It takes a lot of time to understand a new codebase
• New programmers spend 80% of their time understanding code
• Experienced programmers forget what they wrote six months ago
• Documentation becomes outdated ...

It can take a lot of time to learn a new tool
• Familiar and intuitive Diagrams help in understanding code
• Need to support easy exploration around code details
• Integration into user tools can be helpful (source repository, search results, navigation history, etc)

“Interactive Exploration of Compacted Visualizations for Understanding Behavior in Complex Software”

Elizabeth L. Murnane, emurnane@architexa.com
Vineet Sinha, vineet@architexa.com

How do the diagrams remain useful when the code gets complicated?

By selecting it, a methods button is shown that can be pushed to view the methods the class contains.

Selecting a method from the list adds it to the diagram and allows the user to being exploring how the system operates.

A user can start simply with a single class.

Editors icons and locations indicate the type of interactions that they are used to display.

Automatic and manual reduction of less relevant information.

It is easy for a sequence diagram to get large fast, so the diagram automatically condenses components and removes irrelevant information.

A user can also hide, compact, and delete components directly.

How do the diagrams remain useful when the code gets complicated?

Exploration

Users can interact with an incrementally explore and build a diagram piece by piece.

Integration and Automation

Tight integration with the development environment makes use familiar and intuitive and minimizes learning curve

Automatic eases navigation process while allowing users to maintain control

Integration and Automation

Also:

• Integration with CVS and Subversion to generate diagrams based on uncommitted changes and revision histories.
• Option to automatically display entire lifecycle of a method (all the calls a method makes, the calls those methods make, and so on).
• Display of all interactions among components with a single button push.

Layers Diagrams

Architexa is an MIT spinoff that helps developers make sense of code by providing an interactive reverse engineering engine.

See: http://www.architexa.com
E-mail: info@architexa.com

Conditional blocks

Conditional blocks indicated by a highlight that is only shown on an event over and that moves the code components underneath it.

Don’t want to display the same statement inside a long multiple times or a large number of parts of an if-else block when one is actually executed. So, long conditions and only displayed once, and only one execution branch (by default the first one). If the initial if statement is true it is displayed while it is indicated that other paths are possible but are not shown.

Hiding of library code

Library code is code not defined in the developer’s code but rather in an external library. By providing a button, all components corresponding to Library code are hidden from view in order to facilitate the diagram and emphasize the parts most relevant to the user (those corresponding to elements in the own code).

Also:

• Basic definition: only deletes the selected component and anything that cannot exist without it (e.g. invocation cannot exist without a declaration and method can not exist over their distance is defined).
• By clicking extended definition, the selected component is displayed along with any other components whose existence depends on it. If there are no subsequent calls that these methods make.

Chained calls

Chained calls tools that move the review of the previous invocation are displayed as a single message. The user can select one of the methods in the chain in order to expand and display it separately in the diagram and see exactly which method statements are executed after each other.

Backward messages

Chained calls tools that move the review of the previous invocation are displayed as a single message. The user can select one of the methods in the chain in order to expand and display it separately in the diagram and see exactly which method statements are executed after each other.

Automatic and manual reduction of less relevant information.

It is easy for a sequence diagram to get large fast, so the diagram automatically condenses components and removes irrelevant information.

A user can also hide, compact, and delete components directly.

Chained calls

The call is only shown when the reason is located over one of the figures.

Backward messages

Chained calls tools that move the review of the previous invocation are displayed as a single message. The user can select one of the methods in the chain in order to expand and display it separately in the diagram and see exactly which method statements are executed after each other.

Automatic and manual reduction of less relevant information.

It is easy for a sequence diagram to get large fast, so the diagram automatically condenses components and removes irrelevant information.

A user can also hide, compact, and delete components directly.