Xtremesoft

# AppMetrics Solutions for QA and Test Professionals

Improving performance by giving QA insight into the Applications real operational characteristics

www.xtremesoft.com

# Why do users come to Xtremesoft?

- “The Application is failing/slowing and needs urgent help” – 30% of the trials of the product.
- “The critical applications need constant performance management, so we can constantly monitor and improve them” – 30% of the trials.
- “We need to be alerted, before the user calls, that the Application’s performance (not just the machines and OS) is slowing or it has stopped responding” – 20% of the trials.
- **QA/Capacity planning/testing management** – “We have test suites, but we really don’t know how the application really functions internally and if our test scenarios are even similar to the real work load applied to the application. ” – 20% of the trials, but the fastest growing.

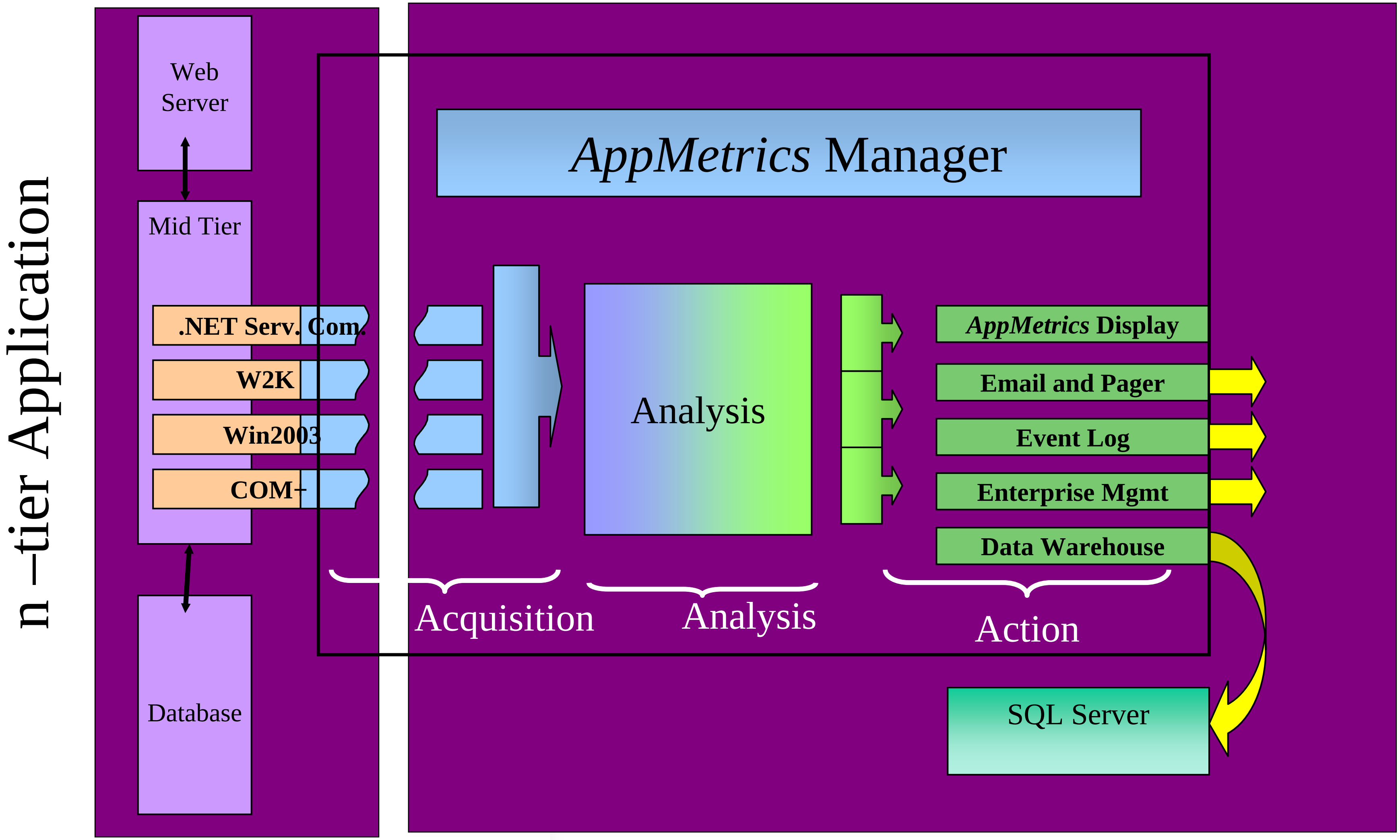
# Why is use of  AppMetrics in QA/testing management rising so quickly ?

- QA understand that Com+ Applications carry the bulk of the day-to-day operational load, for most large Microsoft centric shops. All the ‘buzz’ may be around .net, but real world is that very few serious transactional .net applications are running, today, and questions on performance remain.
- QA/testing has to understand the application – what calls what - before they can build representative tests. The original developers of the Application are often no longer available to ask!
- QA have no way to understand the real traffic on the application – what are the users really doing each day and at what times, and how does the application, react internally.
- Development want to make progressive modifications and use newer technologies, but no one (including QA) can really tell what impact a change has had, or (better) will have, on various parts of the application. We cannot wait to find out when it goes into production!

# AppMetrics Architecture

- AppMetrics is not application invasive – no hooks are required into the application. QA has to control load on the servers, but are otherwise independent of Development and operations in the use of AppMetrics.
- The AppMetrics Agent runs on the AppServer where it collects and forwards application events to the AppMetrics Manager
- The AppMetrics Manager, usually runs on its own machine, in the QA department, and correlates the events, and generates unique application metrics and adds them to a database for reporting
- The AppMetrics Manager monitors the Application metrics in real-time, compares them to benchmarks and can alert based on predefined alert criteria

# Architecture



# Typical problems AppMetrics can help resolve

- What is going on within the application and why?
- Which component and method is slowing down the business process?
- Where should I invest development/engineering resources?
- Which component is hung? More importantly which components are causing me potential issues?
- Where did performance change?
- How can I gain visibility into the application, to understand its real usage and loads?

# What do we *not* provide

- All the answers!

We bring you data on two levels – reporting and alerting.

- An understanding of your application –

We look at it generically, from a “what is doing what to what” perspective! You have to apply the Meta Knowledge to the data we provide, to get information and answer the question, “why is it doing it?”

# AppMetrics Uses for QA & Test

- **Reducing the time spent at ‘finger-pointing’ meetings**

QA spends too long in Development ‘Finger-pointing’ Meetings, where no one has any data on why the real application problems are? If only we had some concrete data to work with.

- **What is really happening?**

The original application architect has long gone and no one can tell QA the sequence of events that the application executes. So we are flying blind when we try and build our test scripts.

- **Capturing real world usage**

QA has no reliable way of capturing the real world application usage. QA test data may or may not represent what the users do each day.

- **Impact of changes**

Changes are made to the application, but without being able to understand the underlying impact. QA really can’t tell if they are good or bad! Measuring performance from modification to modification in a standard way is vital?

- **Sharing information with Development and Operations**

QA needs to be able to share its data with Dev and Ops. Typically we use tools that other groups do not regularly use. A tool that serves all groups would be invaluable.

# Finger–pointing Meetings?

Problem:

A multi-tier application slows down. The Team Leader calls a meeting.

“What’s causing the slowdown?”

Each attendee points to the person on the right!

- How do you stop the finger-pointing?
- How do you eliminate these meetings?
- How do you identify the root cause?

# Breaking the Cycle

- The Web team blames the middle-tier team
- The mid-tier team blames the DBA’s
- The DBA’s blame the network
- The Network team blames the web server and so it goes around … while
- QA sits on the sidelines with no reliable way to resolve the who and what of the issue!

# Eliminate Unnecessary Participants

- By identifying where performance has changed, you can focus on the likely root causes.
- This is true in all software phases:
  - Development, testing before Check-in
  - Quality Assurance, verifying before Release
  - Operations, resolving production problems

# Finger-pointing Meetings?

What our Customers are saying:

*"You guys are awesome! In less than 5 minutes, these drilldown reports traced the exact component and method that were giving me trouble - we saved weeks."*

# What is really happening?

Problem: **Transactions are running slowly.**

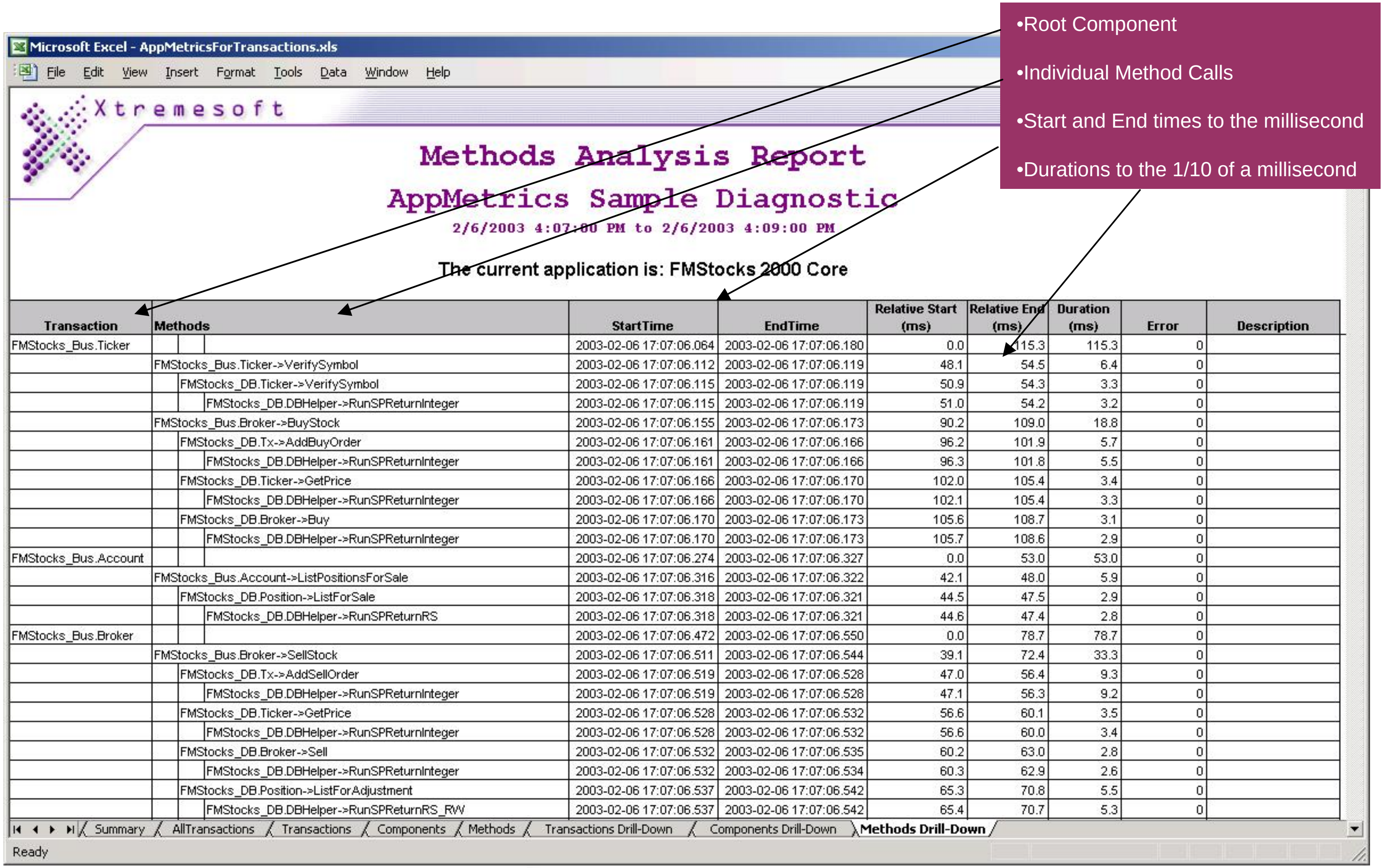
How can I tell which method(s) in each transaction is causing the problem?

- Transactions involve multiple method calls. You may remember the list of method calls involved in a transaction, but which method calls which?
- The relationships between method calls can be easily forgotten. How can you tell which method in the call chain is causing the problem? (“Fred has left”)

# What is really happening?

## Solution: AppMetrics’ Method Analysis Report

- Root Component
- Individual Method Calls
- Start and End times to the millisecond
- Durations to the 1/10 of a millisecond



Microsoft Excel - AppMetricsForTransactions.xls
File Edit View Insert Format Tools Data Window Help
Xtremesoft

Methods Analysis Report

AppMetrics Sample Diagnostic

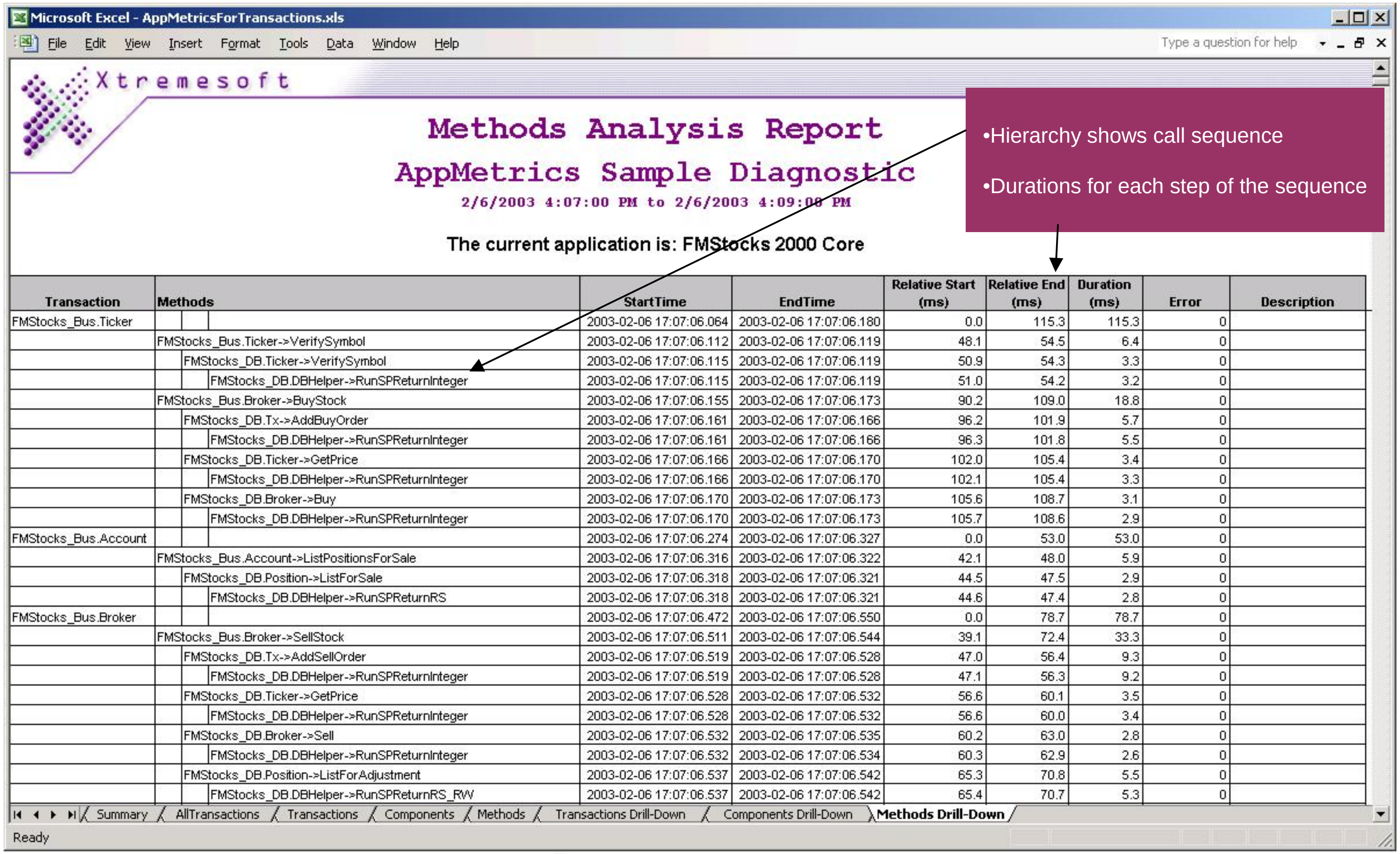
2/6/2003 4:07:00 PM to 2/6/2003 4:09:00 PM

The current application is: FMStocks 2000 Core

| Transaction | Methods | StartTime | EndTime | Relative Start (ms) | Relative End (ms) | Duration (ms) | Error | Description |
|---|---|---|---|---|---|---|---|---|
| FMStocks_Bus.Ticker | | 2003-02-06 17:07:06.064 | 2003-02-06 17:07:06.180 | 0.0 | 115.3 | 115.3 | 0 | |
| | FMStocks_Bus.Ticker->VerifySymbol | 2003-02-06 17:07:06.112 | 2003-02-06 17:07:06.119 | 48.1 | 54.5 | 6.4 | 0 | |
| | FMStocks_DB.Ticker->VerifySymbol | 2003-02-06 17:07:06.115 | 2003-02-06 17:07:06.119 | 50.9 | 54.3 | 3.3 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.115 | 2003-02-06 17:07:06.119 | 51.0 | 54.2 | 3.2 | 0 | |
| | FMStocks_Bus.Broker->BuyStock | 2003-02-06 17:07:06.155 | 2003-02-06 17:07:06.173 | 90.2 | 109.0 | 18.8 | 0 | |
| | FMStocks_DB.Tx->AddBuyOrder | 2003-02-06 17:07:06.161 | 2003-02-06 17:07:06.166 | 96.2 | 101.9 | 5.7 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.161 | 2003-02-06 17:07:06.166 | 96.3 | 101.8 | 5.5 | 0 | |
| | FMStocks_DB.Ticker->GetPrice | 2003-02-06 17:07:06.166 | 2003-02-06 17:07:06.170 | 102.0 | 105.4 | 3.4 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.166 | 2003-02-06 17:07:06.170 | 102.1 | 105.4 | 3.3 | 0 | |
| | FMStocks_DB.Broker->Buy | 2003-02-06 17:07:06.170 | 2003-02-06 17:07:06.173 | 105.6 | 108.7 | 3.1 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.170 | 2003-02-06 17:07:06.173 | 105.7 | 108.6 | 2.9 | 0 | |
| FMStocks_Bus.Account | | 2003-02-06 17:07:06.274 | 2003-02-06 17:07:06.327 | 0.0 | 53.0 | 53.0 | 0 | |
| | FMStocks_Bus.Account->ListPositionsForSale | 2003-02-06 17:07:06.316 | 2003-02-06 17:07:06.322 | 42.1 | 48.0 | 5.9 | 0 | |
| | FMStocks_DB.Position->ListForSale | 2003-02-06 17:07:06.318 | 2003-02-06 17:07:06.321 | 44.5 | 47.5 | 2.9 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnRS | 2003-02-06 17:07:06.318 | 2003-02-06 17:07:06.321 | 44.6 | 47.4 | 2.8 | 0 | |
| FMStocks_Bus.Broker | | 2003-02-06 17:07:06.472 | 2003-02-06 17:07:06.550 | 0.0 | 78.7 | 78.7 | 0 | |
| | FMStocks_Bus.Broker->SellStock | 2003-02-06 17:07:06.511 | 2003-02-06 17:07:06.544 | 39.1 | 72.4 | 33.3 | 0 | |
| | FMStocks_DB.Tx->AddSellOrder | 2003-02-06 17:07:06.519 | 2003-02-06 17:07:06.528 | 47.0 | 56.4 | 9.3 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.519 | 2003-02-06 17:07:06.528 | 47.1 | 56.3 | 9.2 | 0 | |
| | FMStocks_DB.Ticker->GetPrice | 2003-02-06 17:07:06.528 | 2003-02-06 17:07:06.532 | 56.6 | 60.1 | 3.5 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.528 | 2003-02-06 17:07:06.532 | 56.6 | 60.0 | 3.4 | 0 | |
| | FMStocks_DB.Broker->Sell | 2003-02-06 17:07:06.532 | 2003-02-06 17:07:06.535 | 60.2 | 63.0 | 2.8 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.532 | 2003-02-06 17:07:06.534 | 60.3 | 62.9 | 2.6 | 0 | |
| | FMStocks_DB.Position->ListForAdjustment | 2003-02-06 17:07:06.537 | 2003-02-06 17:07:06.542 | 65.3 | 70.8 | 5.5 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnRS_RW | 2003-02-06 17:07:06.537 | 2003-02-06 17:07:06.542 | 65.4 | 70.7 | 5.3 | 0 | |

# What is really happening?

**Solution: AppMetrics’ Method Analysis Report**


Microsoft Excel - AppMetricsForTransactions.xls
File Edit View Insert Format Tools Data Window Help
Type a question for help
Summary AllTransactions Transactions Components Methods Transactions Drill-Down Components Drill-Down Methods Drill-Down
Ready

Xtremesoft

Methods Analysis Report

AppMetrics Sample Diagnostic

2/6/2003 4:07:00 PM to 2/6/2003 4:09:00 PM

The current application is: FMStocks 2000 Core

| Transaction | Methods | | | StartTime | EndTime | Relative Start (ms) | Relative End (ms) | Duration (ms) | Error | Description |
|---|---|---|---|---|---|---|---|---|---|---|
| FMStocks_Bus.Ticker | | | | 2003-02-06 17:07:06.064 | 2003-02-06 17:07:06.180 | 0.0 | 115.3 | 115.3 | 0 | |
| | FMStocks_Bus.Ticker->VerifySymbol | | | 2003-02-06 17:07:06.112 | 2003-02-06 17:07:06.119 | 48.1 | 54.5 | 6.4 | 0 | |
| | | FMStocks_DB.Ticker->VerifySymbol | | 2003-02-06 17:07:06.115 | 2003-02-06 17:07:06.119 | 50.9 | 54.3 | 3.3 | 0 | |
| | | | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.115 | 2003-02-06 17:07:06.119 | 51.0 | 54.2 | 3.2 | 0 | |
| | FMStocks_Bus.Broker->BuyStock | | | 2003-02-06 17:07:06.155 | 2003-02-06 17:07:06.173 | 90.2 | 109.0 | 18.8 | 0 | |
| | | FMStocks_DB.Tx->AddBuyOrder | | 2003-02-06 17:07:06.161 | 2003-02-06 17:07:06.166 | 96.2 | 101.9 | 5.7 | 0 | |
| | | | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.161 | 2003-02-06 17:07:06.166 | 96.3 | 101.8 | 5.5 | 0 | |
| | | FMStocks_DB.Ticker->GetPrice | | 2003-02-06 17:07:06.166 | 2003-02-06 17:07:06.170 | 102.0 | 105.4 | 3.4 | 0 | |
| | | | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.166 | 2003-02-06 17:07:06.170 | 102.1 | 105.4 | 3.3 | 0 | |
| | | FMStocks_DB.Broker->Buy | | 2003-02-06 17:07:06.170 | 2003-02-06 17:07:06.173 | 105.6 | 108.7 | 3.1 | 0 | |
| | | | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.170 | 2003-02-06 17:07:06.173 | 105.7 | 108.6 | 2.9 | 0 | |
| FMStocks_Bus.Account | | | | 2003-02-06 17:07:06.274 | 2003-02-06 17:07:06.327 | 0.0 | 53.0 | 53.0 | 0 | |
| | FMStocks_Bus.Account->ListPositionsForSale | | | 2003-02-06 17:07:06.316 | 2003-02-06 17:07:06.322 | 42.1 | 48.0 | 5.9 | 0 | |
| | | FMStocks_DB.Position->ListForSale | | 2003-02-06 17:07:06.318 | 2003-02-06 17:07:06.321 | 44.5 | 47.5 | 2.9 | 0 | |
| | | | FMStocks_DB.DBHelper->RunSPReturnRS | 2003-02-06 17:07:06.318 | 2003-02-06 17:07:06.321 | 44.6 | 47.4 | 2.8 | 0 | |
| FMStocks_Bus.Broker | | | | 2003-02-06 17:07:06.472 | 2003-02-06 17:07:06.550 | 0.0 | 78.7 | 78.7 | 0 | |
| | FMStocks_Bus.Broker->SellStock | | | 2003-02-06 17:07:06.511 | 2003-02-06 17:07:06.544 | 39.1 | 72.4 | 33.3 | 0 | |
| | | FMStocks_DB.Tx->AddSellOrder | | 2003-02-06 17:07:06.519 | 2003-02-06 17:07:06.528 | 47.0 | 56.4 | 9.3 | 0 | |
| | | | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.519 | 2003-02-06 17:07:06.528 | 47.1 | 56.3 | 9.2 | 0 | |
| | | FMStocks_DB.Ticker->GetPrice | | 2003-02-06 17:07:06.528 | 2003-02-06 17:07:06.532 | 56.6 | 60.1 | 3.5 | 0 | |
| | | | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.528 | 2003-02-06 17:07:06.532 | 56.6 | 60.0 | 3.4 | 0 | |
| | | FMStocks_DB.Broker->Sell | | 2003-02-06 17:07:06.532 | 2003-02-06 17:07:06.535 | 60.2 | 63.0 | 2.8 | 0 | |
| | | | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.532 | 2003-02-06 17:07:06.534 | 60.3 | 62.9 | 2.6 | 0 | |
| | | FMStocks_DB.Position->ListForAdjustment | | 2003-02-06 17:07:06.537 | 2003-02-06 17:07:06.542 | 65.3 | 70.8 | 5.5 | 0 | |
| | | | FMStocks_DB.DBHelper->RunSPReturnRS_RW | 2003-02-06 17:07:06.537 | 2003-02-06 17:07:06.542 | 65.4 | 70.7 | 5.3 | 0 | |

- Hierarchy shows call sequence
- Durations for each step of the sequence

# What is really happening?

Our Customers say:

*“Xtremesoft is the only company that translates application logic metrics into business performance information."*

# Capturing real world usage?

Problem: **I need to know what the real usage looks like**

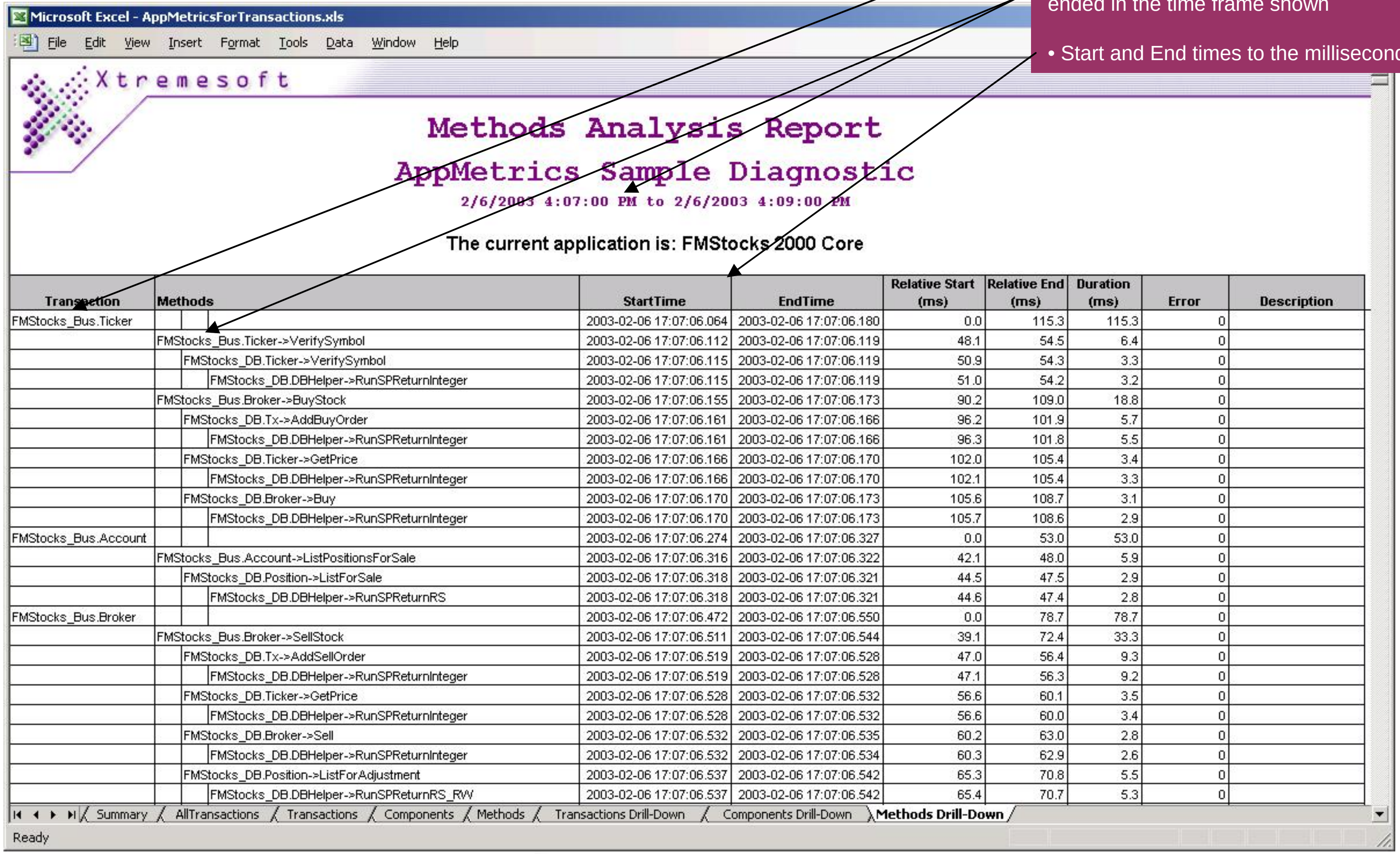
How can I tell which components call what method(s) in each transaction?

- Transactions involve multiple method calls. You may remember the list of method calls involved in a transaction, but which method calls which?
- The relationships between method calls can be easily forgotten. How can you tell which method in the call chain is causing the problem?

# Capturing real world usage?

**Solution: AppMetrics’ Method Analysis Report shows all transactions during the period requested**

- Root Component and calling agent
- Reports all  transactions begun and ended in the time frame shown
- Start and End times to the millisecond



Xtremesoft

## Methods Analysis Report

## AppMetrics Sample Diagnostic

2/6/2003 4:07:00 PM to 2/6/2003 4:09:00 PM

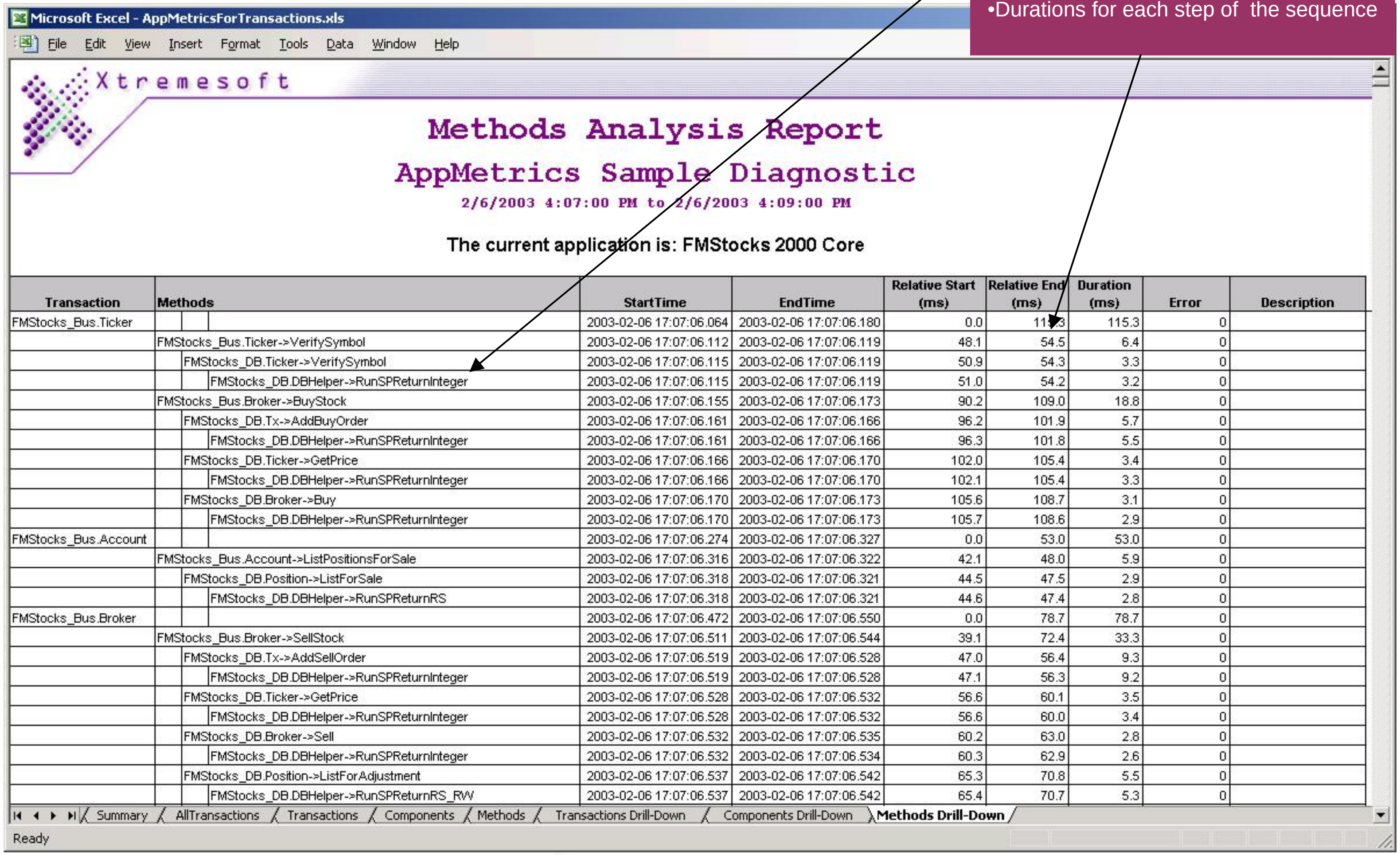
The current application is: FMStocks 2000 Core

| Transaction | Methods | StartTime | EndTime | Relative Start (ms) | Relative End (ms) | Duration (ms) | Error | Description |
|---|---|---|---|---|---|---|---|---|
| FMStocks_Bus.Ticker | | 2003-02-06 17:07:06.064 | 2003-02-06 17:07:06.180 | 0.0 | 115.3 | 115.3 | 0 | |
| | FMStocks_Bus.Ticker->VerifySymbol | 2003-02-06 17:07:06.112 | 2003-02-06 17:07:06.119 | 48.1 | 54.5 | 6.4 | 0 | |
| | FMStocks_DB.Ticker->VerifySymbol | 2003-02-06 17:07:06.115 | 2003-02-06 17:07:06.119 | 50.9 | 54.3 | 3.3 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.115 | 2003-02-06 17:07:06.119 | 51.0 | 54.2 | 3.2 | 0 | |
| | FMStocks_Bus.Broker->BuyStock | 2003-02-06 17:07:06.155 | 2003-02-06 17:07:06.173 | 90.2 | 109.0 | 18.8 | 0 | |
| | FMStocks_DB.Tx->AddBuyOrder | 2003-02-06 17:07:06.161 | 2003-02-06 17:07:06.166 | 96.2 | 101.9 | 5.7 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.161 | 2003-02-06 17:07:06.166 | 96.3 | 101.8 | 5.5 | 0 | |
| | FMStocks_DB.Ticker->GetPrice | 2003-02-06 17:07:06.166 | 2003-02-06 17:07:06.170 | 102.0 | 105.4 | 3.4 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.166 | 2003-02-06 17:07:06.170 | 102.1 | 105.4 | 3.3 | 0 | |
| | FMStocks_DB.Broker->Buy | 2003-02-06 17:07:06.170 | 2003-02-06 17:07:06.173 | 105.6 | 108.7 | 3.1 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.170 | 2003-02-06 17:07:06.173 | 105.7 | 108.6 | 2.9 | 0 | |
| FMStocks_Bus.Account | | 2003-02-06 17:07:06.274 | 2003-02-06 17:07:06.327 | 0.0 | 53.0 | 53.0 | 0 | |
| | FMStocks_Bus.Account->ListPositionsForSale | 2003-02-06 17:07:06.316 | 2003-02-06 17:07:06.322 | 42.1 | 48.0 | 5.9 | 0 | |
| | FMStocks_DB.Position->ListForSale | 2003-02-06 17:07:06.318 | 2003-02-06 17:07:06.321 | 44.5 | 47.5 | 2.9 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnRS | 2003-02-06 17:07:06.318 | 2003-02-06 17:07:06.321 | 44.6 | 47.4 | 2.8 | 0 | |
| FMStocks_Bus.Broker | | 2003-02-06 17:07:06.472 | 2003-02-06 17:07:06.550 | 0.0 | 78.7 | 78.7 | 0 | |
| | FMStocks_Bus.Broker->SellStock | 2003-02-06 17:07:06.511 | 2003-02-06 17:07:06.544 | 39.1 | 72.4 | 33.3 | 0 | |
| | FMStocks_DB.Tx->AddSellOrder | 2003-02-06 17:07:06.519 | 2003-02-06 17:07:06.528 | 47.0 | 56.4 | 9.3 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.519 | 2003-02-06 17:07:06.528 | 47.1 | 56.3 | 9.2 | 0 | |
| | FMStocks_DB.Ticker->GetPrice | 2003-02-06 17:07:06.528 | 2003-02-06 17:07:06.532 | 56.6 | 60.1 | 3.5 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.528 | 2003-02-06 17:07:06.532 | 56.6 | 60.0 | 3.4 | 0 | |
| | FMStocks_DB.Broker->Sell | 2003-02-06 17:07:06.532 | 2003-02-06 17:07:06.535 | 60.2 | 63.0 | 2.8 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.532 | 2003-02-06 17:07:06.534 | 60.3 | 62.9 | 2.6 | 0 | |
| | FMStocks_DB.Position->ListForAdjustment | 2003-02-06 17:07:06.537 | 2003-02-06 17:07:06.542 | 65.3 | 70.8 | 5.5 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnRS_RW | 2003-02-06 17:07:06.537 | 2003-02-06 17:07:06.542 | 65.4 | 70.7 | 5.3 | 0 | |

# Capturing real-world usage

## Solution: AppMetrics’ Method Analysis Report

- Hierarchy shows call sequence
- Durations for each step of  the sequence



Methods Analysis Report

AppMetrics Sample Diagnostic

2/6/2003 4:07:00 PM to 2/6/2003 4:09:00 PM

The current application is: FMStocks 2000 Core

| Transaction | Methods | StartTime | EndTime | Relative Start (ms) | Relative End (ms) | Duration (ms) | Error | Description |
|---|---|---|---|---|---|---|---|---|
| FMStocks_Bus.Ticker | | 2003-02-06 17:07:06.064 | 2003-02-06 17:07:06.180 | 0.0 | 115.3 | 115.3 | 0 | |
| | FMStocks_Bus.Ticker->VerifySymbol | 2003-02-06 17:07:06.112 | 2003-02-06 17:07:06.119 | 48.1 | 54.5 | 6.4 | 0 | |
| | FMStocks_DB.Ticker->VerifySymbol | 2003-02-06 17:07:06.115 | 2003-02-06 17:07:06.119 | 50.9 | 54.3 | 3.3 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.115 | 2003-02-06 17:07:06.119 | 51.0 | 54.2 | 3.2 | 0 | |
| | FMStocks_Bus.Broker->BuyStock | 2003-02-06 17:07:06.155 | 2003-02-06 17:07:06.173 | 90.2 | 109.0 | 18.8 | 0 | |
| | FMStocks_DB.Tx->AddBuyOrder | 2003-02-06 17:07:06.161 | 2003-02-06 17:07:06.166 | 96.2 | 101.9 | 5.7 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.161 | 2003-02-06 17:07:06.166 | 96.3 | 101.8 | 5.5 | 0 | |
| | FMStocks_DB.Ticker->GetPrice | 2003-02-06 17:07:06.166 | 2003-02-06 17:07:06.170 | 102.0 | 105.4 | 3.4 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.166 | 2003-02-06 17:07:06.170 | 102.1 | 105.4 | 3.3 | 0 | |
| | FMStocks_DB.Broker->Buy | 2003-02-06 17:07:06.170 | 2003-02-06 17:07:06.173 | 105.6 | 108.7 | 3.1 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.170 | 2003-02-06 17:07:06.173 | 105.7 | 108.6 | 2.9 | 0 | |
| FMStocks_Bus.Account | | 2003-02-06 17:07:06.274 | 2003-02-06 17:07:06.327 | 0.0 | 53.0 | 53.0 | 0 | |
| | FMStocks_Bus.Account->ListPositionsForSale | 2003-02-06 17:07:06.316 | 2003-02-06 17:07:06.322 | 42.1 | 48.0 | 5.9 | 0 | |
| | FMStocks_DB.Position->ListForSale | 2003-02-06 17:07:06.318 | 2003-02-06 17:07:06.321 | 44.5 | 47.5 | 2.9 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnRS | 2003-02-06 17:07:06.318 | 2003-02-06 17:07:06.321 | 44.6 | 47.4 | 2.8 | 0 | |
| FMStocks_Bus.Broker | | 2003-02-06 17:07:06.472 | 2003-02-06 17:07:06.550 | 0.0 | 78.7 | 78.7 | 0 | |
| | FMStocks_Bus.Broker->SellStock | 2003-02-06 17:07:06.511 | 2003-02-06 17:07:06.544 | 39.1 | 72.4 | 33.3 | 0 | |
| | FMStocks_DB.Tx->AddSellOrder | 2003-02-06 17:07:06.519 | 2003-02-06 17:07:06.528 | 47.0 | 56.4 | 9.3 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.519 | 2003-02-06 17:07:06.528 | 47.1 | 56.3 | 9.2 | 0 | |
| | FMStocks_DB.Ticker->GetPrice | 2003-02-06 17:07:06.528 | 2003-02-06 17:07:06.532 | 56.6 | 60.1 | 3.5 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.528 | 2003-02-06 17:07:06.532 | 56.6 | 60.0 | 3.4 | 0 | |
| | FMStocks_DB.Broker->Sell | 2003-02-06 17:07:06.532 | 2003-02-06 17:07:06.535 | 60.2 | 63.0 | 2.8 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnInteger | 2003-02-06 17:07:06.532 | 2003-02-06 17:07:06.534 | 60.3 | 62.9 | 2.6 | 0 | |
| | FMStocks_DB.Position->ListForAdjustment | 2003-02-06 17:07:06.537 | 2003-02-06 17:07:06.542 | 65.3 | 70.8 | 5.5 | 0 | |
| | FMStocks_DB.DBHelper->RunSPReturnRS_RW | 2003-02-06 17:07:06.537 | 2003-02-06 17:07:06.542 | 65.4 | 70.7 | 5.3 | 0 | |

# Capturing real-world usage

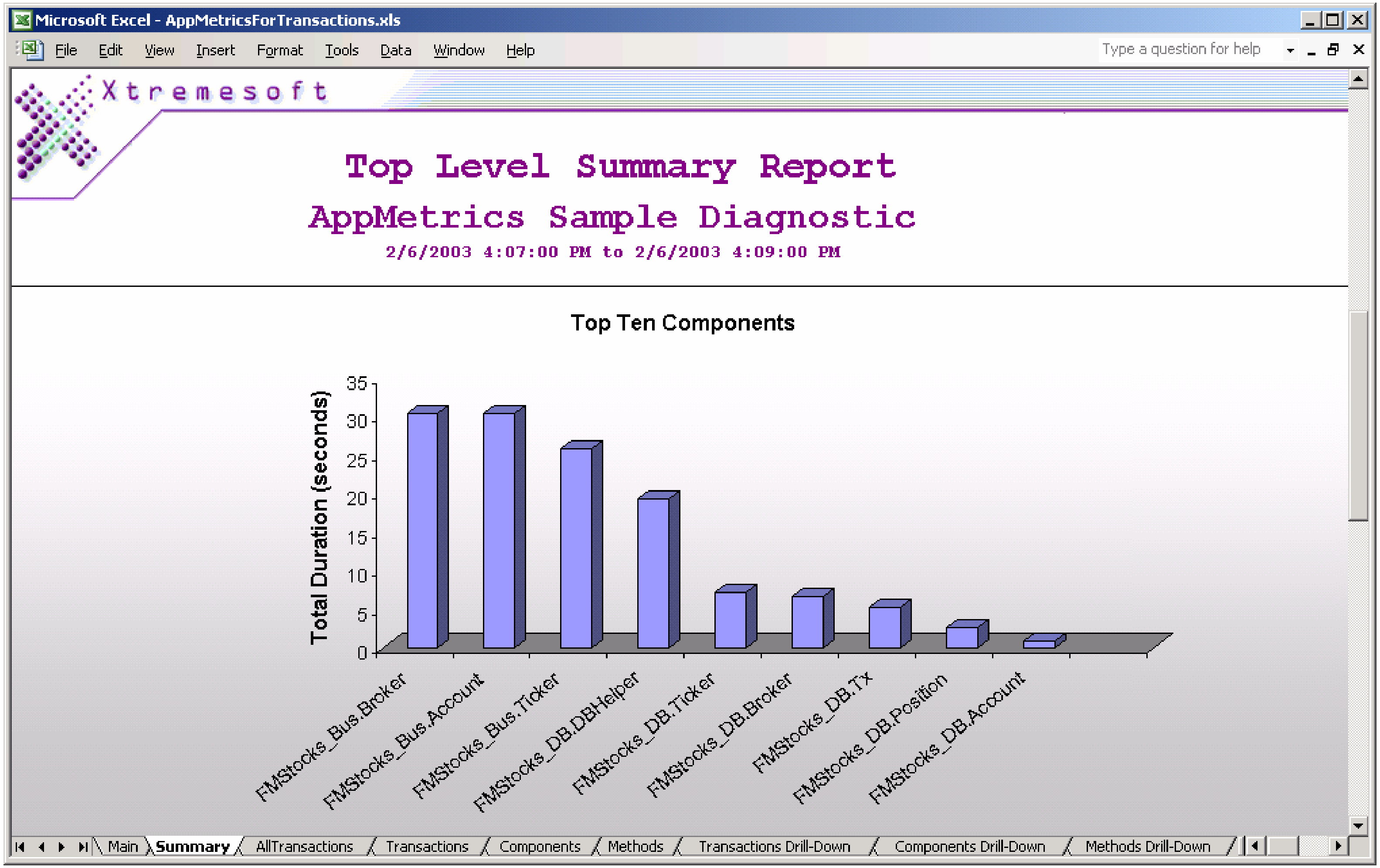
Our Customers accept that for the first time they can now ….

- *See which components are being used by what applications*
- *In what sequence components are being called*
- *See how often they are called in the real world and how long they typically take!*
- *Design tests that accurately reflect the real world, and therefore make them so much better and relevant.*

# Impact of change

- You can identify the responsible module…
- You can compare:
  - same component, different backend
  - same load, different software version
  - same database, new web page
  - And so forth…
- …and eliminate those modules (and team members) whose performance has not changed…

# If it looked like this last time…



Microsoft Excel - AppMetricsForTransactions.xls

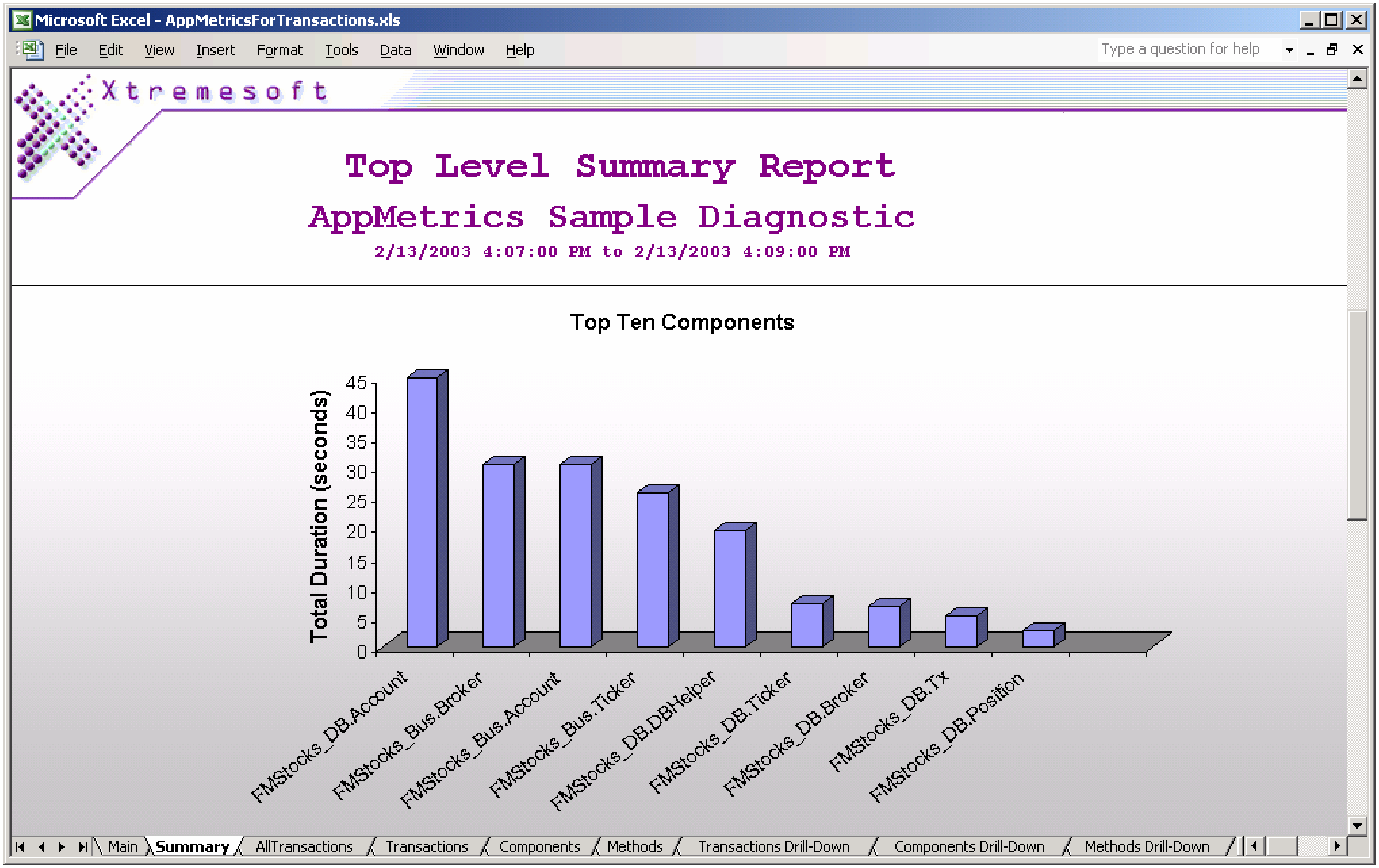
Xtremesoft

Top Level Summary Report

AppMetrics Sample Diagnostic

2/6/2003 4:07:00 PM to 2/6/2003 4:09:00 PM

# why does it look like this now?



Top Level Summary Report

AppMetrics Sample Diagnostic

2/13/2003 4:07:00 PM to 2/13/2003 4:09:00 PM

# Where to Optimize?

Problem:

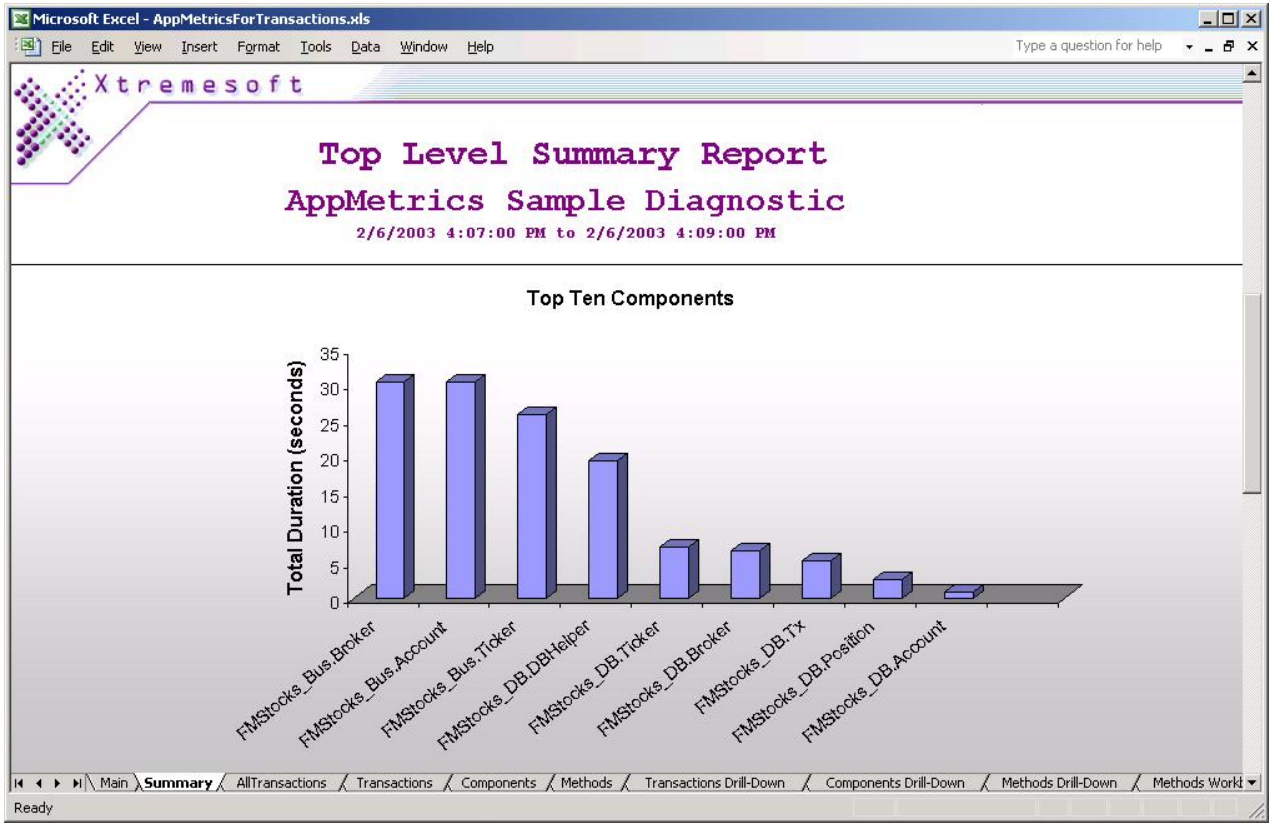
**I need to know my ‘most expensive’ components.**

When told the application is too slow and the application has tens, often hundreds of components, where do you start looking for possible candidates for optimization?

# Where to Optimize?

## Solution: AppMetrics' Top Ten Component Report

This report reveals which components are spending the most time running on the machine. The total duration of all component instances of each component type is calculated, and then the component types are sorted by total duration. This view helps you to choose which components are likely performance problems, because this algorithm will, for example rate a component that runs 100 times for an average of 1 second each time higher than a component that only runs 1 time, but for 50 seconds. The second component may need work, but it is less likely to be the cause of the problem.



Microsoft Excel - AppMetricsForTransactions.xls

Xtremesoft

Top Level Summary Report

AppMetrics Sample Diagnostic

2/6/2003 4:07:00 PM to 2/6/2003 4:09:00 PM

Top Ten Components

# Inconsistent Performance?

Problem:

**I don’t know my ‘typical’ method durations.**

You have to know,

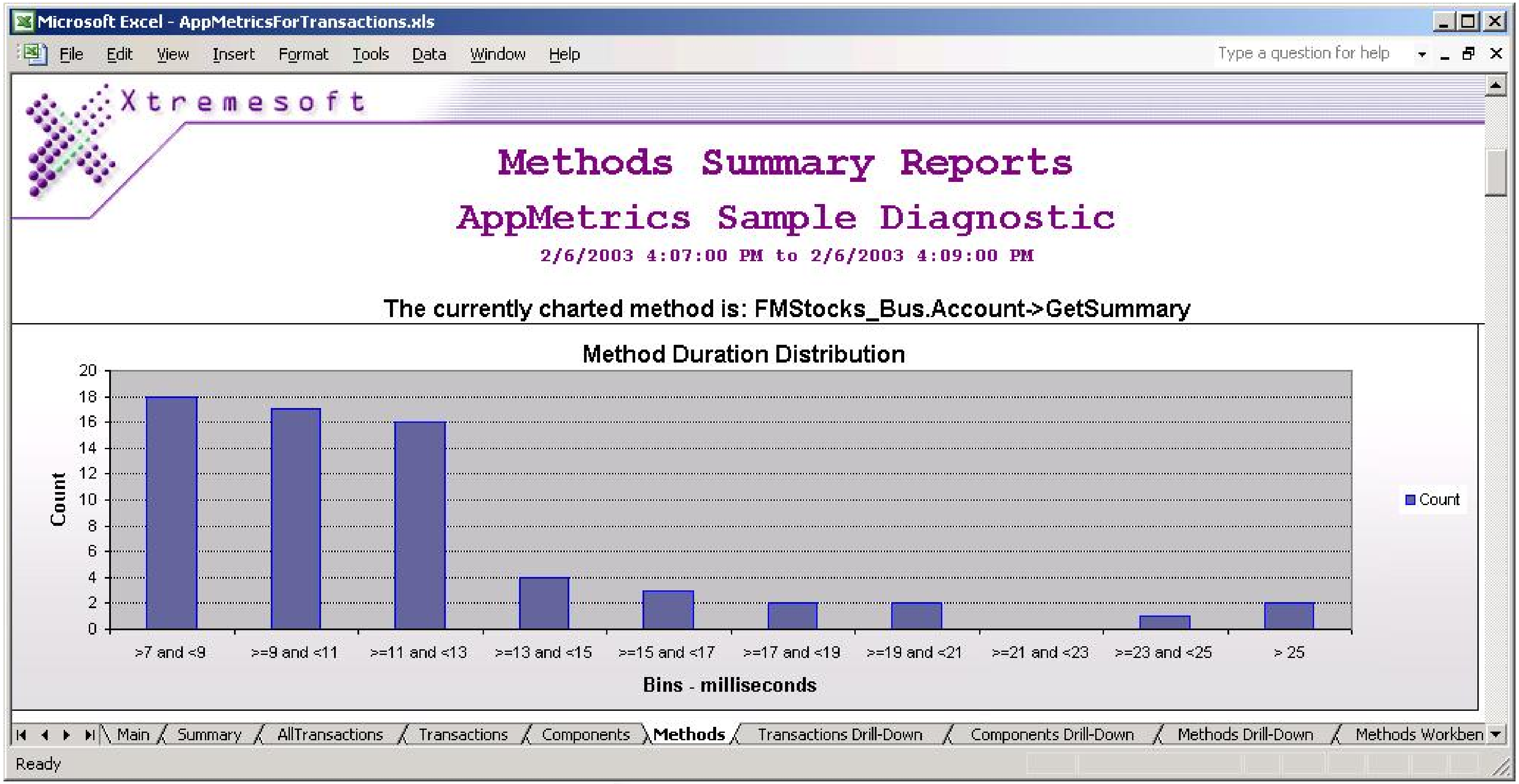
- what is if the average duration metric
- Is it representative of the typical duration, or if the average is being skewed by ‘outlier’ method instances.

If you are to identify the possible performance bottlenecks

# Inconsistent Performance?

## Solution: Method Duration Distribution Report

The Method Duration Distribution Report creates 10 evenly-sized ‘bins’, and displays the count of method instances that fell into each bin during the selected time window. This provides a view of the typical durations of methods, with an indication of the quantity of outliers -



Methods Summary Reports

AppMetrics Sample Diagnostic

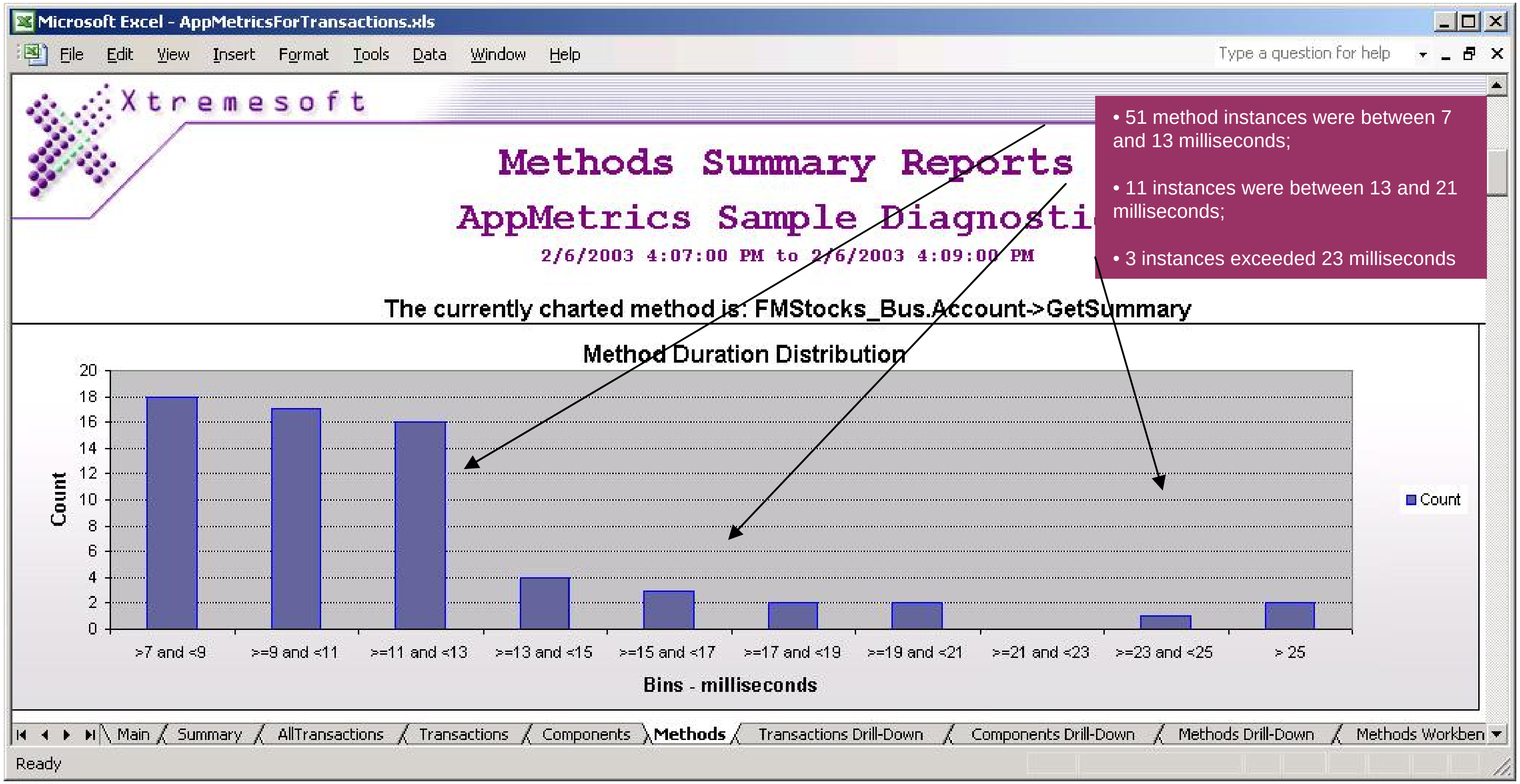
2/6/2003 4:07:00 PM to 2/6/2003 4:09:00 PM

The currently charted method is: FMStocks_Bus.Account->GetSummary

# Inconsistent Performance?

## Solution: Method Duration Distribution Report

The Method Duration Distribution Report creates 10 evenly-sized ‘bins’, and displays the count of method instances that fell into each bin during the selected time window.  This provides a view of the typical method durations of methods in production, with an indication of the quantity of outliers -



Microsoft Excel - AppMetricsForTransactions.xls

Methods Summary Reports

AppMetrics Sample Diagnosti

2/6/2003 4:07:00 PM to 2/6/2003 4:09:00 PM

The currently charted method is: FMStocks_Bus.Account->GetSummary

- 51 method instances were between 7 and 13 milliseconds;
- 11 instances were between 13 and 21 milliseconds;
- 3 instances exceeded 23 milliseconds

# Conclusion

- A variety of common COM+ scenarios are quickly and easily addressed with AppMetrics
- Increased Application uptime increases return on investment in that application
- Staff Productivity increases because the time to resolve problems is greatly reduced.
- QA is a critical element in this – AppMetrics is a great addition to their understanding of the critical applications