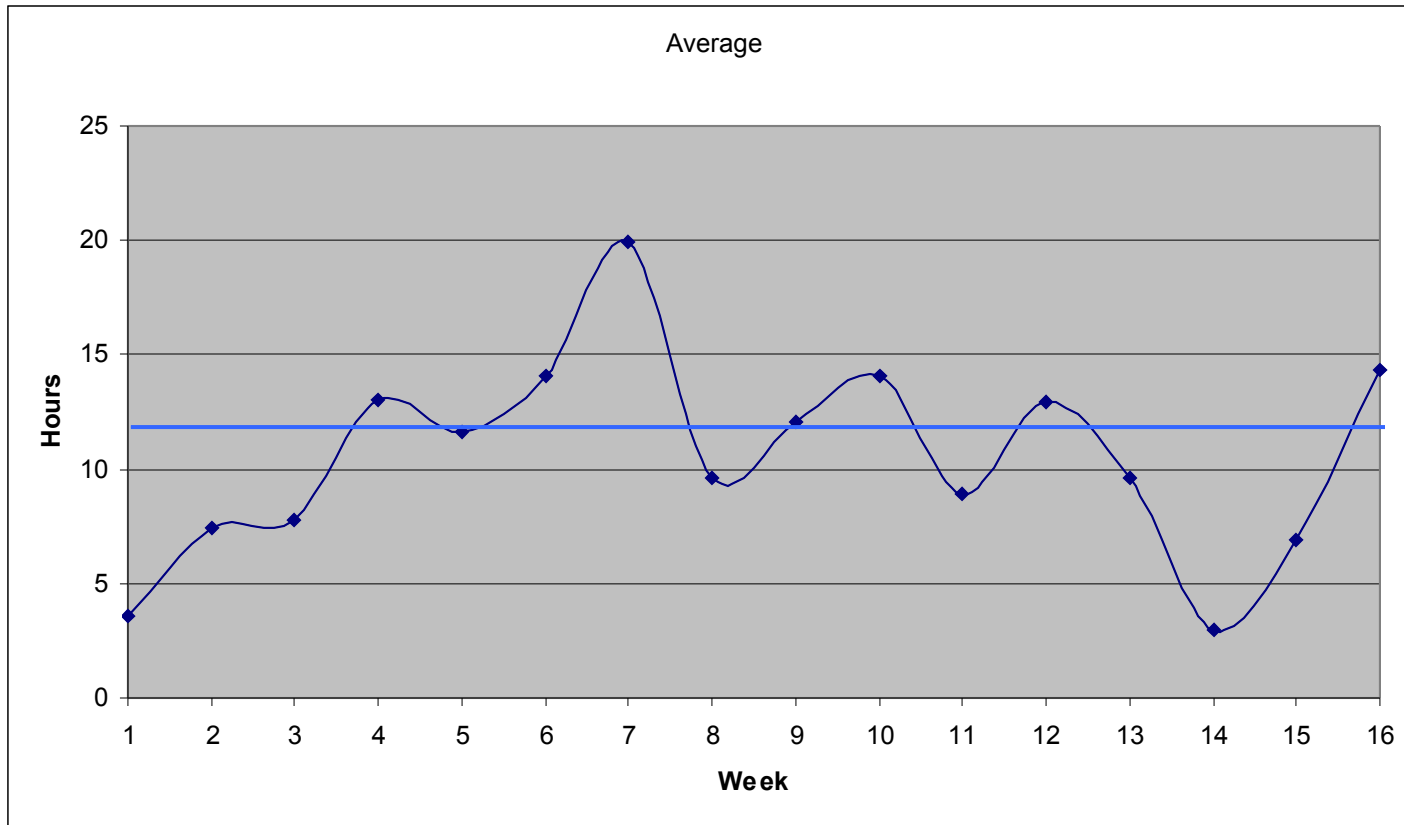
# Project Direction

- Yes the Project did change direction.
  - Petabyte scale DB. Scope and therefore feasibility of project affected by revelation that the datasources will be in the petabyte range.
  - Cross Platform Compatibility. Mono is not as fully developed as advertised. Many features that .NET implements, Mono does not. Therefore Linux is not well supported.
  - Multiple Databases. Making joins and constrains across multiple large databases was determined to be infeasible to develop with limited time we had for development.

# Project Direction cont'd

- Lessons learned.
  - Need clear communication with sponsor from day 1.
  - Long distance communication for requirements elicitation is extremely difficult.
  - Sponsors are very understanding and patient.
  - When possible, having a flexible design that does not lock you into a specific paradigm is invaluable. Allows for adaptability.

# Average Hours vs Planned



# Average Hours

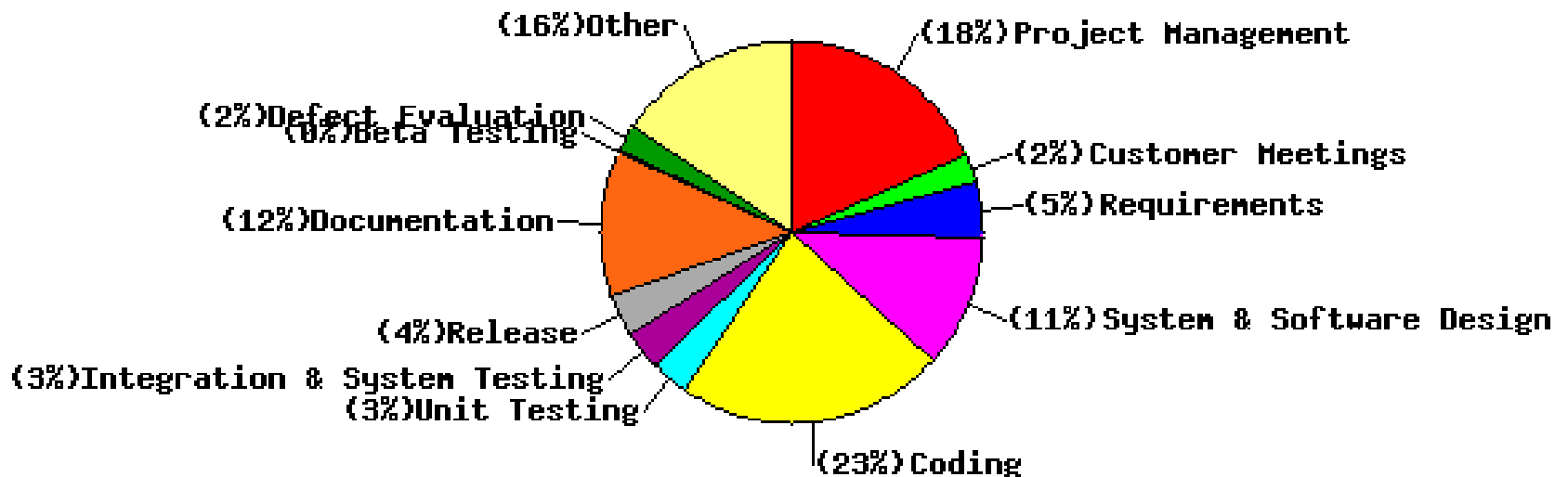
Final average 10.5 hours per week

– 11 hours average omitting thanksgiving break

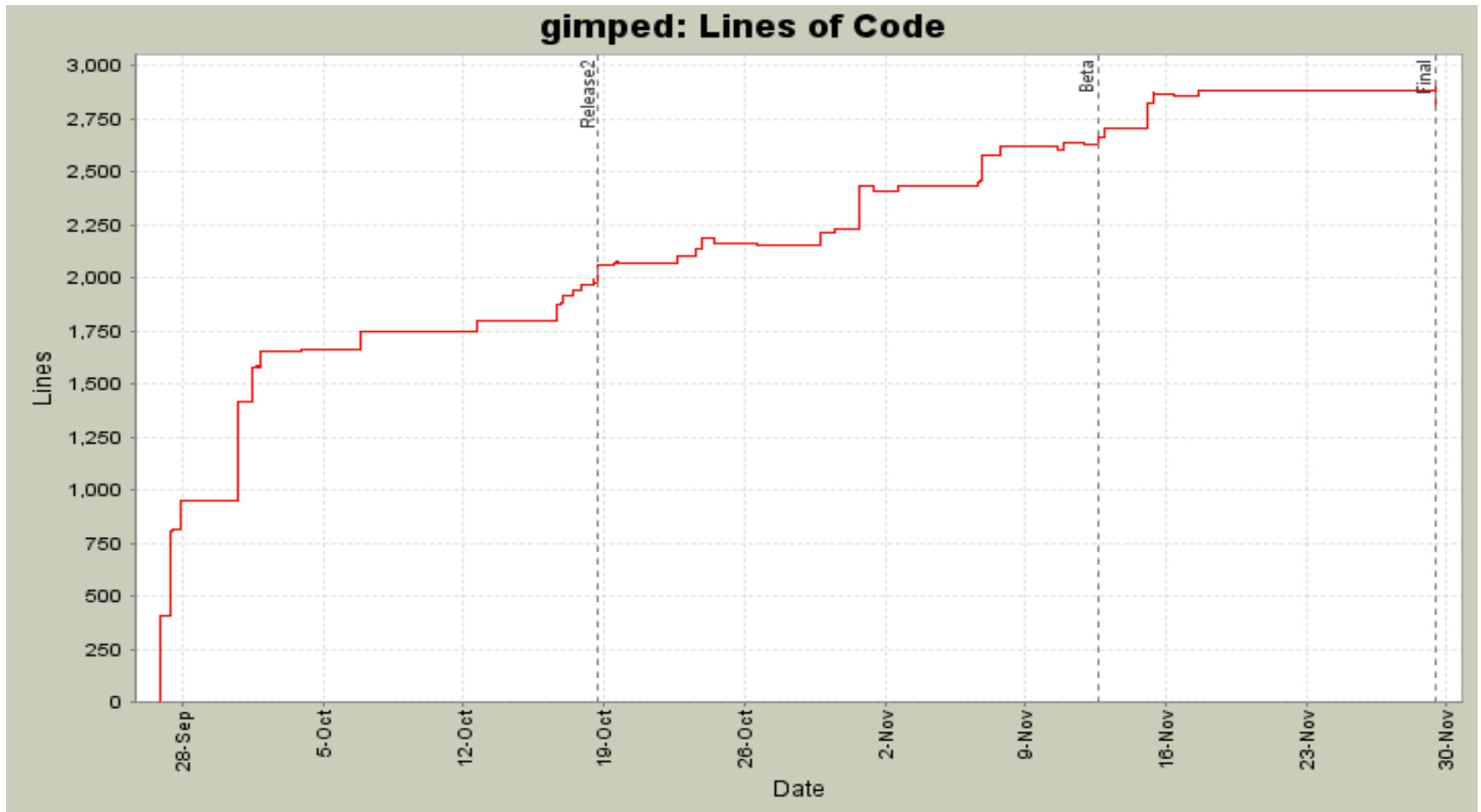
Final Total 658 hours for entire project

# Effort Distribution

Team 51 - GUI-DB Activity Data



# Product Size



# Testing -Description

- We tested using a regression suite that included all xml test files ever created, and an ever growing source of test databases.
- Bugs were logged in the PRS system, and all have been closed.
- Some feedback from sponsors

# Testing Details

- Cross Platform Testing:
  - We performed all tests on Windows.
  - We performed all tests on Gentoo
    - Noted that Mono is not fully developed.
- Correct XML configuration testing
  - Verified that GIMpeD would work given correct XML configuration file

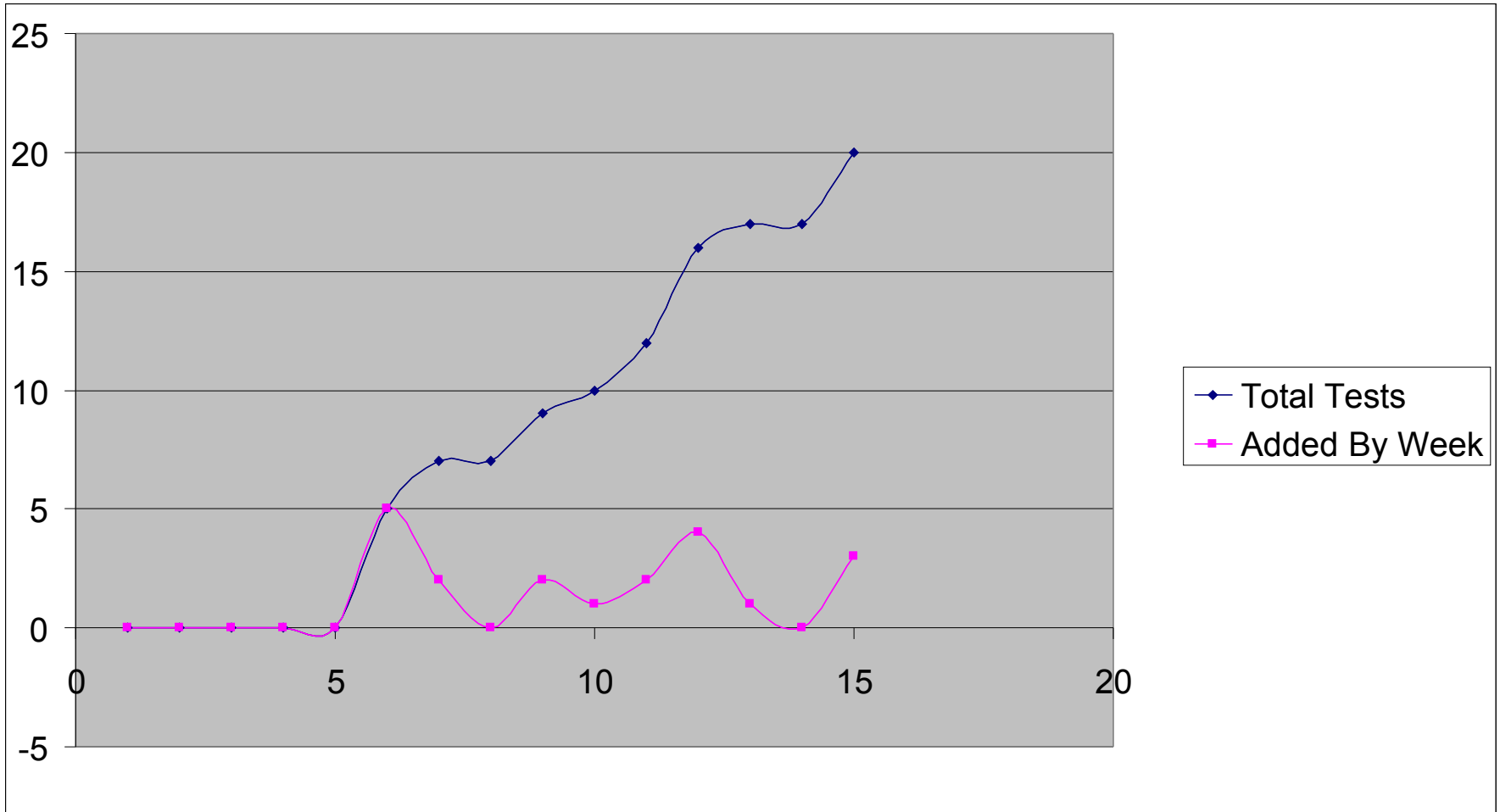
# Testing Details cont'd

- Incorrect XML configuration file testing.
  - Tested for error handling and response of GIMpeD when variations of the known good XML config file were used with specific faults.
- GUI Response to XML configuration file testing
  - Tested different XML configuration files, and verified proper response from the GUI including display, joins and constrains.

# Testing Details cont'd

- Database Connectivity Testing
  - Tested GIMpeD's ability to connect to different Databases.
    - varying authentication methods
    - empty databases
    - database whose structure does not match the XML configuration file.

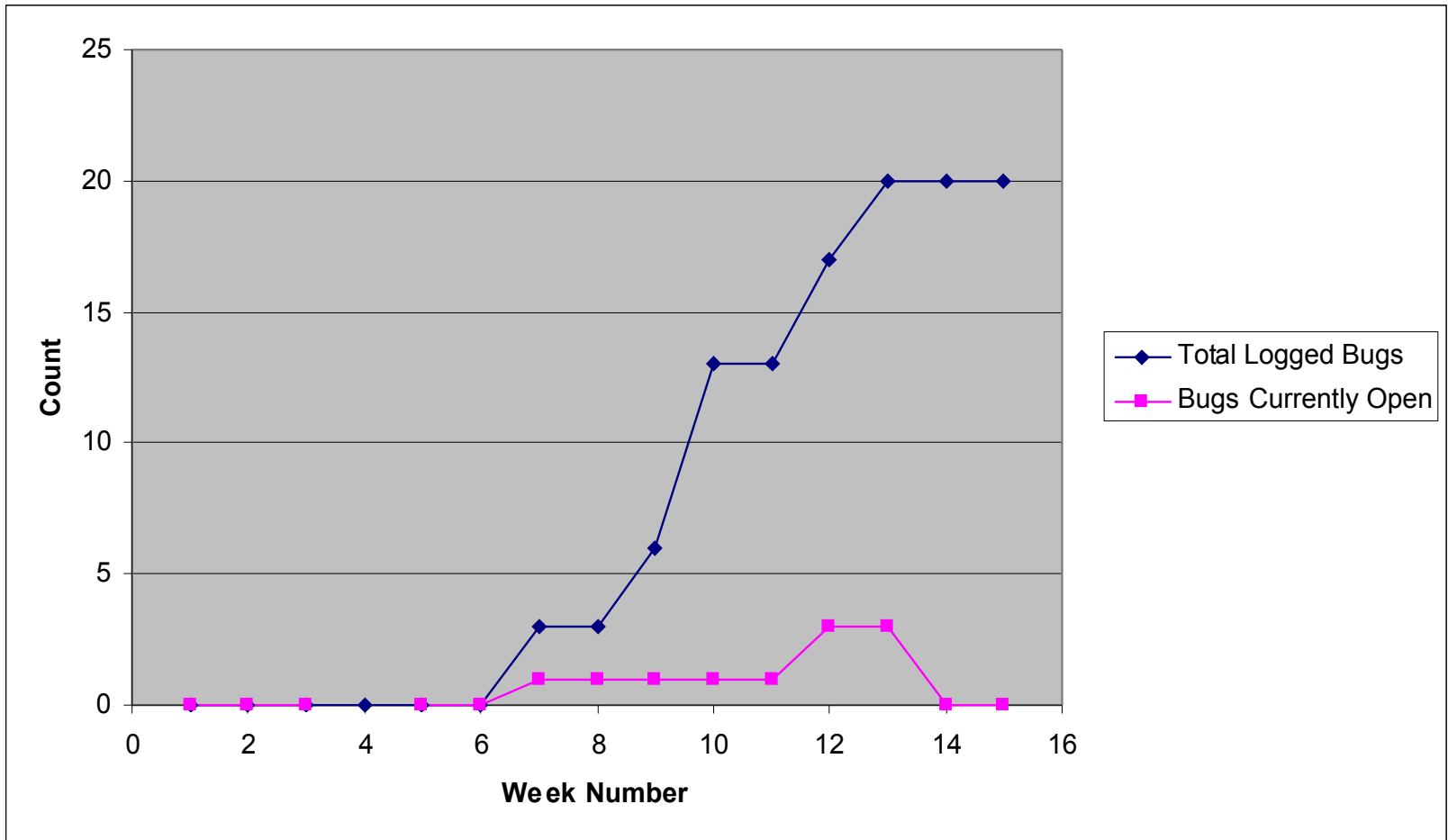
# Testing – plot of test cases per week.



# Testing, Bug summary

- Our first bug was discovered in the 7<sup>th</sup> week.
- Our last bug was discovered in the 13<sup>th</sup> week
- Our last bug was closed in the 14<sup>th</sup> week.
- Discovered a total of 20 bugs.

# Testing – plot of total bugs and bug open bugs by week.

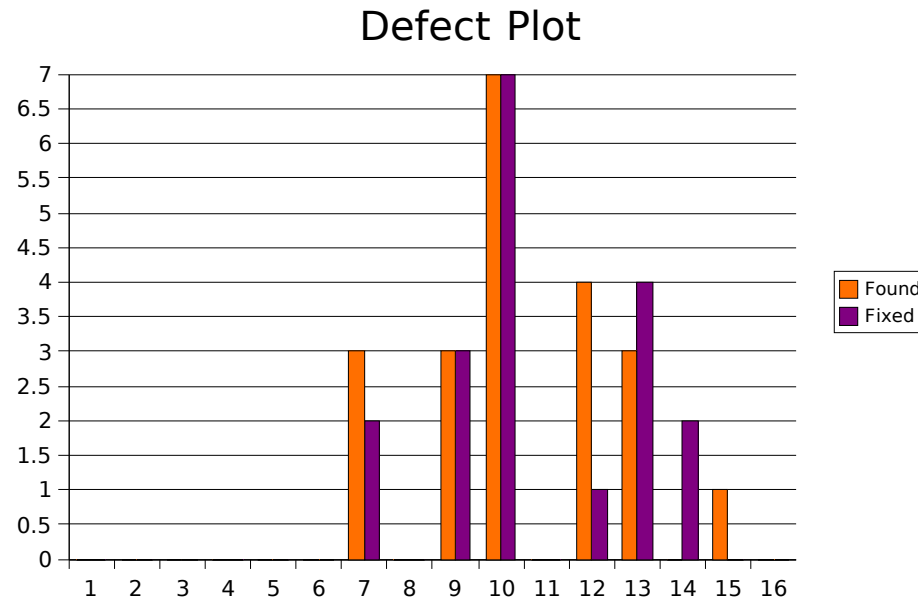


# Testing Conclusions

- We are confident in the current release as being free of bugs.
  - As seen on the previous graph we had three weeks in the end of no new bugs being introduced.
  - We also had the last two weeks where all bugs were closed.
- There are still a few issues, but those are issues, and not bugs.
  - Refer to the Issues list for details.
    - Petabyte Range.
    - Multiple Databases.
    - IP data type display.

# Defects Found per Week

Week	Found	Fixed
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	3	2
8	0	0
9	3	3
10	7	7
11	0	0
12	4	1
13	3	4
14	0	2
15	1	0
16	0	0
total	21	19



# Project Retrospective

- In order to achieve a functional Product in Linux we would not have used mono as it is not quite ready for prime time yet
- We would have had more communication with our sponsors at the very beginning
- Would have laid out clearer objective/goal assignments to individual team members with more specific deadlines

# Significant Things Learned

- Team Leadership
- Client Interfacing
- Project Management
- Time Management
- Public Speaking/Presentations
- Software Repository/Version-Control
- Required Documentation