Conversion Tools - Feature #3699

improve call-graph visualization to make it much more usable

08/23/2018 10:03 AM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Conversion Tools - Feature #3277: implement call-graph v3	**Closed**
Related to Conversion Tools - Bug #3742: IE11 doesn't render the flowchart or...	**New**

History

#1 - 08/23/2018 10:22 AM - Greg Shah

The current call-graph visualization becomes unusable for even small number of programs when complex frameworks (such as ADM/ADM2) are in use.

The biggest issue is the inefficient use of space. The need to avoid overlapping sibling nodes led to an approach where we created cells that acted like jails (contained nodes can't move past the cell boundaries). The problem here is that the cells must be big enough to contain the largest sibling nodes leaving most of the cells greatly oversized for the actual nodes being displayed. This makes the container node very large and wastes much of the interior space.

Even at the top (page) level, the jail cells approach leaves the result very difficult to use. This was also made more complicated because we had limited control over the distance between nodes and we needed to use the controls that did exist in a way that helped us avoid collisions. This made a bad situation worse.

I realize that we can also improve the visualization by enhancing the filtering, search and the expand/collapse capabilities. This is something we do want to explore as already documented in #3277.

But the core issue still exists. We either have to have a much more sophisticated approach to cells OR take a different approach.

I've recently found a potential framework that provides a different approach. It is called cola.js (see http://ialab.it.monash.edu/webcola/). What is really nice about this is that it implements the non-overlapping layout including alignment and grouping but with a significantly better use of the space. The resulting graph will be contained in a much smaller space, which should significantly improve the usability. What is critical, is that this works as a drop-in replacement for the D3 force directed layout, which is exactly what we need.

The down side is that they say it is really slow for larger graphs, so we may have to just be patient while it is loading.

I'd like to explore using this. It should be pretty straightforward to enable, but we will have to remove our current jail cell approach to make it work.

#2 - 08/23/2018 10:44 AM - Greg Shah

- Related to Feature #3277: implement call-graph v3 added

#3 - 09/07/2018 04:23 PM - Constantin Asofiei

- Status changed from New to WIP

- Assignee set to Constantin Asofiei

- % Done changed from 0 to 80

The changes for this task are in 3701a.

#4 - 10/03/2018 10:10 AM - Constantin Asofiei

Greg, please take a look at 3701a rev 11303 - I don't think I can find something better than this.

Note that cola.js will not work with the 'Include All' option - the internal layout computation gets too expensive.

#5 - 10/03/2018 11:54 AM - Greg Shah

Do you want a code review? History entries and javadoc seem to be missing, so I can wait until you are ready.

I'm running ant report_server on Hotel GUI right now so I can test the results.

#6 - 10/03/2018 11:54 AM - Constantin Asofiei

Greg Shah wrote:

> Do you want a code review? History entries and javadoc seem to be missing, so I can wait until you are ready.

Not yet, I'll let you know. I need to finish the legend for the flowchart and code cleanup/javadocs.

#7 - 10/03/2018 01:10 PM - Constantin Asofiei

The flowchart legend is in 11304.

#8 - 10/03/2018 01:44 PM - Greg Shah

This is very cool! I think it is a nice improvement for call graph and the flow chart support is in a good place for a v1.0.

Please make sure that the code reports are brought up by default when you start analytics. Otherwise, it is good to go. I'll do the code review when you are ready.

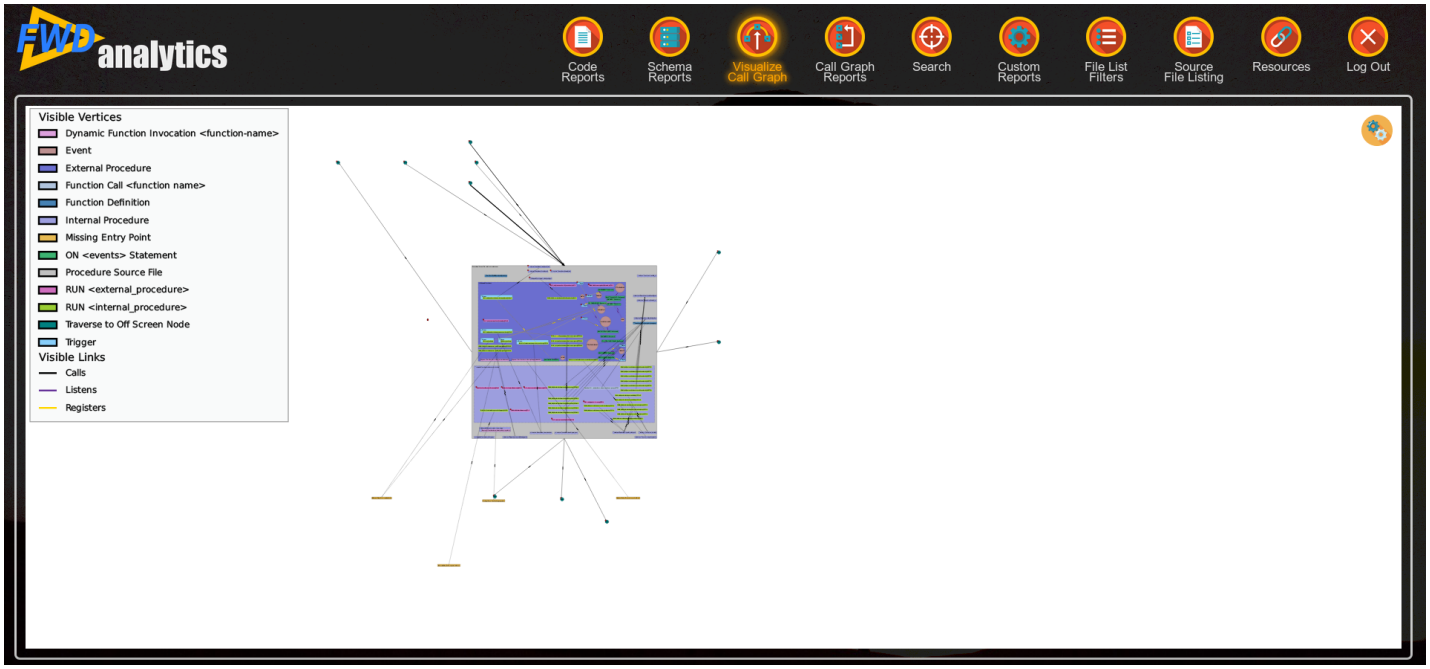
#9 - 10/03/2018 01:46 PM - Greg Shah

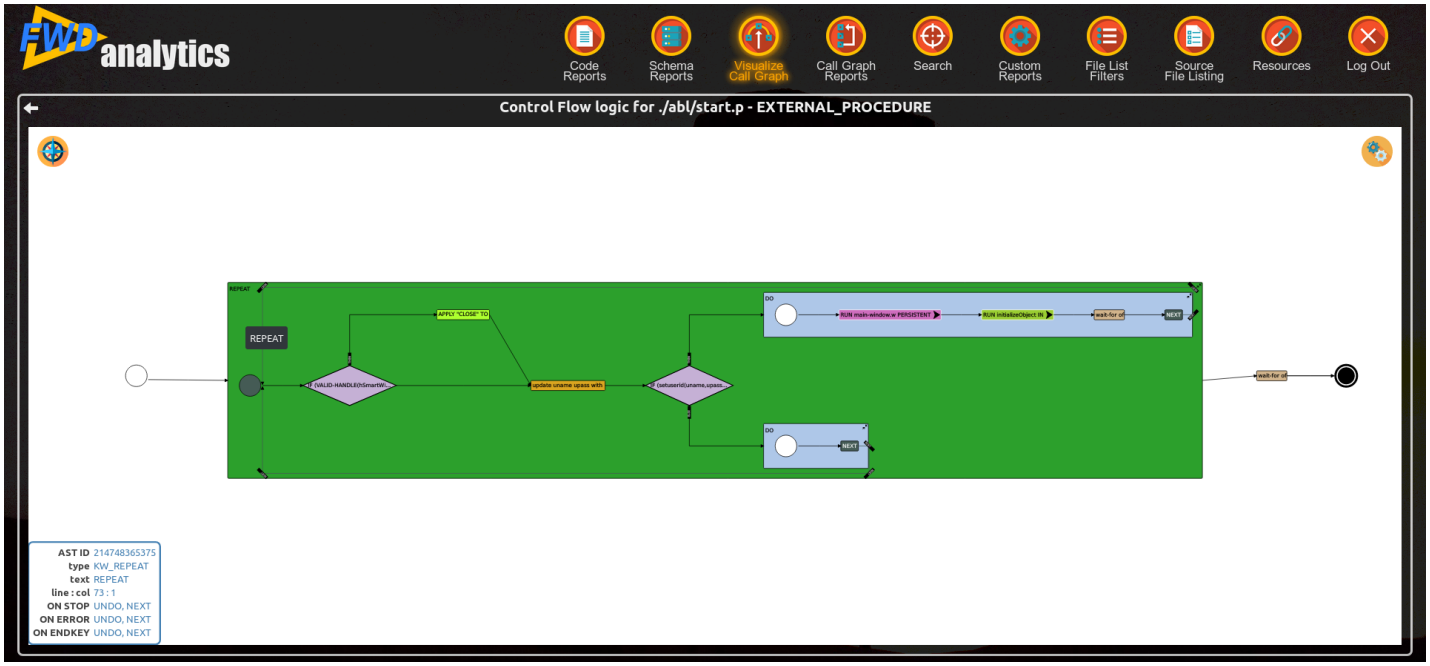
- File fwd_code_analytics_code_control_flow_logic_main_window_external_proc_screen.png added

- File fwd_code_analytics_code_control_flow_logic_main_window_external_proc.png added

- File fwd_code_analytics_call_graph_visualization_zoomed_out_graph_only.png added

- File fwd_code_analytics_call_graph_visualization_screen.png added


analytics
Code Reports
Schema Reports
Visualize Call Graph
Call Graph Reports
Search
Custom Reports
File List Filters
Source File Listing
Resources
Log Out
Visible Vertices
Dynamic Function Invocation <function-name>
Event
External Procedure
Function Call <function name>
Function Definition
Internal Procedure
Missing Entry Point
ON <events> Statement
Procedure Source File
RUN <external_procedure>
RUN <internal_procedure>
Traverse to Off Screen Node
Trigger
Visible Links
Calls
Listens
Registers


analytics
Code Reports
Schema Reports
Visualize Call Graph
Call Graph Reports
Search
Custom Reports
File List Filters
Source File Listing
Resources
Log Out
Control Flow logic for ./abl/start.p - EXTERNAL_PROCEDURE
REPEAT
AST ID 214748365375
type KW_REPEAT
text REPEAT
line : col 73 : 1
ON STOP UNDO, NEXT
ON ERROR UNDO, NEXT
ON ENDKEY UNDO, NEXT

#10 - 10/03/2018 05:22 PM - Constantin Asofiei

- Status changed from WIP to Review

- % Done changed from 80 to 100

The code reports are now the default view after login.

Please review 3701a rev 11305.

#11 - 10/04/2018 12:07 PM - Greg Shah

Code Review Task Branch3701a Revision 11305

The results are really good. FYI, considering the amount of change in callgraph.js and the amount of code in flowchart.js, I haven't done a comprehensive analysis of all code there.

1. Why are the private methods describeNode() and describeTooltip() in CallGraphApi being moved up into the public section? Were you planning to expose those as public?

2. The usage of arrow function syntax (=>) might be a problem. I think some browsers still don't support it. For example, at least at the beginning of the year IE v11 did not support it. Please check this. I think IE support is required.

#12 - 10/04/2018 12:55 PM - Constantin Asofiei

Greg Shah wrote:

> 1. Why are the private methods describeNode() and describeTooltip() in CallGraphApi being moved up into the public section? Were you planning to expose those as public?

Since ever I thought that static methods should be before instance, regardless of access modifier... I'll move them back.

> 2. The usage of arrow function syntax (=>) might be a problem. I think some browsers still don't support it. For example, at least at the beginning of the year IE v11 did not support it. Please check this. I think IE support is required.

Argh, IE doesn't support lambda => notation. I'll fix this.

#13 - 10/04/2018 01:46 PM - Constantin Asofiei

IE11 has problems rendering the links (which are polyline SVG elements). I think is related to stroke-width, but I'm not sure. Should I chase this

down?

#14 - 10/04/2018 01:54 PM - Greg Shah

Constantin Asofiei wrote:

> IE11 has problems rendering the links (which are polyline SVG elements). I think is related to stroke-width, but I'm not sure. Should I chase this down?

If it is quick, yes. If it gets too involved we can discuss.

#15 - 10/04/2018 02:12 PM - Constantin Asofiei

Greg Shah wrote:

> Constantin Asofiei wrote:
>
> > IE11 has problems rendering the links (which are polyline SVG elements). I think is related to stroke-width, but I'm not sure. Should I chase this down?
>
> If it is quick, yes. If it gets too involved we can discuss.

If I change the stroke-width from 1 to 2, the polyline is rendered; back to 1, is rendered again. Something is preventing IE11 to render the polyline, I can't find anything online.

This is happening for the callgraph and flowchart, but if you open another flowchart and go back to the previous (which can be the callgraph or flowchart), then that will have the links rendered, now. Something is preventing IE to render the polyline elements when the SVG is first shown.

#16 - 10/04/2018 02:24 PM - Greg Shah

Just open a task for this and we won't work on it now. It seems likely this is just a browser bug and may get fixed in a later version.

#17 - 10/04/2018 02:34 PM - Constantin Asofiei

- Related to Bug #3742: IE11 doesn't render the flowchart or callgraph links added

#18 - 10/04/2018 02:38 PM - Constantin Asofiei

Greg Shah wrote:

> Just open a task for this and we won't work on it now. It seems likely this is just a browser bug and may get fixed in a later version.

Done, see #3742.

3701a rev 11306 fixes the IE11 compatibility issues (=> and other JS issues, plus a toolbox positioning issue).

I can merge it to trunk if you are OK with it.

What is left is changing FlowChart.dumpExpression to use the anti-parser (now it uses a crude/incomplete way of anti-parsing an AST).

#19 - 10/04/2018 03:01 PM - Greg Shah

I'm OK with the changes. Please merge to trunk. When 3696 is finished we'll handle the anti-parsing integration.

#20 - 10/04/2018 03:14 PM - Constantin Asofiei

3701a was merged to trunk rev 11285 and archived.

#21 - 10/04/2018 03:15 PM - Greg Shah

- Status changed from Review to Closed

#22 - 04/06/2022 05:20 PM - Greg Shah

- File forceatlas2_graph_network_layout_algorithm.pdf added

I found an interesting article on the ForceAtlas2 graph layout algorithm which is something we might want to consider for improving the layout of the call graph. I've also attached the paper here.

Files

fwd_code_analytics_call_graph_visualization_zoomed_out_graph_only.png	69.6 KB	10/03/2018	Greg Shah
fwd_code_analytics_code_control_flow_logic_main_window_external_proc.png	35.5 KB	10/03/2018	Greg Shah
fwd_code_analytics_code_control_flow_logic_main_window_external_proc_screen.png	364 KB	10/03/2018	Greg Shah
fwd_code_analytics_call_graph_visualization_screen.png	389 KB	10/03/2018	Greg Shah
forceatlas2_graph_network_layout_algorithm.pdf	2.17 MB	04/06/2022	Greg Shah