Conversion Tools - Feature #3277

Feature # 1511 (New): Reporting v3

implement call-graph v3

04/19/2017 04:46 PM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Greg Shah	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Reporting 3.0		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Conversion Tools - Feature #2251: improve the call graph generation	**Closed**	**03/03/2014**
Related to Conversion Tools - Feature #3359: enhance the callgraph to better ...	**New**	
Related to Conversion Tools - Feature #3695: instrument 4GL code for automate...	**New**	
Related to Conversion Tools - Feature #3697: provide a web UI to edit/manage ...	**New**	
Related to Conversion Tools - Feature #3699: improve call-graph visualization...	**Closed**	

History

#1 - 04/19/2017 04:46 PM - Greg Shah

- Related to Feature #2251: improve the call graph generation added

#2 - 04/20/2017 10:32 AM - Greg Shah

This work is intended to provide a web UI to display and explore a more complete version of the call graph.

Objectives

- expand the current call graph
  - fix bugs (see #2357 and #2356)
  - ~~make the ADM/ADM2 linkage processing a standard part of the call graph rules~~
  - ~~fix any open issues with the ADM/ADM2 rules~~
  - ~~add all entry points as full vertices in the database, with enough information to associate them back to the original AST nodes)~~
  - ~~add any missing call sites (at a minimum, all the internal call sites are not there) as vertices in the database, with enough information to associate them back to the original AST nodes)~~
  - ~~create the appropriate edges to link all the vertices, including linkages back to the external proc itself~~
- ~~web UI~~
  - ~~written in HTML5/CSS/Javascript with calls to the server API to get the graph data for display, data will be passed and returned as a JSON object graph)~~
  - ~~force directed graph as the display (to give a good default layout), the plan is to use D3 for this~~
  - ~~each vertex and edge will have a distinct visual rendering and will include some text data about the associated object~~
  - user will have dynamic control over:
    - ~~the amount of the graph that is displayed at any one time~~
    - whether individual external procedure nodes are expanded (all contained call sites/targets are displayed) or collapsed
    - filtering what is displayed
    - searching for specific matches to "move" the parts of the graph that are displayed
- ~~user selection/interaction can be used to traverse:~~
  - ~~linkages that are not currently displayed such that nodes that are "off the screen" will be moved onto the screen~~
  - ~~to/from the source code view (cache file/AST view) based on data that associates with AST nodes and/or line/column numbers~~
- ~~the reporting v3 engine will host both the code/schemata reports and the call graph UI~~
  - ~~same JS/Java RPC mechanism~~
  - ~~call graph API (different from the rest of the reporting)~~
- ~~run call graph processing by default as part of reporting setup phase~~
- ~~rework all previous outputs (call graph reports) as outputs in the web UI~~
- possibly provide a UI to configure the root nodes and hints (and re-run), making it easier to iteratively complete the graph (e.g. to get rid of all ambiguous references)

#3 - 04/20/2017 10:35 AM - Greg Shah

Constantin: I am refreshing my understanding of the current implementation. As I dig into the code, it would be very helpful if you would prepare a list of those items that I would need to address to handle the "expand the current call graph" objectives. In particular, the list of missing call linkages (internal and external) and a list of the linkages that need to be reworked into the "normalized" graph (where all linkage types are first class graph vertices or edges).

#4 - 04/20/2017 11:34 AM - Greg Shah

One question I am considering is whether we should move to a newer graph database implementation. It seems like the right time to make the change.

FWD trunk revision 11149 uses Titan 0.4.2 and Tinkerpop 2.4.

Tinkerpop has subsequently been moved to Apache as a separate project:

http://tinkerpop.apache.org/ (main project)
http://www.tinkerpop.com/docs/javadocs/blueprints/2.4.0/ (2.4 javadoc)
http://tinkerpop.apache.org/javadocs/3.2.2/core/index.html (3.2.2 javadoc)
http://tinkerpop.apache.org/docs/current/reference/ (documentation)

The current version is Tinkerpop 3 which seems improved over v2.

Titan:

Main Site
Last Version
Titan Documentation
Tital 0.4.2 Javadoc (the main documentation seems to no longer exist for this version and the code is very different from 1.0.0)

While Tinkerpop has moved ahead since we implemented it, Titan is a different story. Titan itself is a dead project:

Beyond Titan: The Evolution of DataStax's New Graph Database
Is TitanDB dead?
JanusGraph connects the past and future of Titan

JanusGraph is the successor to Titan. It does seem to be getting some activity.

http://janusgraph.org/
https://github.com/JanusGraph/janusgraph/releases
http://docs.janusgraph.org/0.1.0-SNAPSHOT/index.html

It seems relatively straightforward to shift to JanusGraph, but it is unclear how much rework is required for the query usage to match the Tinkerpop 3 standard. On the other hand, the move to a fully normalized graph will probably require extensive changes to all queries so it may not really be an issue.

Another consideration is whether a different graph database makes sense. I still want to avoid Neo4J. But I've been looking at OrientDB for some time and it does have some advantages:

- The security model seems to be much better (i.e. there is a real security model vs no security model).
- Encryption support (at the database level).
- It is the second most popular graph database.
- It can support documents and key/value in addition to being a full property graph database.
- Each vertex can actually be a document, allowing the storage of much more complex (including nested) data.
- Richer data type options than just the most primitive types. It supports a wide range of types including fixed point decimals, date types, collections, maps, binary data and some form of objects.
- Schema-full, schema-less and hybrid options.

Both databases have some level of scalability built in, they both have strong transaction processing and ACID.

OrientDB has replication, sharding, clustering etc... but scalability is certainly an advantage for JanusGraph which is designed with Big Data in mind. On the other hand, I doubt we need that kind of scalability.

Although OrientDB does support Tinkerpop 3, the support does seem to come from an external project which has documentation that suggests the implementation is not 100%.

https://db-engines.com/en/system/JanusGraph%3BOrientDB
http://vschart.com/compare/titan-database/vs/orientdb
http://pettergraff.blogspot.com/2013/12/orientdb-thanks.html
http://orientdb.com/orientdb-vs-neo4j/
https://www.3pillarglobal.com/insights/a-quick-look-into-the-popular-graph-databases
https://groups.google.com/forum/#!msg/orient-database/CpPh42ukfH4/Tf6Nz93bmzkJ

My sense is that OrientDB may be the better solution. There are two concerns:

- It will take some effort to shift over. Is the effort worth it?
- Will the Tinkerpop support be limited or cause problems? Since there are additional capabilities in OrientDB, Tinkerpop may not expose all of them, may have a buggy implementation or otherwise it might not be the best query language to use.

Constantin: thoughts?

#5 - 04/20/2017 11:37 AM - Greg Shah

Interesting D3 samples:

http://vowl.visualdataweb.org/webvowl/ (very close, code is at https://github.com/VisualDataWeb/WebVOWL)
https://github.com/nylen/d3-process-map (nice legend, integrated text)
http://orgo.stolarsky.com/ (nice selection highlight)
http://mbostock.github.io/d3/talk/20111116/force-collapsible.html (collapsible sections)
https://bl.ocks.org/mbostock/4062045 (most simple example)
https://timebandit.github.io/graphSub/

#6 - 04/20/2017 12:27 PM - Constantin Asofiei

Greg, a good read is the documentation in callgraph/callgraph_lib.rules - it documents all the call sites which are included in the graph; the implemented ones are the list in the first paragraph from https://proj.goldencode.com/issues/2251#note-31 I need to go through the list from https://proj.goldencode.com/issues/2251#note-7 and see what exactly is missing.

If you add new call sites, see add_external_node - there the external node is created with a specific type, depending on the callSiteKey.

> a list of the linkages that need to be reworked into the "normalized" graph (where all linkage types are first class graph vertices or edges).

I don't understand this one (sorry, is been a while since I worked on the callgraph) - can you give an example/more details?

#7 - 04/20/2017 01:27 PM - Constantin Asofiei

Greg Shah wrote:

> My sense is that OrientDB may be the better solution. There are two concerns:
>
> - It will take some effort to shift over. Is the effort worth it?

I think so; as TitanDB is dead, OrientDB looks like is a good alternative; especially that at some point we want to move the conversion to work in the DB. Note that OrientDB v2.x.x still uses Tinkerpop2 (Blueprints 2.6.0) - and OrientDB v3 has releases for both Tinkperpop 2 and Tinkerpop 3.

Anyway, I've looked at the Blueprints -> Gremlin Structure (Tinkerpop2 to Tinkerpop3) equivalent and the APIs for i.e. Vertex/Edge interfaces have changed... so if we switch to TP3 there will be changes both in TRPL and java code; also, the query mechanism has changed completely, now I think

it relies on GraphTraversal (http://tinkerpop.apache.org/javadocs/current/full/org/apache/tinkerpop/gremlin/process/traversal/dsl/graph/GraphTraversal.html) which is a different than how the GraphQuery in v2 worked.

- Will the Tinkerpop support be limited or cause problems? Since there are additional capabilities in OrientDB, Tinkerpop may not expose all of them, may have a buggy implementation or otherwise it might not be the best query language to use.

They claim OrientDB 3.0 M1 has "official Tinkperpop support", so I suggest to get the release which uses Tinkerpop 2.6 and work with that for now.

#8 - 04/20/2017 04:05 PM - Greg Shah

Something that makes me concerned about OrientDB:

http://orientdbleaks.blogspot.com/2016/06/one-year-of-orientdb-leaks.html

#9 - 04/20/2017 04:09 PM - Greg Shah

Having a mechanism for Object to Graph Mapping (OGM) seems useful. Unfortunately, the existing solutions are relatively immature.

http://hibernate.org/ogm/ (doesn't yet support OrientDB)
http://in.relation.to/2016/05/23/meet-sergey-chernolyas-hibenrate-ogm-orientdb/
http://jotschi.de/2015/06/10/graphdb-ogm-comparison/
https://github.com/Syncleus/Ferma

#10 - 04/20/2017 04:09 PM - Greg Shah

Interesting article on using the later Tinkerpop with the older OrientDB:

https://christinemdraper.wordpress.com/2017/04/02/using-tinkerpopgremlin-3-with-orientdb-2/

#11 - 04/20/2017 04:26 PM - Greg Shah

> a list of the linkages that need to be reworked into the "normalized" graph (where all linkage types are first class graph vertices or edges).

I don't understand this one (sorry, is been a while since I worked on the callgraph) - can you give an example/more details?

I'm referring to the design approach as described in #2251-59 where:

> In the graph DB, from the AST files associated with an external program, only the root AST node (for the external program) has an associated node in the DB. The call-sites don't have such a node; instead, the info about the call-site is specified at the edge's properties, the call-site-key property.

I want to fix this so that all call-sites have vertices in the graph database. This is what I'm calling "normalized".

#12 - 04/21/2017 04:02 PM - Constantin Asofiei

Greg Shah wrote:

> a list of the linkages that need to be reworked into the "normalized" graph (where all linkage types are first class graph vertices or edges).

I don't understand this one (sorry, is been a while since I worked on the callgraph) - can you give an example/more details?

I'm referring to the design approach as described in #2251-59 where:

> In the graph DB, from the AST files associated with an external program, only the root AST node (for the external program) has an associated node in the DB. The call-sites don't have such a node; instead, the info about the call-site is specified at the edge's properties, the call-site-key property.

> I want to fix this so that all call-sites have vertices in the graph database. This is what I'm calling "normalized".

This description is not valid, the actual design does add a vertex for all call-sites; see this description in callgraph_lib.rules:

```
A. The supported node-type values in the callgraph DB are:
1. "node-type" values which identify legacy source code or schema:
- INCLUDE - to identify a physical include file. Has as "node-key" the "filename" string and
in the "filename" property it holds the project-home relative name of this include file.
- EXTERNAL_PROCEDURE - to identify an existing external program.  Has as "node-key" the
"filename" string and in the "filename" property it holds the project-home relative name of this
external program.  The "node-id" property holds the AST id of this external program.
- TABLE - to identify a table from the permanent schema.  Has as "node-key" the
"schemaname" string and in the "schemaname" property it holds the schema name of this table.
The "node-id" property holds the AST id of this schema table, in the associated .schema file.
Note that currently only schema-tables which have a schema trigger are loaded.
- the call-site's AST type, for other nodes.
```

Also, see the link_to_target function which uses create_call_site_node - it will create a vertex for the callSite AST and create an edge between the out vertex (which is the external program) and the new vertex for callSite.

Also, note that the out var name is reused after <rule>out = execLib("create_call_site_node", out, callSite)</rule> is called - thus out references the callSite's vertex after this call, and not the external program.

#13 - 04/26/2017 02:05 PM - Greg Shah

1. The CGW.CallGraph.addTarget() does not add to the traversed list. Is that expected?

2. I don't understand the TODO comment in link_to_include. There is no loop there, so it seems safe.

#14 - 04/26/2017 02:18 PM - Constantin Asofiei

Greg Shah wrote:

> 1. The CGW.CallGraph.addTarget() does not add to the traversed list. Is that expected?

I think you are mixing a callgraph-generation method and a method used by reporting:

1. addTarget() is used only by link_to_target in callgraph_lib.rules
2. traversed in CGW$CallGraph.traversalIterator is used only by reporting in list_dead_files.rules

So, these two APIs don't intersect.

> 2. I don't understand the TODO comment in link_to_include. There is no loop there, so it seems safe.

Correct, I think the comment should have been removed (and at some point might have been a loop).

#15 - 05/02/2017 04:21 PM - Greg Shah

In link_to_include or print_call_site we see this code: cgw.edgeIterator(out, "include-location", "call-site-id", ...). In what cases will this use return more than one result? A specific AST node can only originate in a single location, include or "direct" from the procedure source file.

This use of an iterator seems counter-intuitive unless I am misunderstanding things.

#16 - 05/02/2017 05:17 PM - Constantin Asofiei

Greg Shah wrote:

> In link_to_include or print_call_site we see this code: cgw.edgeIterator(out, "include-location", "call-site-id", ...). In what cases will this use return more than one result? A specific AST node can only originate in a single location, include or "direct" from the procedure source file.

You are correct, the iterator returned by edgeIterator is used either to check if an edge doesn't exists and create it if it doesn't (in link_to_include) , or get the edge if it does exist (in print_call_site).

> This use of an iterator seems counter-intuitive unless I am misunderstanding things.

Yes, it looks counter-intuitive to me too... I think I wanted at that time to keep the APIs in CallGraphWorker as low-level as possible, and build the rules using them.

#17 - 05/05/2017 10:38 AM - Greg Shah

In JanusGraphDB.createIndex*Impl(), the text parameter is described as this:

```
    * @param    text
    *           {@code true} if this index allows text-search.
```

But it is implemented like this:

```
            if (text)
            {
               km.indexed(indexNames[i], indexed,
                           Parameter.of(Mapping.MAPPING_PREFIX, Mapping.STRING));
            }
```

Mapping.STRING means "that a string be indexed as a whole and not tokenized", which is the opposite of full-text searching. Mapping.STRING is useful for exact string matching. Mapping.TEXT specifies full-text searching, where the string is tokenized and one can use partial matching techniques (which will be fast). In fact, I don't really see those text search techniques being used, except possibly in CGW.resolveExternalPrograms() which uses the Text.PREFIX option.

The titan 0.4.2 version is so old that its core documentation no longer exists. I can only find the javadoc. The titan 1.0.0 version is much different, its APIs are greatly changed and classes/interfaces are all in different packages and have different names. 1.0.0 is very close to the janusgraph stuff (almost identical). Because of this it is hard to ensure that everything I am reading is matching up with the older 0.4.2 version.

As far as I can tell, the javadoc parameter is probably just a bit misleading. Am I missing something?

#18 - 05/05/2017 12:55 PM - Constantin Asofiei

Greg Shah wrote:

> As far as I can tell, the javadoc parameter is probably just a bit misleading. Am I missing something?

You are correct - there is no full-text search used, only prefix-based (i.e. column LIKE 'something%').

This prefix-search is used also in in CGW.resolveExternalPrograms:

```
for (Vertex v : db.query().has("reverse-filename", Text.PREFIX, search).vertices())
```

This will search for an exact match for a given filename, but as the filename is the last one in the path, there is a reverse-filename property which contains the entire path reversed - thus allowing to use a prefix-match against it (with the reversed filename received as param).

#19 - 05/09/2017 01:20 PM - Greg Shah

- File janusgraph_0.2.0_javadoc.zip added

Attached is the JanusGraph 0.2.0 javadoc which is not available on the Internet (at least as known by Google and by the links on the JanusGraph web site.

#20 - 05/09/2017 04:22 PM - Constantin Asofiei

From Greg:

> I need help with the expansion and rework of the core graph itself. The biggest areas are:
>
> add all entry points as full vertices in the database, with enough information to associate them back to the original AST nodes)

I want to make sure we are on the same page here: by entry points you mean the external programs?

> add any missing call sites (at a minimum, all the internal call-sites are not there) as vertices in the database, with enough information to associate them back to the original AST nodes)

Yes, internal call sites (like function calls, internal procedure calls) are not in the current implementation. This will be tricky, with super-procs involved: a call-site can target multiple sites, depending on the runtime; will we use hint-level disambiguation? With the ADM2 code in mind, I'm not sure how easy this will be.

> create the appropriate edges to link all the vertices, including linkages back to the external proc itself
> I'm planning a different approach to the graph. The idea is to keep the edges more simple and use a generic edge labels like "contains", "includes", "calls" to define relationships. I don't want to duplicate state in edges and vertices.

OK, I think I understand this. contains will be the mapping of the AST tree, right? In this case, at some point we will need to provide an order of the children (via some index).

Something else to keep in mind: if we want to overlay the call-graph over the AST graph (the includes and contains relations), there will be vertices to target external dependencies (native libraries, ocx, sockets, etc).

> add all entry points as full vertices in the database, with enough information to associate them back to the original AST nodes)

I want to make sure we are on the same page here: by entry points you mean the external programs?

No. I mean each possible call target, whether it is used internally or externally. This would include user-defined functions, internal procedures, triggers and so forth.

The v2 implementation only had a vertex for the external procedure itself. And that vertex was shared between the actual external procedure (the entry-point) and the file containing the procedure (now called the procedure file).

> This will be tricky, with super-procs involved: a call-site can target multiple sites, depending on the runtime; will we use hint-level disambiguation?

Yes.

> With the ADM2 code in mind, I'm not sure how easy this will be.

I understand. I've already integrated the previously customer-specific ADM/ADM2 rules into the code as a standard feature. That should be expanded to handle any other cases which we can determine in advance/through automation.

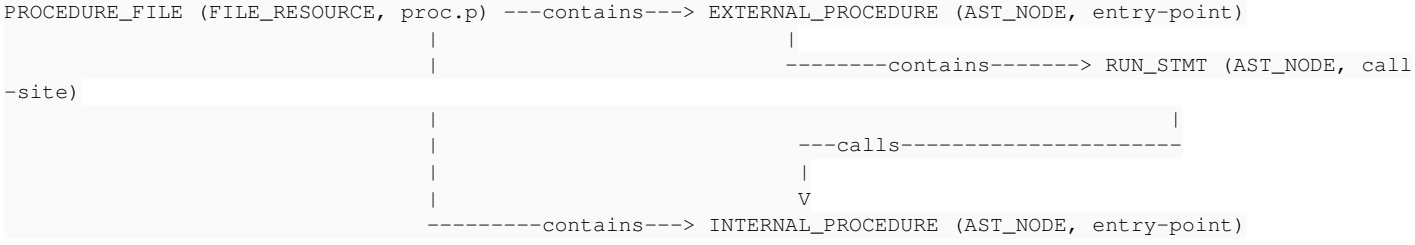
Hints are a last resort here.

> OK, I think I understand this. contains will be the mapping of the AST tree, right? In this case, at some point we will need to provide an order of the children (via some index).

Not exactly. The idea here is that we will be maintaining multiple "domains". There will be the following (so far):

FILE_RESOURCE (PROCEDURE_FILE, INCLUDE_FILE...)
AST_NODE (any vertex that maps directly to an AST node)
IPC_RESOURCE (external resource that is contacted via interprocess communications like a web service)
NATIVE_PROGRAM_RESOURCE (an API in a shared library, a child process launched via the shell...)
UNKNOWN_RESOURCE (ambiguous...)

These domains will allow us to view portions of the call graph and/or to aggregate things in useful ways. For example, every vertex in the AST_NODE domain must by definition have an inbound edge with a "contains" label that leads back to a PROCEDURE_FILE vertex (which is in the FILE_RESOURCE domain) or to an entry-point vertex that has an inbound edge with a "contains" label that leads back to a PROCEDURE_FILE vertex. We can "expand" and "collapse" all the "contained" vertices for each PROCEDURE_FILE by following these "contains" edges.



At this point I've been planning to ignore the actual AST parent/child relationships. I was also planning to avoid putting all AST nodes into the graph database, though Eric would prefer if we did put them there). I'm just trying to keep the scope manageable for now.

> Something else to keep in mind: if we want to overlay the call-graph over the AST graph (the includes and contains relations), there will be vertices to target external dependencies (native libraries, ocx, sockets, etc).

Yes, there will be non-AST nodes in the IPC_RESOURCE and NATIVE_PROGRAM_RESOURCE domains. And there will be AST_NODE vertices that have outboud edges with a "calls" label that has the inbound vertex as the target IPC_RESOURCE and NATIVE_PROGRAM_RESOURCE node.

I'm trying to make the structure of the call graph more obvious/explicit and easier to display visually.

#22 - 05/09/2017 05:56 PM - Greg Shah

I've just checked in branch 1514a revision 11161 which is a shift to JanusGraph 0.2.0 and Tinkerpop 3.2.2 plus all the rewriting of code to compile as a replacement of Titan 0.4.2 aand Tinkerpop 2.4. This has not yet been tested and it is most likely broken, but it doesn't break the build. This is also the start of the domain classification system for resources as well as the start of adding new entry points and a more explicit call graph structure.

Some of the documentation relating to the old Titan version is missing. In addition, the API (both Titan 1.0.0, JanusGraph 0.1.0 and the v3+ Tinkerpop) is very different. There are many functional improvements and it doesn't make sense to try to limp by with the old Titan 0.4.2. For this reason, I did the upgrade. I did not move forward with the OrientDB approach since there were some negative reports on line. We can revisit that later. BTW, Titan 1.0.0 and JanusGraph 0.1.0 are very, very close to the same project. So even moving to the later Titan (dead code), would have been the same amount of work.

The new graph traversal approach (based on Gremlin) is present and is a major improvement over the Tinkerpop v2.4 approach.

#23 - 05/11/2017 06:07 AM - Constantin Asofiei

About the index issues: this code in JanusGraphDB.createIndexImpl needs to be changed, as composite indexes don't allow parameters...

```
         if (String.class.equals(datatype))
         {
            bld.addKey(prop, Mapping.STRING.asParameter());
         }
         else
         {
            bld.addKey(prop);
         }
```

If we add just bld.addKey(prop);, there are no more warnings that 'an index is not used'.

#24 - 05/11/2017 06:15 AM - Greg Shah

Constantin, please rework and extend the run_statements.rules, for each call-site:

- establish a "contains" edge from the containing external_procedure or internal_procedure vertex (see the TODO on line 181 of load_code_set.rules)

- change the linkage to the target vertex to be a "calls" edge
- eliminate the dependence on storing state in the edge that describes the nodes on both sides, I want derive the state from the structure of the graph unless there is a really good functional or technical reason to do otherwise
- where the call-site references an external procedure it needs to link to the new external_procedure node and not to the procedure_file node
- add internal_procedure support, including handling super procedures
- add run_super
- rework the native_api, port_type and other non-procedure resources support to properly link, add_external_node now creates a single unique "supernode" for these; for example native_api in load_code_set.rules line 166

I may have forgotten or overlooked something. Generally, please get all RUN statements moved to the new call graph approach.

#25 - 05/11/2017 06:16 AM - Greg Shah

> About the index issues: this code in JanusGraphDB.createIndexImpl needs to be changed, as composite indexes don't allow parameters...

Yes, rev 11166 fixed this. Sorry, I just checked it in this morning.

#26 - 05/11/2017 06:33 AM - Constantin Asofiei

Greg Shah wrote:

> > About the index issues: this code in JanusGraphDB.createIndexImpl needs to be changed, as composite indexes don't allow parameters...
>
> Yes, rev 11166 fixed this. Sorry, I just checked it in this morning.

It doesn't work, I get a lucene exception.

Do we really need in Janus to map the key as String? Because if instead of bld.addKey(prop, Mapping.STRING.asParameter()); I do bld.addKey(prop);, I get no complain that the prefix search doesn't use an index, with lucene or not.

#27 - 05/11/2017 07:34 AM - Greg Shah

> Do we really need in Janus to map the key as String? Because if instead of bld.addKey(prop, Mapping.STRING.asParameter());

The prefix matching that is done to match the reversed filename supposedly needs this feature. I have not tested it to prove it and the documentation has not always been fully accurate.

I am looking at this next error now.

#28 - 05/12/2017 05:45 PM - Constantin Asofiei

Just a heads up: the Element.property(String) getter has changed, it returns a Property instance - I'll add a cgw.property(String) getter equivalent.

#29 - 05/15/2017 08:37 AM - Constantin Asofiei

Greg, a quick question: the callgraph needs to be built on the F2's ASTs, correct? And not the full conversion's modified ASTs.

#30 - 05/15/2017 10:47 AM - Eric Faulhaber

Constantin Asofiei wrote:

> Greg, a quick question: the callgraph needs to be built on the F2's ASTs, correct? And not the full conversion's modified ASTs.

Yes. If there is some information/annotation we need for the call graph that is not determined until later stages, we can consider moving it into the front end, if that is feasible. Do you have something specific in mind?

#31 - 05/15/2017 11:08 AM - Constantin Asofiei

Eric Faulhaber wrote:

> Constantin Asofiei wrote:
>
> > Greg, a quick question: the callgraph needs to be built on the F2's ASTs, correct? And not the full conversion's modified ASTs.
>
> Yes. If there is some information/annotation we need for the call graph that is not determined until later stages, we can consider moving it into the front end, if that is feasible. Do you have something specific in mind?

No, just wanted to make sure we are on the same page.

A secondary issue: for defined procedures or functions, which are 'not local' (i.e. IN SUPER, IN handle or EXTERNAL), I want to do the graph like this:

1. a REFERENCES edge between the local definition and the remote one (i.e. "references" link from the local internal procedure to the native API

node). For the IN SUPER and IN handle, there are two ways to do this:
- create ~~calls~~ references links to all internal procs/funcs with the same name
- use hints (i.e. all external programs with actual runtime definition possible being called) and link only to those definitions

2. when a RUN statement or function call targets a local definition, it will link to it. The report can later see that the local definition "references" this external one.

The idea is that I don't want to miss from the graph that an external procedure is defining some internal procedure/function which is non-local.

Also, the RUN statement can't disambiguate at conversion time between internal procedure and external procedure calls: runtime depends on which one is found first, following the disambiguation rules:

1. local definition
2. definition in this-procedure:super-procedures'
3. definition in session:super-procedures
4. external program

If there is no local definition, the target will be either a super definition or an external program. I can either use hints to disambiguate and/or link to all names (external program or internal proc) which match the target...

#32 - 05/15/2017 11:09 AM - Constantin Asofiei

Constantin Asofiei wrote:

> - create calls links to all internal procs/funcs with the same name

This is references, not calls.

#33 - 05/15/2017 04:55 PM - Constantin Asofiei

Greg Shah wrote:

> Constantin, please rework and extend the run_statements.rules, for each call-site:
>
> - establish a "contains" edge from the containing external_procedure or internal_procedure vertex (see the TODO on line 181 of load_code_set.rules)

Done - see CGW.createAstNode; this is generic code.

- change the linkage to the target vertex to be a "calls" edge

Done. See link_to_target - this is generic code. Are you planning to use create_entry_point?

- eliminate the dependence on storing state in the edge that describes the nodes on both sides, I want derive the state from the structure of the graph unless there is a really good functional or technical reason to do otherwise

There is some redundant information in create_edge:

```
        <rule>cgw.property(edge, "call-site-id",  callSite.getId())</rule>
        <rule>cgw.property(edge, "call-site-key", callSiteKey)</rule>
        <rule>cgw.property(edge, "line",          callSite.getLine())</rule>
        <rule>cgw.property(edge, "column",        callSite.getColumn())</rule>
```

which is used by current reports, but we can derive them, if they are removed.

- where the call-site references an external procedure it needs to link to the new external_procedure node and not to the procedure_file node

Done - see link_to_target.

- add internal_procedure support, including handling super procedures

WIP. I want to include function calls (including dynamic-function), too, they will be similar (as approach).

- add run_super

WIP (will add super() too)

- rework the native_api, port_type and other non-procedure resources support to properly link, add_external_node now creates a single unique "supernode" for these; for example native_api in load_code_set.rules line 166

This works OK: if mylib.dll defines a native-proc-api, which is referred via an external procedure from proc1.p and proc2.p, the two procedure native-proc-api external mylib.dll associated vertices will:

1. create a references edge to the proc.native_api represented by native-proc-api (which is linked via 'contains' from its shared library, mylib.dll)
2. any RUN stmt will create a calls edge to the vertex represented by the internal proc native-proc-api.

#34 - 05/16/2017 02:04 PM - Greg Shah

> A secondary issue: for defined procedures or functions, which are 'not local' (i.e. IN SUPER, IN handle or EXTERNAL), I want to do the graph like this:
> ...
> The idea is that I don't want to miss from the graph that an external procedure is defining some internal procedure/function which is non-local.

OK.

I had initially been of "two minds" on this. I can see your point and am OK with the approach. It makes the exploration and reporting slightly more complicated, but it also is more representative of the ABL code itself.

> If there is no local definition, the target will be either a super definition or an external program. I can either use hints to disambiguate and/or link to all names (external program or internal proc) which match the target...

How can it be a super definition? A super definition is only an internal procedure. Doesn't every internal procedure need a local definition, even if it is IN SUPER? As such we should be able to disambiguate the super cases. The rest are external procedures. Both cases can be dealt with using hints or the link-to-all-possible approach.

#35 - 05/16/2017 02:14 PM - Constantin Asofiei

Greg Shah wrote:

> > If there is no local definition, the target will be either a super definition or an external program. I can either use hints to disambiguate and/or link to all names (external program or internal proc) which match the target...
>
> How can it be a super definition? A super definition is only an internal procedure. Doesn't every internal procedure need a local definition, even if it is IN SUPER? As such we should be able to disambiguate the super cases. The rest are external procedures. Both cases can be dealt with using hints or the link-to-all-possible approach.

The runtime resolution of the target of a RUN statement is done as I specified above; an internal procedure doesn't have to be defined in the current external program, for a RUN statement to find it.

OTOH, PROCEDURE ... IN SUPER is a kind of internal procedure definition where its body resides in another program, determined by the runtime. If you are referring to this case, then yes, in our graph we can add REFERENCES links to the possible real targets (I want to do this via hints).

The same REFERENCES edge will apply for functions defined IN SUPER or IN handle, where their real 'targets' will be resolved via hints.

#36 - 05/16/2017 02:31 PM - Greg Shah

> There is some redundant information in create_edge:
> ...
> which is used by current reports, but we can derive them, if they are removed.

Yes, this redundant info should be removed.

> WIP. I want to include function calls (including dynamic-function), too, they will be similar (as approach).
> WIP (will add super() too)

OK, good.

> This works OK: if mylib.dll defines a native-proc-api, which is referred via an external procedure from proc1.p and proc2.p, the two procedure native-proc-api external mylib.dll associated vertices will:

OK.

#37 - 05/17/2017 09:42 AM - Constantin Asofiei

Greg, a quick status:

1. func/proc calls, super and dynamic calls are added (including 'references' via hints for 'virtual' proc/funcs, with IN SUPER and IN handle options at their definition)
2. fixed cgw.addTable and load_schema_triggers - AST nodes are created as SCHEMA_FILE -contains-> DATABASE -contains-> TABLE -contains-> KW_TAB_TRG -calls-> ext-prog

I assume we don't want to add to the graph the entire schema, correct?

My next step are:

1. check for bugs
2. remove the redundant edge properties
3. check the other callgraph code (native processes, remote calls, etc)

Also, should I fix the existing reports to work on the new graph structure? A lot of logic relies on the redundant info kept at the edge (I think this can be reworked for all cases via some TRPL functions), but there might be other changes required in the existing TRPL callgraph reports.

#38 - 05/17/2017 11:07 AM - Greg Shah

> I assume we don't want to add to the graph the entire schema, correct?

Yes.

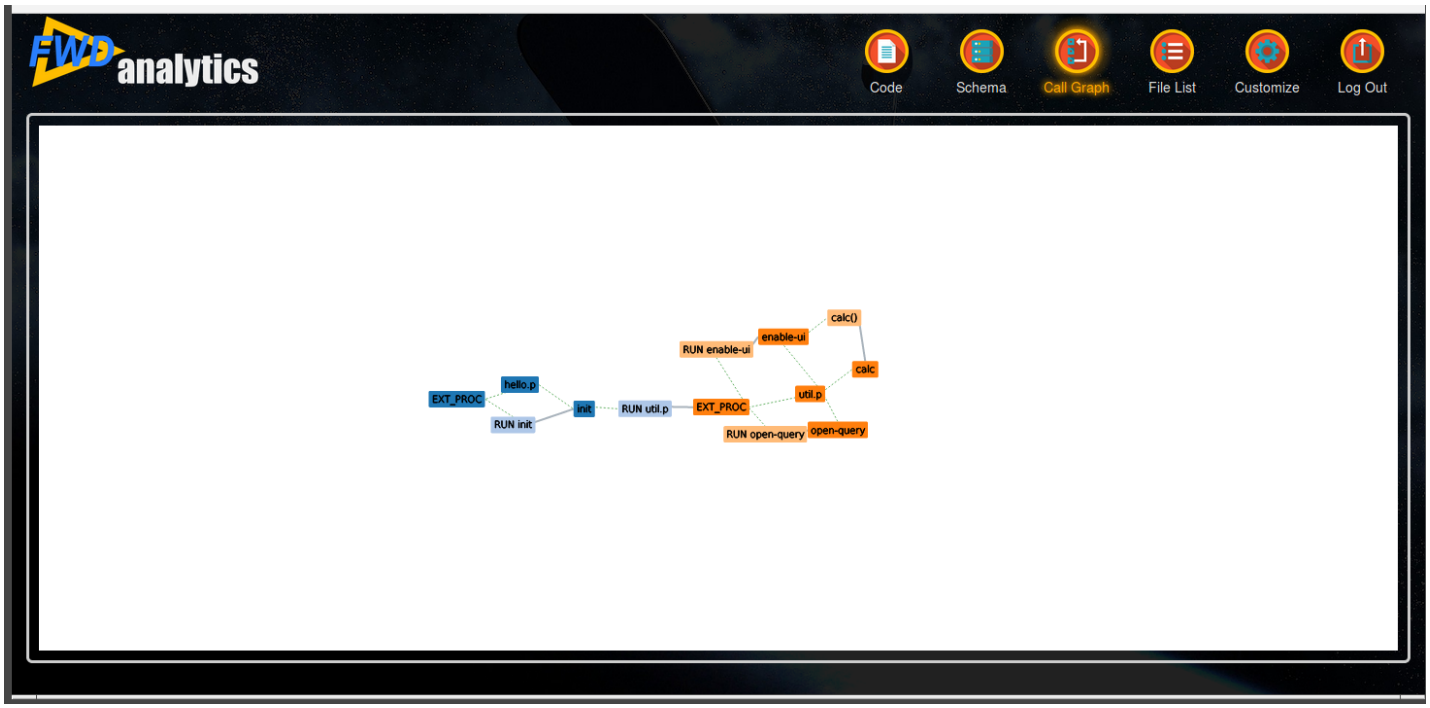
> Also, should I fix the existing reports to work on the new graph structure? A lot of logic relies on the redundant info kept at the edge (I think this can be reworked for all cases via some TRPL functions), but there might be other changes required in the existing TRPL callgraph reports.

No, let's wait on this. The reporting will be displayed in the web app, via JS upcalls to a callgraph API. The old TRPL code won't be used.

Please work through the other entry points and call-sites from #2251-7 which are not yet supported (or not yet fixed to work in the new model).

#39 - 05/22/2017 08:17 AM - Greg Shah

As of branch 1514a rev 11188, there is a good starting point for the call graph visualization. I have implemented the "force layout graph" using d3, with a start on some of the more advanced features.



Features that are already implemented:

- nodes appear as a rounded rectangle with text inside, the rectangle's size is calculated based on the text bounding box
- contains links are dashed green lines, calls links are solid slategray
- mouse wheel can be used to zoom in and out

- the svg has a white background and is sized with a 20px margin in its div container
- dragging on the background allows the graph to be moved
- dragging on a specific node allows moving it (through it will not remain fixed at the drop location)
- nodes and links have tooltips
- the link target point is calculated as the center of the side of the rectangle that is intersected

Features still to come:

- source link point needs to be moved to the intersecting side of the source node
- an arrow head marker needs to be added to the calls link target side
- add weighting to the link strength for the procedure_file node to its "contains" nodes, see if this leaves that node near the center of the cluster for that file
- fine tune the force values and/or add a custom collision prevention force to ensure the graph nodes don't overlap
- implement a fixed location at the drop point and a way to clear the fixed position (possibly with a double click)
- fine tune the look and feel, consider these:
  - map the colors into more specific roles so that a legend can be made and there is some consistency in color meaning
  - consider improvements to the link and node stroking
  - consider using different shapes for nodes (this is tricky... but could be useful)
  - consider containing all nodes within a given file within a single large rectangle that might be show with a dashed stroke and a label (tricky to do)
- add highlighting so that a click on a node will dim all of the graph except for the node and it's links and connected nodes at 1 level of separation
- add a legend
- add details to the tooltips so that a more complete rendering of the AST node state is shown
- we might want to display these same details (as a text area on the side of the screen) when a node is highlighted
- implement support for proper resize event handling
- make the call graph reload properly when it is already loaded and you click on the call graph menu item

That will comprise the primary visual work. From there I need to work on features for filtering/searching/traversal and linking to the AST source view.

How to try this out:

1. Checkout and build 1514a.
2. Checkout Hotel GUI and link in 1514a.
3. Run "ant rpt".
4. cd deploy/server/ && ./report.sh
5. Point browser to https://localhost:8443.
6. Type any text into the userid and password boxes and then press enter.
7. Wait for the code reports to load.
8. Click on the Call Graph menu item.

#40 - 05/22/2017 08:17 AM - Greg Shah

- File call_graph_visualization_branch_1514a_rev_11188_20170522.png added

#41 - 05/22/2017 10:13 AM - Constantin Asofiei

This is a summary/grouping of #2251-7
What is marked with an X means is implemented in current 1514a branch.

Procedure and Function Calls

Notes:

- 16. appserver exports - currently there is no tracking of a i.e. RUN ... IN handle which targets a internal procedure from another application via a AppServer connection. This can be disambiguated via hints, but the target external program will be marked as **missing**.

Call sites:

X 1. run filename - an external procedure defined in the specified source file
X 2. run filename ON server_handle - the target session is found at runtime via the given handle which can be:
- the local session handle
- a remote appserver handle
X 3. run internal_proc - procedure stmt defined in the local source file
X 4. run internal_proc - procedure stmt defined in super
X 5. run internal_proc IN proc_handle - target is found at runtime via the given handle which can be:
- the local file
- a procedure that is on the stack
- a local persistent procedure
- a remote persistent procedure
X 6. run internal_proc - procedure stmt defined as API call to Windows DLL or UNIX shared library
X 7. run value(expr) - an internal or external procedure found by evaluating the expression and then matching the name to one of the cases above
X 8. run value(expr) ON server_handle - same as case 2 but with a runtime-generated filename
X 9. run value(expr) IN proc_handle - same as case 5 but with a runtime-generated procedure name
X 10. run library_ref - runs a specific external procedure that is packaged in a procedure library (.pl file)
X 11. run super - runs the immediate super procedure for the currently executing internal procedure
X 12. function call - function stmt defined in local file
X 13. function call - function stmt defined IN SUPER
X 14. function call - function stmt defined IN proc_handle with the target file found at runtime via the given handle:
- the local file
- a procedure that is on the stack
- a local persistent procedure
- a remote persistent procedure
X 15. super() - calls the user-defined function that exists in the immediate super procedure for the currently executing local user-defined function
X 16. dynamic-function(funcname_expr) - calls the user-defined function in the local file that is determined at runtime by evaluating the funcname_expr, based on the associated function stmt, this may be any of the following:
- function in local file
- function defined IN SUPER
- function defined IN proc_handle
X 17. dynamic-function(funcname_expr IN proc_handle) - calls the user-defined defined in the procedure specified by the given handle, which is determined at runtime to be one of:
- the local file
- a procedure that is on the stack
- a local persistent procedure
- a remote persistent procedure
20. use of PUBLISH to invoke internal procedures that are registered with SUBSCRIBE
X 34. ON statement invocation of a procedure via PERSISTENT RUN

Additional cases:

- procedure targeted by a read-response event.

Entry Points:

X 1. external procedure - each 4GL source file
X 2. internal procedure - procedure stmt
X 3. user-defined functions
X 6. TRIGGER-PROCEDURE - schema-level database triggers
11. internal procedures registered via SUBSCRIBE statement
X 13. Asynchronous Request Procedure PROCEDURE/KW_PROC/KW_ASYNC/KW_EVT_PROC
16. appserver exports (internal procedure, external procedure or user-defined function)

Other Misc

Call sites:

X 18. shell-based command execution of a child process (e.g. unix or os-command)
X 19. stream I/O to/from a child process (input thru, output thru and input-output thru)
21. invocation of a statically linked user C function via CALL
X 22. usage of DDE EXECUTE (command execution)
X 23. COM automation object method calls
24. OCX control method calls **need to double-check**
25. .NET method calls/constructor (property access?)
31. CALL handle object
37. web services invocation
38. SOAP invocation
39. stored procedure invocation

Entry Points:
12. OCX Event Procedures
14. DDE-NOTIFY trigger
15. exported web services

OO 4GL

Call sites:

X 26. OO 4GL method calls
X 27. OO 4GL property getters/setters (hidden dispatch)
X 28. OO 4GL constructors/destructors (hidden dispatch)
29. class events pub/sub
X 30. dynamic-invoke and dynamic-new

Entry Points:
X 7. OO 4GL user-defined methods
X 8. OO 4GL user-defined properties (getters/setters)
X 9. OO 4GL constructor
X 10. OO 4GL destructor

Others:
X - SUPER c'tor, THIS-OBJECT c'tor and SUPER:method
- load the .NET classes as external dependencies, and not 4GL code.

Event processing

Call Sites:

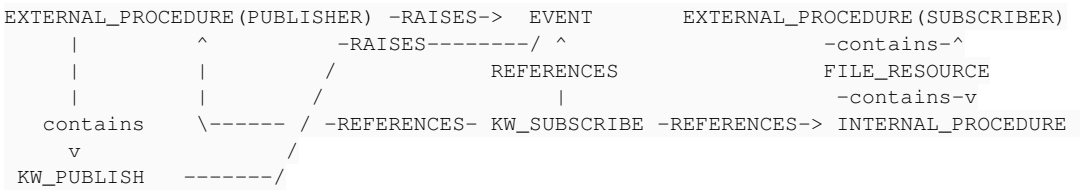
32. forced invocation of a widget trigger via the APPLY language stmt (which can invoke arbitrary code that is on the call stack)
33. forced invocation of a procedure trigger via the APPLY language stmt
35. input blocking stmts (events may be generated during these):
- choose
- insert
- message (only with VIEW-AS ALERT-BOX, UPDATE or SET)
- prompt-for
- set
- update
- readkey (not when reading from file)
- wait-for
- pause
36. process-events will cause events to be processed, in combination with apply, enable or direct setting of the sensitive attribute it is possible this could be a vector for triggers to be called

Entry Points:

4. ON stmt - trigger definitions, includes UI, procedure and database event processing, these can be defined as an in-line block of code or as a PERSISTENT RUN of a procedure name (this later form is really a call site, see above)
5. KW_TRIGGERS - trigger phrase on widgets, including dynamically created widgets

#42 - 05/22/2017 12:48 PM - Constantin Asofiei

Greg, about PUBLISH/SUBSCRIBE. The events are raised by the "external procedure", and not by internal procedure or function (for example, PUBLISH can be executed from a function, but this doesn't mean the function publishes the event - only the external procedure does this). So, I propose a vertex/edge structure like this:



Why there are two RAISES edges? One is from the subscriber (which expects an EVENT to be published by that external procedure) and one is from the actual PUBLISH statement, where the event is raised. If the SUBSCRIBE is not ANYWHERE, then it will have a REFERENCES to the external procedure handle which is expected to publish the event. This allows us to find procedures which don't publish any event.

At the PUBLISH statement, there is no explicit program name to be invoked - this is always registered by the SUBSCRIBE. I don't want to hard-code via hints the internal procedures which would be reached by PUBLISH, unless you want me to do so. The targets of a PUBLISH statement can be computed by looking at the EVENT and check to which procedures the SUBSCRIBE statements (referencing this event) are linked.

Also, the PUBLISH node will link to the (event, ext-proc), where the ext-proc might not be the same one which contains the KW_PUBLISH.

#43 - 05/22/2017 01:10 PM - Constantin Asofiei

Greg, any idea if the testcases/uast/oo tests will parse in FWD? Because I'm getting some weird parsing errors...

#44 - 05/22/2017 02:02 PM - Greg Shah

See OO Skeleton Classes

#45 - 05/22/2017 02:11 PM - Greg Shah

> Why there are two RAISES edges? One is from the subscriber (which expects an EVENT to be published by that external procedure) and one is from the actual PUBLISH statement, where the event is raised.

The RAISES don't seem to be connected to the subscriber, so I am confused on that point.

I don't understand the need for 2 RAISES edges. The fact that the EXTERNAL_PROCEDURE is the context for the EVENT should be established in some other way. The PUBLISH statement should be the only RAISES edge.

> If the SUBSCRIBE is not ANYWHERE, then it will have a REFERENCES to the external procedure handle which is expected to publish the event. This allows us to find procedures which don't publish any event.

I think having 3 REFERENCES edges is confusing. If we really want to model all of these relationships, then the edges should have different labels to allow us to differentiate them later.

> At the PUBLISH statement, there is no explicit program name to be invoked - this is always registered by the SUBSCRIBE. I don't want to hard-code via hints the internal procedures which would be reached by PUBLISH, unless you want me to do so. The targets of a PUBLISH statement can be computed by looking at the EVENT and check to which procedures the SUBSCRIBE statements (referencing this event) are linked.

I agree that PUBLISH should link to the EVENT node. I agree that the EVENT node (by default) can link to all places where that same event is SUBSCRIBEd. I think we still need to allow hints to so that one can optionally limit those edges to a smaller list otherwise it might pull in much more code than is actually possible at runtime.

#46 - 05/22/2017 03:00 PM - Constantin Asofiei

Greg Shah wrote:

> See OO Skeleton Classes

But aren't these supposed to be loaded into SymbolResolver$WorkArea.classDict? When parsing, it fails to load Progress.Lang.Object, as it doesn't exist in classDict (I supposed it should be in the global scope?). The skeleton classes are loaded in SymbolResolver$WorkArea.oo4glCls, but just by name, not parsed.

#47 - 05/22/2017 03:06 PM - Greg Shah

> But aren't these supposed to be loaded into SymbolResolver$WorkArea.classDict

The actual skeleton classes need to be available in the file system as they get parsed at the same time as the application classes. The parser does not have any knowledge of the internals of those 4GL and .NET classes.

#48 - 05/22/2017 03:12 PM - Constantin Asofiei

Greg Shah wrote:

> But aren't these supposed to be loaded into SymbolResolver$WorkArea.classDict

> The actual skeleton classes need to be available in the file system as they get parsed at the same time as the application classes. The parser does not have any knowledge of the internals of those 4GL and .NET classes.

The skeleton classes are there. The exception I get is this:

```
./oo/oo-test-driver.p
com.goldencode.ast.AstException: Error processing ./oo/Stateful.cls
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:992)
    at com.goldencode.p2j.uast.SymbolResolver.loadClass(SymbolResolver.java:2055)
    at com.goldencode.p2j.uast.ProgressParser.user_defined_type_name(ProgressParser.java:7296)
    at com.goldencode.p2j.uast.ProgressParser.var_type(ProgressParser.java:45828)
    at com.goldencode.p2j.uast.ProgressParser.as_clause(ProgressParser.java:19968)
    at com.goldencode.p2j.uast.ProgressParser.def_var_stmt(ProgressParser.java:10596)
    at com.goldencode.p2j.uast.ProgressParser.define_stmt(ProgressParser.java:9149)
    at com.goldencode.p2j.uast.ProgressParser.stmt_list(ProgressParser.java:22927)
    at com.goldencode.p2j.uast.ProgressParser.statement(ProgressParser.java:6201)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5074)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.external_proc(ProgressParser.java:4756)
    at com.goldencode.p2j.uast.AstGenerator.parse(AstGenerator.java:1488)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:987)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:859)
    at com.goldencode.p2j.uast.ScanDriver.lambda$0(ScanDriver.java:373)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:408)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:247)
    at com.goldencode.p2j.convert.ConversionDriver.runScanDriver(ConversionDriver.java:480)
    at com.goldencode.p2j.convert.ConversionDriver.front(ConversionDriver.java:361)
    at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1933)
Caused by: java.lang.NullPointerException
    at com.goldencode.p2j.uast.SymbolResolver.annotateClassRef(SymbolResolver.java:2154)
    at com.goldencode.p2j.uast.ProgressParser.lvalue(ProgressParser.java:14289)
    at com.goldencode.p2j.uast.ProgressParser.primary_expr(ProgressParser.java:52783)
    at com.goldencode.p2j.uast.ProgressParser.chained_object_members(ProgressParser.java:18997)
    at com.goldencode.p2j.uast.ProgressParser.un_type(ProgressParser.java:52493)
    at com.goldencode.p2j.uast.ProgressParser.prod_expr(ProgressParser.java:52376)
    at com.goldencode.p2j.uast.ProgressParser.sum_expr(ProgressParser.java:36931)
    at com.goldencode.p2j.uast.ProgressParser.compare_expr(ProgressParser.java:51983)
    at com.goldencode.p2j.uast.ProgressParser.log_not_expr(ProgressParser.java:51854)
    at com.goldencode.p2j.uast.ProgressParser.log_and_expr(ProgressParser.java:51795)
    at com.goldencode.p2j.uast.ProgressParser.expr(ProgressParser.java:8644)
    at com.goldencode.p2j.uast.ProgressParser.return_core(ProgressParser.java:46346)
    at com.goldencode.p2j.uast.ProgressParser.return_stmt(ProgressParser.java:30527)
    at com.goldencode.p2j.uast.ProgressParser.stmt_list(ProgressParser.java:23141)
    at com.goldencode.p2j.uast.ProgressParser.statement(ProgressParser.java:6201)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5074)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.user_defined_method(ProgressParser.java:5707)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5058)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.class_def(ProgressParser.java:5319)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5042)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.external_proc(ProgressParser.java:4756)
    at com.goldencode.p2j.uast.AstGenerator.parse(AstGenerator.java:1488)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:987)
    ... 20 more
```

caused when parsing super:ToString() from Stateful.cls:

```
  method public override character ToString ():
```

```
      return super:ToString() + " (valid = " + string(valid) + ")".
   end method.
```

The skeleton folder is placed in the same folder as the FWD cfg/ folder.

#49 - 05/22/2017 04:02 PM - Greg Shah

From the project root, you should have a p2j/skeleton/ directory.

#50 - 05/22/2017 05:11 PM - Constantin Asofiei

Greg Shah wrote:

> From the project root, you should have a p2j/skeleton/ directory.

Tried your idea, also I've set the oo-skeleton-path to skeleton/ and it doesn't work... same NPE.

#51 - 05/22/2017 05:20 PM - Greg Shah

Check out the latest cfg/class_map.xml. See if that makes a difference.

#52 - 05/22/2017 05:27 PM - Constantin Asofiei

Greg Shah wrote:

> Check out the latest cfg/class_map.xml. See if that makes a difference.

No, it doesn't work. Looks like the implicit inherits Progress.Lang.Object is not treated by FWD. If I add it explicitly to the Stateful.cls, then it gets parsed completely.

I also get other parse errors in the oo/ code, but I think I can work around them.

#53 - 05/22/2017 07:15 PM - Greg Shah

Something is definitely wrong there. All of these parsed the last time I tried it. The implicit parent class was working too.

I see now that the p2j.cfg.xml had this:

```
<parameter name="oo-skeleton-path" value="./skeleton/" />
```

So putting the skeleton/ directory in the project root was correct. Please post your skeleton/ directory structure. It should have an skeleton/oo4gl/ directory where the Progress/... sub-directory tree exists.

In SymbolResolver, you can set breakpoints on access to ROOT_OBJ_NAME to see how the Progress.Lang.Object inheritance works.

#54 - 05/23/2017 07:40 AM - Constantin Asofiei

Greg Shah wrote:

> In SymbolResolver, you can set breakpoints on access to ROOT_OBJ_NAME to see how the Progress.Lang.Object inheritance works.

The problem is fixed with this patch:

```
### Eclipse Workspace Patch 1.0
#P p2j
Index: src/com/goldencode/p2j/uast/SymbolResolver.java
===================================================================
--- src/com/goldencode/p2j/uast/SymbolResolver.java    (revision 1054)
+++ src/com/goldencode/p2j/uast/SymbolResolver.java    (working copy)
@@ -1772,9 +1772,10 @@
             throw new RuntimeException(err);
          }
       }
-      else
+      else if (!ROOT_OBJ_NAME.equals(name))
       {
          // default parent for all classes
+         loadClass(ROOT_OBJ_NAME);
          parent = lookupClass(ROOT_OBJ_NAME);
       }
```

The reason: the parent class was just looked up, and not loaded...

There are other errors related to chaining and static method references; I'm postponing these.

```
./oo/oo-test-driver.p
line 14:10: unexpected token: get-printer
    at com.goldencode.p2j.uast.ProgressParser.downstream_chained_reference(ProgressParser.java:52961)
    at com.goldencode.p2j.uast.ProgressParser.chained_object_members(ProgressParser.java:19026)
    at com.goldencode.p2j.uast.ProgressParser.assignment(ProgressParser.java:6397)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5078)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.inner_block(ProgressParser.java:6009)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5066)
    at com.goldencode.p2j.uast.ProgressParser.then_clause(ProgressParser.java:34273)
    at com.goldencode.p2j.uast.ProgressParser.if_stmt(ProgressParser.java:28496)
    at com.goldencode.p2j.uast.ProgressParser.stmt_list(ProgressParser.java:23008)
    at com.goldencode.p2j.uast.ProgressParser.statement(ProgressParser.java:6201)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5074)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.external_proc(ProgressParser.java:4756)
    at com.goldencode.p2j.uast.AstGenerator.parse(AstGenerator.java:1488)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:987)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:859)
    at com.goldencode.p2j.uast.ScanDriver.lambda$0(ScanDriver.java:373)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:408)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:247)
    at com.goldencode.p2j.convert.ConversionDriver.runScanDriver(ConversionDriver.java:480)
    at com.goldencode.p2j.convert.ConversionDriver.front(ConversionDriver.java:361)
    at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1933)
line 14:21: unexpected token: (
    at com.goldencode.p2j.uast.ProgressParser.assignment(ProgressParser.java:6410)
```

```
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5078)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.inner_block(ProgressParser.java:6009)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5066)
    at com.goldencode.p2j.uast.ProgressParser.then_clause(ProgressParser.java:34273)
    at com.goldencode.p2j.uast.ProgressParser.if_stmt(ProgressParser.java:28496)
    at com.goldencode.p2j.uast.ProgressParser.stmt_list(ProgressParser.java:23008)
    at com.goldencode.p2j.uast.ProgressParser.statement(ProgressParser.java:6201)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5074)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.external_proc(ProgressParser.java:4756)
    at com.goldencode.p2j.uast.AstGenerator.parse(AstGenerator.java:1488)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:987)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:859)
    at com.goldencode.p2j.uast.ScanDriver.lambda$0(ScanDriver.java:373)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:408)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:247)
    at com.goldencode.p2j.convert.ConversionDriver.runScanDriver(ConversionDriver.java:480)
    at com.goldencode.p2j.convert.ConversionDriver.front(ConversionDriver.java:361)
    at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1933)
line 14:24: unexpected token: render
    at com.goldencode.p2j.uast.ProgressParser.lvalue(ProgressParser.java:13265)
    at com.goldencode.p2j.uast.ProgressParser.primary_expr(ProgressParser.java:52783)
    at com.goldencode.p2j.uast.ProgressParser.chained_object_members(ProgressParser.java:18997)
    at com.goldencode.p2j.uast.ProgressParser.un_type(ProgressParser.java:52493)
    at com.goldencode.p2j.uast.ProgressParser.prod_expr(ProgressParser.java:52376)
    at com.goldencode.p2j.uast.ProgressParser.sum_expr(ProgressParser.java:36931)
    at com.goldencode.p2j.uast.ProgressParser.compare_expr(ProgressParser.java:51983)
    at com.goldencode.p2j.uast.ProgressParser.log_not_expr(ProgressParser.java:51854)
    at com.goldencode.p2j.uast.ProgressParser.log_and_expr(ProgressParser.java:51795)
    at com.goldencode.p2j.uast.ProgressParser.expr(ProgressParser.java:8644)
    at com.goldencode.p2j.uast.ProgressParser.assignment(ProgressParser.java:6439)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5078)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.inner_block(ProgressParser.java:6009)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5066)
    at com.goldencode.p2j.uast.ProgressParser.then_clause(ProgressParser.java:34273)
    at com.goldencode.p2j.uast.ProgressParser.if_stmt(ProgressParser.java:28496)
    at com.goldencode.p2j.uast.ProgressParser.stmt_list(ProgressParser.java:23008)
    at com.goldencode.p2j.uast.ProgressParser.statement(ProgressParser.java:6201)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5074)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.external_proc(ProgressParser.java:4756)
    at com.goldencode.p2j.uast.AstGenerator.parse(AstGenerator.java:1488)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:987)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:859)
    at com.goldencode.p2j.uast.ScanDriver.lambda$0(ScanDriver.java:373)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:408)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:247)
    at com.goldencode.p2j.convert.ConversionDriver.runScanDriver(ConversionDriver.java:480)
    at com.goldencode.p2j.convert.ConversionDriver.front(ConversionDriver.java:361)
    at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1933)
line 14:30: unexpected token: (
    at com.goldencode.p2j.uast.ProgressParser.assignment(ProgressParser.java:6464)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5078)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.inner_block(ProgressParser.java:6009)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5066)
    at com.goldencode.p2j.uast.ProgressParser.then_clause(ProgressParser.java:34273)
    at com.goldencode.p2j.uast.ProgressParser.if_stmt(ProgressParser.java:28496)
    at com.goldencode.p2j.uast.ProgressParser.stmt_list(ProgressParser.java:23008)
    at com.goldencode.p2j.uast.ProgressParser.statement(ProgressParser.java:6201)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5074)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.external_proc(ProgressParser.java:4756)
    at com.goldencode.p2j.uast.AstGenerator.parse(AstGenerator.java:1488)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:987)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:859)
    at com.goldencode.p2j.uast.ScanDriver.lambda$0(ScanDriver.java:373)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:408)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:247)
    at com.goldencode.p2j.convert.ConversionDriver.runScanDriver(ConversionDriver.java:480)
    at com.goldencode.p2j.convert.ConversionDriver.front(ConversionDriver.java:361)
    at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1933)
./oo/resource-user.p
```

```
Failure in file './oo/resource-user.p':
com.goldencode.ast.AstException: Error processing ./oo/resource-user.p
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:992)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:859)
    at com.goldencode.p2j.uast.ScanDriver.lambda$0(ScanDriver.java:373)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:408)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:247)
    at com.goldencode.p2j.convert.ConversionDriver.runScanDriver(ConversionDriver.java:480)
    at com.goldencode.p2j.convert.ConversionDriver.front(ConversionDriver.java:361)
    at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1933)
Caused by: java.lang.NullPointerException
    at com.goldencode.p2j.uast.SymbolResolver.loadViaWalk(SymbolResolver.java:5868)
    at com.goldencode.p2j.uast.SymbolResolver.loadClass(SymbolResolver.java:2077)
    at com.goldencode.p2j.uast.ProgressParser.user_defined_type_name(ProgressParser.java:7296)
    at com.goldencode.p2j.uast.ProgressParser.var_type(ProgressParser.java:45828)
    at com.goldencode.p2j.uast.ProgressParser.as_clause(ProgressParser.java:19968)
    at com.goldencode.p2j.uast.ProgressParser.def_var_stmt(ProgressParser.java:10596)
    at com.goldencode.p2j.uast.ProgressParser.define_stmt(ProgressParser.java:9149)
    at com.goldencode.p2j.uast.ProgressParser.stmt_list(ProgressParser.java:22927)
    at com.goldencode.p2j.uast.ProgressParser.statement(ProgressParser.java:6201)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5074)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.external_proc(ProgressParser.java:4756)
    at com.goldencode.p2j.uast.AstGenerator.parse(AstGenerator.java:1488)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:987)
    ... 7 more
./oo/static-resource-user.p
line 3:1: unexpected token: UpperStaticResourceHolderFailure in file './oo/static-resource-user.p':

    at com.goldencode.p2j.uast.ProgressParser.lvalue(ProgressParser.java:13265)
    at com.goldencode.p2j.uast.ProgressParser.primary_expr(ProgressParser.java:52783)
    at com.goldencode.p2j.uast.ProgressParser.chained_object_members(ProgressParser.java:18997)
    at com.goldencode.p2j.uast.ProgressParser.assignment(ProgressParser.java:6397)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5078)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.external_proc(ProgressParser.java:4756)
    at com.goldencode.p2j.uast.AstGenerator.parse(AstGenerator.java:1488)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:987)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:859)
    at com.goldencode.p2j.uast.ScanDriver.lambda$0(ScanDriver.java:373)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:408)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:247)
    at com.goldencode.p2j.convert.ConversionDriver.runScanDriver(ConversionDriver.java:480)
    at com.goldencode.p2j.convert.ConversionDriver.front(ConversionDriver.java:361)
    at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1933)
com.goldencode.ast.AstException: Error processing ./oo/static-resource-user.p
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:992)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:859)
    at com.goldencode.p2j.uast.ScanDriver.lambda$0(ScanDriver.java:373)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:408)
    at com.goldencode.p2j.uast.ScanDriver.scan(ScanDriver.java:247)
    at com.goldencode.p2j.convert.ConversionDriver.runScanDriver(ConversionDriver.java:480)
    at com.goldencode.p2j.convert.ConversionDriver.front(ConversionDriver.java:361)
    at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1933)
Caused by: java.lang.NullPointerException
    at com.goldencode.p2j.uast.ProgressParser.isComHandleType(ProgressParser.java:3992)
    at com.goldencode.p2j.uast.ProgressParser.chained_object_members(ProgressParser.java:19011)
    at com.goldencode.p2j.uast.ProgressParser.assignment(ProgressParser.java:6397)
    at com.goldencode.p2j.uast.ProgressParser.single_block(ProgressParser.java:5078)
    at com.goldencode.p2j.uast.ProgressParser.block(ProgressParser.java:4829)
    at com.goldencode.p2j.uast.ProgressParser.external_proc(ProgressParser.java:4756)
    at com.goldencode.p2j.uast.AstGenerator.parse(AstGenerator.java:1488)
    at com.goldencode.p2j.uast.AstGenerator.processFile(AstGenerator.java:987)
    ... 7 more
```

#55 - 05/23/2017 02:11 PM - Constantin Asofiei

Greg, some questions about OO:

- when an implicit constructor or destructor is called (as there is no actual implementation), how do I link it? At least the implicit c'tor should be referenced by the call-site. Or I can just link to CLASS_DEF vertex?
- the matching for constructors and methods is done by name and parameter types (as 4GL allows method overloading at class level). If there is a poly param at the call-site, I don't know how 4GL treats it. So, the question is: if the passed argument type(s) at the call-site can't be resolved entirely, how do I link it? Link to the method which matches the most types? Hints?
- SUPER call-sites at c'tor and methods - do you want me to add them, too?
- when resolving a method referenced by a call-site, its definition is searched in the class hierarchy, up the inherits edges. But this will link using the declared type, as we can't know the actual runtime type (for overridden methods).

#56 - 05/23/2017 06:14 PM - Greg Shah

> At least the implicit c'tor should be referenced by the call-site. Or I can just link to CLASS_DEF vertex?

I think that is fine for now.

> the matching for constructors and methods is done by name and parameter types (as 4GL allows method overloading at class level). If there is a poly param at the call-site, I don't know how 4GL treats it. So, the question is: if the passed argument type(s) at the call-site can't be resolved entirely, how do I link it? Link to the method which matches the most types? Hints?

I would be surprised if a poly was possible in this case. Progress resolves all of this statically at compile time. Could you please check?

If it is possible, then hints would be the way, unless there is no ambiguity (e.g. no other method that could match).

> SUPER call-sites at c'tor and methods - do you want me to add them, too?

Yes, please.

> when resolving a method referenced by a call-site, its definition is searched in the class hierarchy, up the inherits edges. But this will link using the declared type, as we can't know the actual runtime type (for overridden methods).

How much variance is possible? As noted above, the way I remember it, Progress does static compilation of these. I recall it may be quite sensitive to this, so it may not be possible.

#57 - 05/24/2017 03:35 AM - Constantin Asofiei

Greg Shah wrote:

> At least the implicit c'tor should be referenced by the call-site. Or I can just link to CLASS_DEF vertex?

I think that is fine for now.

Sorry, I don't understand. Leave it as is (with warnings) or create an edge to CLASS_DEF?

> the matching for constructors and methods is done by name and parameter types (as 4GL allows method overloading at class level). If there is a poly param at the call-site, I don't know how 4GL treats it. So, the question is: if the passed argument type(s) at the call-site can't be resolved entirely, how do I link it? Link to the method which matches the most types? Hints?

I would be surprised if a poly was possible in this case. Progress resolves all of this statically at compile time. Could you please check?

Yes, 4GL allows POLY at compile time, but the runtime doesn't do automatic casting of the value, it expects exact (or compatible - i.e. numeric for numeric) type.

If it is possible, then hints would be the way, unless there is no ambiguity (e.g. no other method that could match).

OK, I'll look into fixing it.

> SUPER call-sites at c'tor and methods - do you want me to add them, too?

Yes, please.

Done.

> when resolving a method referenced by a call-site, its definition is searched in the class hierarchy, up the inherits edges. But this will link using the declared type, as we can't know the actual runtime type (for overridden methods).

How much variance is possible? As noted above, the way I remember it, Progress does static compilation of these. I recall it may be quite sensitive to this, so it may not be possible.

The problem here is this: if we have Active and Stateful instances, where Active inherits Stateful, and we do a:

```
act = new Active().
s = cast(act, oo.Stateful).
message s:ToString().
```

the s:ToString() will point to the Stateful implementation, and not Active - as this is the compile-time type, and we can't know the actual runtime type for s.

Otherwise, going up the inherits (i.e. checking super-classes in order of inheritance) is OK, as we will find the first class which defines this method.

And actually I forgot something and need to fix: is not enough to check just inherits, we need implements, too - as the method might exist as a prototype in an interface, and the compile-time type might not have an implementation for it (just the runtime type). The resolution will be similar to how Java resolves the method.

#58 - 05/24/2017 07:05 AM - Greg Shah

> create an edge to CLASS_DEF

This one.

> the s:ToString() will point to the Stateful implementation, and not Active - as this is the compile-time type, and we can't know the actual runtime type for s.

The 2nd parameter to cast(expr, classname) is hard coded to a specific class. Thus we can actually know that the consumer of the result is of a specific type.

The problem is in how we use that information. I'm not sure that the ECW properly handles this case for sub-expressions but right now that would only affect conversion. Are you saying that the explicit type of s is not used in the method resolution at compile time?

#59 - 05/24/2017 07:14 AM - Constantin Asofiei

Greg Shah wrote:

> > create an edge to CLASS_DEF
>
> This one.

OK

> > the s:ToString() will point to the Stateful implementation, and not Active - as this is the compile-time type, and we can't know the actual runtime type for s.
>
> The 2nd parameter to cast(expr, classname) is hard coded to a specific class. Thus we can actually know that the consumer of the result is of a specific type.

What if there are multiple cast with different super-classes? Or if is done in some procedure/function? We can't easily decide at conversion time which concrete implementation a class var will have at runtime...

> Are you saying that the explicit type of s is not used in the method resolution at compile time?

No. At compile time, the type of s is well known. But, at runtime, s might be referring any possible sub-class of its type; if the method is overridden in a sub-class, the conversion time will not be able to link to it, it will always link to the method as defined in the compile-time type of s (which might be even an interface).

Currently, I have only inherits and implements links at class-level. I think it will be good to add override edges from a sub-class method to its correspondent in a parent class or interface.

#60 - 05/24/2017 10:47 AM - Constantin Asofiei

Greg, I think everything is there for the OO call-sites (except events/triggers at class-level, which I haven't checked). See 1514 rev 11193 and 11194.

I'm focusing next on the Other Misc and what remains from the procedures section, as noted in #3277-41

#61 - 05/24/2017 12:30 PM - Greg Shah

The following is a really great resource for understanding zooming/panning and transforms in d3:

https://stackoverflow.com/questions/21344340/semantic-zooming-of-force-directed-graph-in-d3/21356139#21356139

Example which shows a technique that can be used to render different shapes for nodes in the graph:

http://stackoverflow.com/questions/18509792/d3-force-directed-graph-different-shape-according-to-data-and-value-given

This is the closest match to our initial approach:

https://github.com/nylen/d3-process-map/blob/master/script.js

Useful samples and docs explaining forces:

http://www.puzzlr.org/force-graphs-with-d3/

Good examples of some useful techniques (but it is on v3 not v4):

http://www.coppelia.io/2014/07/an-a-to-z-of-extra-features-for-the-d3-force-layout/

D3 API reference:

https://github.com/d3/d3/blob/master/API.md

#62 - 05/26/2017 06:00 PM - Constantin Asofiei

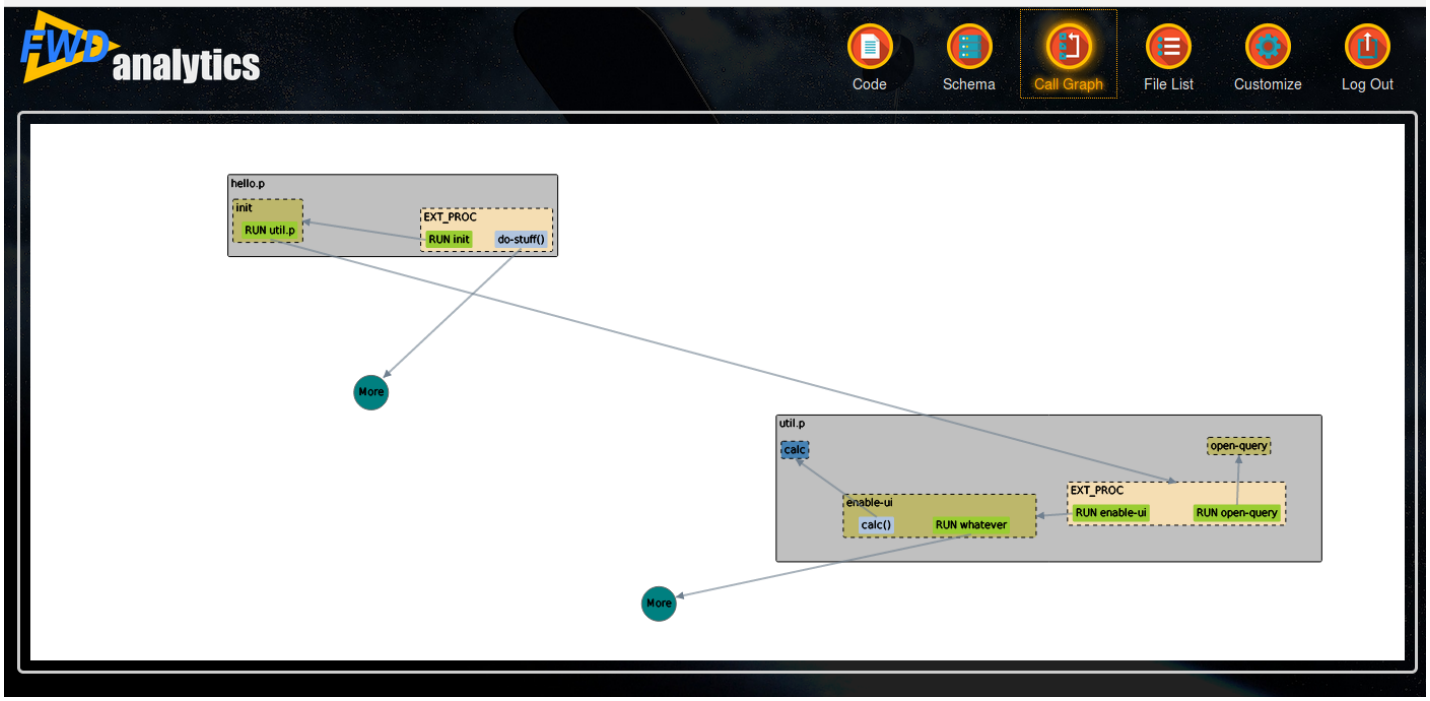
Greg, regarding events/triggers, can you help me with something: if ON ... PERSISTENT RUN is a call-site (as in #3277-41), then why is not a SUBSCRIBE a call-site, too? Because is doing something similar, register a piece of code to be executed for an event. I think we need to split the event stuff in two:

1. one graph with the event raising and consuming (i.e. call-sites for these are any event-processing stmt or PUBLISH) and consuming are the entry-points (trigger definition or SUBSCRIBE). We will have RAISES edges from the call-sites to the EVENT vertex and LISTENS from the i.e. trigger to the EVENT vertex.
2. the other graph is made of the links from the events to possible code being ran by that event, a CALLS edge from the event definition (i.e. ON stmt or SUBSCRIBE) to the code being executed - internal procedure, for a ON ... PERSISTENT RUN or SUBSCRIBE.

#63 - 05/26/2017 06:41 PM - Greg Shah

- File call_graph_visualization_branch_1514a_rev_11204_20170526.png added

Current status (rev 11204):



- link z-order is higher than nodes
- both source and target link points are properly intersecting the sides of the respective nodes
- arrow head markers added to the calls link target side and can be added to any other links
- multiple shapes (circles and rectangles for now) are supported
- all contains relationships are represented as container rectangles which bound the contained elements
- drag and drop will cause the node to stay at that fixed location until a double click on the node clears the fixed position
- created a map of visual styling by node type that controls color, stroke, stroke-dashing and so forth
- basic resizing event handling
- the call graph menu is fixed

#64 - 05/26/2017 07:10 PM - Greg Shah

> if ON ... PERSISTENT RUN is a call-site (as in #3277-41), then why is not a SUBSCRIBE a call-site, too? Because is doing something similar, register a piece of code to be executed for an event.

This is a good point. I agree with your approach.

The LISTENS should link from the "registration node" to the EVENT. This would be the ON statement (NOT the TRIGGER itself) or the SUBSCRIBE.

In the case of an in-line TRIGGER block (as part of the ON statement), there is really a code block that is a valid entry point. It would be a target of the CALLS edge from the ON statement.

How quickly can you get this (the graph itself) to a good stopping point? I will need help in the following areas:

- encoding the Hotel GUI hints needed to get a full graph
- rework the current reports, but as dynamically calculated results that we will send down to the browser for display
- help build the API to access the call graph from the client
- possibly help with the client side

I'm quite deep into the visualization. I will get further during the weekend but I still have:

- address the initial layout, positioning bias and collision avoidance (it is tricky because of the container implementation)
- build an initial API to access a subset of the graph and connect the web client to the back end, properly
- continue to tweak the look and feel so that the nodes and links are more intuitively understandable, at a minimum the full set of node and edge types need to be handled with proper shapes, colors, stroking and text
- investigate if curved links help the look
- investigate if text on links can help
- add a legend
- add details to the tooltips so that a more complete rendering of the AST node state is shown
- add a UI for the call graph reports
- search (load a portion of the graph based on specifying search criteria)
- traversal of the graph (follow "More" nodes to off screen areas)
- allow linking to the source/AST view from an AST node
- filtering (control the types of nodes/edges of the graph which are displayed based on user input)
- collapse/expand the containers
- add highlighting so that a click on a node will dim all of the graph except for the node and it's links and connected nodes at 1 level of separation
- we might want to display the AST details (as a text area on the side of the screen) when a node is highlighted
- responsive design

Plus I have quite a bit of other preparation for the conference. :(

#65 - 05/28/2017 02:07 PM - Constantin Asofiei

Greg Shah wrote:

> if ON ... PERSISTENT RUN is a call-site (as in #3277-41), then why is not a SUBSCRIBE a call-site, too? Because is doing something similar, register a piece of code to be executed for an event.

This is a good point. I agree with your approach.

Related to the RAISES edge: hints will allow an event-processing call-site to link to any EVENT vertex. Later on, the reports can do something more interesting: considering the events raised by a call-site, it can report only the triggers 'in the same neighbourhood'; from the location of the call-site vertex and the location of the trigger, the trigger can be seen as 'executed' if:

- the trigger and the call-site vertex are part of the same external program
- the program where the trigger was defined is possibly invoked (i.e. there is a CALLS incoming path from the call-site to the trigger). Maybe use some depth parameter? The idea is to traverse the graph starting from the call-site via the incoming CALLS and see which triggers are defined.

How quickly can you get this (the graph itself) to a good stopping point? I will need help in the following areas:

I think it will be done by tomorrow morning your time

- encoding the Hotel GUI hints needed to get a full graph

I will work on this tomorrow.

- rework the current reports, but as dynamically calculated results that we will send down to the browser for display

Can you give me a hint what will require? Will this be done in TRPL and generate the reports based on some parameters?

- help build the API to access the call graph from the client

If you can give me a short list of what kind of access you need (something like CallGraphWorker?), it would help getting started.

> rework the current reports, but as dynamically calculated results that we will send down to the browser for display

> Can you give me a hint what will require? Will this be done in TRPL and generate the reports based on some parameters?

The idea was to eliminate the current TRPL implementation and replace it with a dynamic up-call to the report server, which would use the graph database to generate the results and send them back to the web client as JSON.

The display, formatting and so forth will be done in the JS client.

> help build the API to access the call graph from the client

> If you can give me a short list of what kind of access you need (something like CallGraphWorker?), it would help getting started.

If you've been following #1514, you may have seen some details about our API approach. See #1514-20 for the particulars about implementing an API method.

In regard to the needs of the call graph visualization, we need a mechanism to return some portion of the call graph down to the JS client for loading into the visualization. It is assumes that for any non-trivial set of programs, the entire graph cannot be loaded at any one time. I would expect that even the Hotel GUI would fall into this category. The objective is to make the visualization something that allows a developer to:

- see relationships, patterns and connections of the application that would be much harder to understand in a text/source view
- control which parts of the graph are loaded
  - by default, all nodes from files in the rootlist will be chosen (plus some nodes from connected files, see below)
  - the user should be able to select a different file set to load (Eric has an implementation of this in the report side that will need to be exposed to us)
  - some search/filtering to dynamically specify the graph starting points
- control the amount of the graph loaded at any one point in time
  - I think the main idea here is to choose the "degrees of separation" from the files that are selected as the starting points
  - by default, 1 or 2 separation levels might be good (we will see how much data it is), so the starting files and any files that are called from nodes contained in the starting files at 1 or 2 levels of call edge
  - I suspect that 3 or 4 levels is probably too much to display but we will have to see
- control some settings of the visualization such that more or less detail (of the currently loaded portion of the graph) is displayed
  - collapsing/expanding nodes will be a big one here, there are 3 levels that are possible: file, file + contained code blocks (but not the contained call site nodes), file + code blocks + call-sites (full details)
- traverse the call graph to explore the application
  - wherever there is a link to a portion of the call graph which is not loaded, there should be a placeholder node there that we will display as a "More" node
  - when clicked, the More node will reload the graph using the target node as the starting point

#67 - 05/29/2017 12:26 PM - Eric Faulhaber

Greg Shah wrote:

> the user should be able to select a different file set to load (Eric has an implementation of this in the report side that will need to be exposed to us)

We'll need to determine the best way to expose this filtering for call graph use. Right now, the set of files in use for a given user session are stored in a temporary table in H2, which is created when the user connects. This allows me to do joins for the reporting side of the house to filter results quickly.

I can give you one or more APIs to get this list of IDs (and/or file names, if that works better for you). Can you then feed this to the graph database to let it do the filtering for you?

#68 - 05/29/2017 01:13 PM - Greg Shah

I guess we need a way to get the list of the file set names.

Then for a given file set name (or ID of the file set, I guess), if you provide us a simple array or list of project-relative filenames, we are good to go.

#69 - 05/30/2017 11:43 AM - Constantin Asofiei

Greg Shah wrote:

- see relationships, patterns and connections of the application that would be much harder to understand in a text/source view

I think we can have at least these graph views:

- OO graph (Classes and so forth)
- callgraph
  - procedure/function/external program level
  - external (other) resources level
- events

...

- control the amount of the graph loaded at any one point in time
  - I think the main idea here is to choose the "degrees of separation" from the files that are selected as the starting points
  - by default, 1 or 2 separation levels might be good (we will see how much data it is), so the starting files and any files that are called from nodes contained in the starting files at 1 or 2 levels of call edge

Can't we make this configurable in the UI?

- control some settings of the visualization such that more or less detail (of the currently loaded portion of the graph) is displayed
  - collapsing/expanding nodes will be a big one here, there are 3 levels that are possible: file, file + contained code blocks (but not the contained call site nodes), file + code blocks + call-sites (full details)
- traverse the call graph to explore the application
  - wherever there is a link to a portion of the call graph which is not loaded, there should be a placeholder node there that we will display as a "More" node
  - when clicked, the More node will reload the graph using the target node as the starting point

Doesn't D3 allow a way to 'insert' dynamically a sub-graph? This might be easier (in terms of data transfer and rendering the graph) than re-creating the entire graph.

#70 - 05/30/2017 11:56 AM - Greg Shah

> Can't we make this configurable in the UI?

Yes, that is the idea.

> Doesn't D3 allow a way to 'insert' dynamically a sub-graph? This might be easier (in terms of data transfer and rendering the graph) than re-creating the entire graph.

It does and we will be using it. Anything that would be moving off the displayed graph will be pruned and any new nodes will be added and linked. But I don't expect to ever load the entire graph at the same time. So we need a way to tell the user that there is more to show and to allow the user to tell us when to show it. A click on the More node will do that. We would then make an up call to get the updated graph. While it is possible that we can do the calculations on the JS side to know how to remove and add things to the data model, I think that would be better done on the server side where the graph database can help us.

My "going in" position is that we may pull more data to the client than needed so that the JS code can be simpler. Still, the D3 update model will be used for the visualization changes.

> I think we can have at least these graph views:

I was planning on these different things being on the same graph. We can provide some filtering options to the user to hide some of it if needed.

Did you plan to separate these?

#71 - 05/30/2017 12:31 PM - Constantin Asofiei

Greg Shah wrote:

> It does and we will be using it. Anything that would be moving off the displayed graph will be pruned and any new nodes will be added and linked. But I don't expect to ever load the entire graph at the same time. So we need a way to tell the user that there is more to show and to allow the user to tell us when to show it. A click on the More node will do that. We would then make an up call to get the updated graph. While it is possible that we can do the calculations on the JS side to know how to remove and add things to the data model, I think that would be better done on the server side where the graph database can help us.

Hm... thinking more about it, yes, I think is better for the server-side to do this. Clicking More will not just create a sub-graph, this might have edges to the other already existing nodes.

> My "going in" position is that we may pull more data to the client than needed so that the JS code can be simpler. Still, the D3 update model will be used for the visualization changes.
>
> > I think we can have at least these graph views:
>
> I was planning on these different things being on the same graph. We can provide some filtering options to the user to hide some of it if needed.
>
> Did you plan to separate these?

No, if you plan to use filtering, then is OK, too.

#72 - 05/31/2017 06:50 AM - Greg Shah

> Greg wrote:
> I was thinking that your next step would be to implement an API call for each of the reports and write the new (non-TRPL) version of the back end report generation.

Constantin wrote:
For the reports, I can generate them easily, using the existing logic from the TRPL rules. I assume they will need to be placed in CallGraphApi,

> but I don't know what data is expected the JS side to receive - a plain-text file? A more complex data, like a list of rows to be displayed in a table?

I am open to ideas. If there are multiple fields, then it is best to create a List of some bean objects so that we get a more rich set of data to display. My guess is that for now, we will use the Tabulator (JS table control) for display of the results. This is the same control we use in the Hotel GUI embedded mode and for the dynamic reports that are part of 1514a.

#73 - 05/31/2017 06:52 AM - Greg Shah

> > Constantin wrote:
>
> > Currently I'm almost finished optimizing the ambiguous sites for Hotel GUI, so that ADM/ADM2 are resolved automatically. Next step is to finish encoding the remainder hints.

> Greg wrote:
>
> Are these generic enhancements to the new callgraph/adm_adm2_indirect_linkage.rules? I want to include as much generic resolution in the trunk as is possible.

Constantin wrote:

They are part of the standard callgraph rules; for example, if a method/function is defined IN SUPER and the only definitions are in ADM programs, then those will be linked via 'references' automatically. Without these, I had thousands of ambiguous virtual func/proc definitions (IN SUPER).

Also, there are lots of RUN ... or DYNAMIC-FUNCTION calls targeting a ADM function which doesn't exist; for these, I'm creating empty hints.

#74 - 05/31/2017 06:56 AM - Greg Shah

1514a rev 11220 has support for these items:

- address the initial layout, positioning bias and collision avoidance (it was tricky because of the container implementation)
- build an initial API to access a subset of the graph and connect the web client to the back end, properly (this is done except that the graph database still must be hooked up to the API)
- continue to tweak the look and feel so that the nodes and links are more intuitively understandable, at a minimum the full set of node and edge types need to be handled with proper shapes, colors, stroking and text
- add a legend

After the full database results are hooked up, I expect some tweaks to the snap points implementation and to the look/feel of some nodes that weren't handled before.

Then the remaining tasks from #1514-64 will be worked.

#75 - 05/31/2017 07:04 AM - Greg Shah

When you have the reports fixed and available in the API, you can work on the client side up call and display of the results in a tabulator. Eric has many examples in the project already and can help answer questions.

We are considering using the controls in JQuery UI for our interactive widget needs (non-svg and non-table). The tabulator uses a custom packaging of JQuery UI and we may need to expand that depending on the controls chosen. We will need to provide a mechanism to switch between the call graph visualization screen and the reports that you will be creating.

In order to add the search/filter/traversal/view settings I was expecting to add some of these controls to the visualization screen, possibly in a horizontal bar below the SVG. It may make sense to use that same space for a button to switch to the CG reports screen and on that screen you would be able to switch between the CG reports and switch back to the visualization.

#76 - 05/31/2017 04:03 PM - Constantin Asofiei

For any kind of callgraph report, I'm planning to send these details about it, to the function which will build it:

1. report title
2. column names and titles
3. a grouping column (if can be used)
4. the report data - a list of beans with each row's data

I want to build a generic approach into showing the report in a tabulator control. If we later want to expand with some custom commands (For example, show the source code for a call-site), then we can add a map with a custom decorator for each column value, as a parameter for the report builder.

#77 - 05/31/2017 04:24 PM - Greg Shah

The approach makes sense.

#78 - 05/31/2017 04:31 PM - Greg Shah

I'm testing a first pass at the CallGraphApi.readGraph(String[] files, int direction, int levels) API call. It successfully processes queries against the graph database and returns them back to JS where they are rendered.

I may need a fresh set of eyes on this. I have moved some code from CGW to static methods in a new CallGraphHelper class. One of the things I moved is the core logic for CGW.loadRootNodes() which is now called CGH.loadFileNodes(). I use it in readGraph() to find the PROCEDURE_FILE nodes matching the files list and do all the gathering from there.

The problem is that the CGH.loadFileNodes() seems to return the EXTERNAL_PROCEDURE node, not the PROCEDURE_FILE node, even though we are doing a lookup of the filename using os-filename. Can you look at this and see if you can find my mistake?

#79 - 05/31/2017 04:57 PM - Constantin Asofiei

Greg Shah wrote:

> The problem is that the CGH.loadFileNodes() seems to return the EXTERNAL_PROCEDURE node, not the PROCEDURE_FILE node, even though we are doing a lookup of the filename using os-filename. Can you look at this and see if you can find my mistake?

I'm looking at this.

#80 - 05/31/2017 05:18 PM - Constantin Asofiei

Greg, I see two issues:

1. when processing BOTH directions, you need to make sure a vertex is processed only once - otherwise, although the sets will prevent a leak, the more edges will leak... plus, stackoverflow issue.
2. you are assuming that node-id is unique for the entire callgraph? This is true only in the same resource-domain: i.e. AST_NODES, FILE_RESOURCE.

#81 - 05/31/2017 05:24 PM - Constantin Asofiei

Greg, and the root cause why the graph isn't loading: in createSnippet, you are using node-id. Instead, you can to use Vertex.id() property, which is a long value unique across the graph vertices. You can place the node-id in some other property?

But, the real problem: JS doesn't support 64-bit value, it will truncate them (as I recall to 53 bits for natural numbers). So, we need a way to keep the IDs unique - maybe make them a string? Or D3 works only with numbers?

#82 - 05/31/2017 05:39 PM - Constantin Asofiei

And something else; in this code:

```
            if (direction == DOWNSTREAM)
            {
               edges = db.traversal().V(v).outE().toList();
               next  = db.traversal().V(v).out().toList();
            }
            else if (direction == UPSTREAM)
            {
               edges = db.traversal().V(v).inE().toList();
               next  = db.traversal().V(v).in().toList();
            }
            else
            {
               // otherwise assume BOTH_DIRECTIONS
               edges = db.traversal().V(v).bothE().toList();
               next  = db.traversal().V(v).both().toList();
            }
```

You are getting any kind of edges, not just contains.

#83 - 05/31/2017 05:58 PM - Constantin Asofiei

Greg, beside some other JS errors, I can get data into the web graph. But, I don't understand something: what kind of graph do you want to load?

You can't just use 'contains' to get all procedure files from the root files - you need to gather all the nodes from the graph explicitly, and load the 'contains' sub-graph for each of them (as there will be no intersect).

The root nodes are best suitable in combination with 'contains' and 'calls', to get the call-graph.

#84 - 05/31/2017 06:26 PM - Greg Shah

> you are assuming that node-id is unique for the entire callgraph? This is true only in the same resource-domain: i.e. AST_NODES, FILE_RESOURCE.

Yes, this is important for d3 to work properly. Not only must the ids be unique, they must be stable so that d3 can figure out the additions and deletions.

> in createSnippet, you are using node-id. Instead, you can to use Vertex.id() property, which is a long value unique across the graph vertices. You can place the node-id in some other property?

We will need the AST node id for finding the AST details to display and for linking with the source code/ast view.

But we don't have to use the AST node id for the d3 node id. The call graph vertex id() is unique, so we can use that.

> But, the real problem: JS doesn't support 64-bit value, it will truncate them (as I recall to 53 bits for natural numbers). So, we need a way to keep the IDs unique - maybe make them a string? Or D3 works only with numbers?

Yes, we can use strings. d3 is fine with that.

> You are getting any kind of edges, not just contains.

This is expected. The contains links are being processed above there. I want all the other edges (calls...). But you do have a point , that we will need to exclude contains edges from these.

> You can't just use 'contains' to get all procedure files from the root files - you need to gather all the nodes from the graph explicitly, and load the 'contains' sub-graph for each of them (as there will be no intersect).
>
> The root nodes are best suitable in combination with 'contains' and 'calls', to get the call-graph.

I thought that was what I was doing. Lines 268 through 274 of CallGraphApi.traverseLevel() are meant to find all the vertices that are connected to a single procedure_file node (in the start list) through a 1st order or 2nd order contains edge.

Then lines 276 through 329 are meant to load everything else into the snippet. As noted above, I do need to exclude the contains relationships from this second set.

#85 - 05/31/2017 06:29 PM - Constantin Asofiei

Greg, please see revision 11227 - this shows data into the D3 web graph, but it doesn't look OK; not sure what is wrong, I'm trying to use a smaller set of programs (standalone testcases) to check.

#86 - 05/31/2017 06:30 PM - Greg Shah

One other thing I need help with: I need a simple and generic way to get a descriptive text string for each node. If I lookup the "node-key" value, can I then use that value to obtain a good descriptive string?

#87 - 05/31/2017 06:32 PM - Constantin Asofiei

Greg Shah wrote:

> One other thing I need help with: I need a simple and generic way to get a descriptive text string for each node. If I lookup the "node-key" value, can I then use that value to obtain a good descriptive string?

You can get this in two steps:

1. get the node-key property value, which maps to a property name with the description
2. get the value of this referenced property

So, you will need to do something like v.property(v.property("node-key").value().toString()).value()

#88 - 05/31/2017 06:44 PM - Greg Shah

The changes in 11227 are not quite correct.

```
if (direction == DOWNSTREAM)
            {
               edges = db.traversal().V(v).outE("contains").toList();
               next  = db.traversal().V(v).out("contains").toList();
            }
            else if (direction == UPSTREAM)
            {
               edges = db.traversal().V(v).inE("contains").toList();
               next  = db.traversal().V(v).in("contains").toList();
            }
            else
            {
               // otherwise assume BOTH_DIRECTIONS
               edges = db.traversal().V(v).bothE("contains").toList();
               next  = db.traversal().V(v).both("contains").toList();
            }
```

These should not be using the "contains" edges. We have already traversed the "contains" up on line 270. The nodes in the contained list are being processed to find the non-contains edges inside the loop on line 276.

Adding the start nodes to the list is an important fix. But the contained also need to be added where I added them before. Then the internal loop at 276 can go back to processing the non-contains links. Perhaps we can use something like outE().not(hasLabel("contains")) and the equivalent for all 6 cases?

Finally, we won't want to use the node-id values to check the direction of traversal to determine the other node. We need to shift that to the vertex id too.

Let me know if I should make these edits. However, I have some other things to do and I think it is a good opportunity for you to keep going. :) If you are willing.

#89 - 05/31/2017 06:58 PM - Constantin Asofiei

Greg Shah wrote:

> These should not be using the "contains" edges. We have already traversed the "contains" up on line 270. The nodes in the contained list are being processed to find the non-contains edges inside the loop on line 276.

OK. But what is the idea to get these "non-contains" edges?

> But the contained also need to be added where I added them before.

Well, with the code as it originally was, there will be stackoverflow, as there is no protection to process same vertex twice. What about this:

```
            if (nodes.contains(v))
            {
               continue;
            }

            nodes.add(v);
```

instead of my previous approach.

> Then the internal loop at 276 can go back to processing the non-contains links. Perhaps we can use something like outE().not(hasLabel("contains")) and the equivalent for all 6 cases?

OK, I'll try.

> Finally, we won't want to use the node-id values to check the direction of traversal to determine the other node. We need to shift that to the vertex id too.

Fixed.

> Let me know if I should make these edits. However, I have some other things to do and I think it is a good opportunity for you to keep going. :) If you are willing.

I'll keep going, but I don't understand why no edges are drawn (and what is the meaning of the graph) in the web UI. I'll try to look into these, too...

#90 - 05/31/2017 07:50 PM - Constantin Asofiei

Greg, see revision 11228 - there was another problem, between first 2 contains levels no edges were being added.

But there is still weird for Hotel GUI - I'm seeing an external proc (I think) included in another one. I'll look into it tomorrow.

#91 - 06/01/2017 05:04 AM - Greg Shah

> still have some KW_DYN_FUNC cases (Which I'm not really sure how to resolve automatically)

What about the techniques we use during conversion? We match the hard coded name references and disambiguate by type information when there are multiple options. Anything remaining is ambiguous and would need hints.

#92 - 06/01/2017 05:08 AM - Constantin Asofiei

Greg Shah wrote:

> > still have some KW_DYN_FUNC cases (Which I'm not really sure how to resolve automatically)
>
> What about the techniques we use during conversion? We match the hard coded name references and disambiguate by type information when there are multiple options. Anything remaining is ambiguous and would need hints.

Well, I'm trying to 'indirectly solve' these cases, as the name is not hard-coded, to avoid writing hints by hand. But is not that explicit in the ADM code, to get all the possible values for the target function.

#93 - 06/01/2017 07:34 AM - Greg Shah

> But what is the idea to get these "non-contains" edges?

The top part of the method (above the non-contains part) is designed to get all of the contained nodes for a specified procedure_file. When these are added into the snippet along with their associated contains edges, the result will draw a 2 level set of container objects. The outer container will be "silver" colored and the inner containers will be the external_procedure, internal_procedure, function and so forth. These are the code blocks which are entry points (and can have inbound edges that are not contains). The actual call sites will be inside these code blocks and they will have outbound edges that are "calls" connecting to the code blocks.

The code section to get the non-contains parts is meant to follow the non-contains links to the vertices that are connected. Some of them may be in the same procedure_file, in 11130 I have excluded these. The rest of these are nodes outside of the procedure file. If the traversal is done, then we create placeholders. If not, we recurse to get the next level. I think that recursion is broken right now and have put in a todo to explain the fix that is needed.

> Well, with the code as it originally was, there will be stackoverflow, as there is no protection to process same vertex twice. What about this:
> ...
> instead of my previous approach.

Yes, it seems good.

> I'll keep going, but I don't understand why no edges are drawn

I don't know why that is either, though I suspect the problem is in createSnippet(), the changes there don't look right.

The visual result is getting closer to the toy graph I previously was using. I also see that the z-order is wrong. The external procedure exists and is drawn underneath the procedure_file node for start.p. This ordering is purely based on the order in which the nodes occur in the graph snippet. We must make sure that all containing nodes occur before their contained nodes.

I also don't see any of the placeholders in the node list, perhaps that is related to the reason why no links are drawn.

> (and what is the meaning of the graph) in the web UI.

I'm not sure what you question is here.

> I'm seeing an external proc (I think) included in another one.

I think this is OK, just a z-order issue. My toy graph was setting the text for the "EXTERNAL_PROCEDURE" node but right now this is pciking up the v.label() which is start.p so it seems wrong, but it is not wrong. We need to clean up the text in these nodes to be better.

I have to get back to creating some slides that have to go to the printer today. Please keep going with this to see if you can clear the known issues here.

#94 - 06/01/2017 11:04 AM - Constantin Asofiei

Greg Shah wrote:

> (and what is the meaning of the graph) in the web UI.

I'm not sure what you question is here.

I meant why the graph looked like that, but I think I understood now - you are correct, the node order is important; so we must use a LinkedHashSet - this ensures the parent vertices are added before the child vertices.

> I'm seeing an external proc (I think) included in another one.

I think this is OK, just a z-order issue. My toy graph was setting the text for the "EXTERNAL_PROCEDURE" node but right now this is pciking up the v.label() which is start.p so it seems wrong, but it is not wrong. We need to clean up the text in these nodes to be better.

Can you point me how D3 computes the z-order? I would expect the top-level FILE_RESOURCE blocks to not intersect...

> Please keep going with this to see if you can clear the known issues here.

Beside the z-order issue, the edges are being drawn correctly now, and the graph can be fully loaded (for example, my simple callgraph tests).

#95 - 06/01/2017 01:36 PM - Greg Shah

> Can you point me how D3 computes the z-order?

D3 does nothing with z-order. The SVG's natural z-order will be what you get. The z-order for SVG is the order of the elements (from top to bottom), which most often is the element insertion order, unless you move things around.

I have made assumptions in how we create nodes, such that any nodes that need to be contained, must appear later in the node list.

I also am intentionally creating the links last so that they appear on top of everything.

> I would expect the top-level FILE_RESOURCE blocks to not intersect...

My code that calls the API is requesting the default rootlist (for Hotel GUI this is just start.p, I've recently checked in that file), BOTH directions and 0 levels. This means that there should only be 1 FILE_RESOURCE which is the PROCEDURE_FILE for start.p.

I've implemented a 2 level containment hierarchy so that we do not add links for "contains". Instead, we visually contain call sites inside code blocks and code blocks inside the PROCEDURE_FILE block.

It does appear that we are incorrectly handling the KW_ON (trigger blocks?) right now. These are being treated as a hybrid of container and a node contained in a code block, with understandably bad results.

What are you working on right now? I've finished the print jobs so I can back to working on this. My plan is to resolve the following:

- fix the node text to be more descriptive, this is especially important for the MORE nodes which should display as a small circle with the text More
- fix the tooltips to display useful text instead of the token type
- add display details for new node and link types so that they are not drawing with "gold" fill and "chartruse" stoke (I picked these defaults so it would be easy to see which ones don't have details)
- fix the legend
- add the node-id to the graph node so that a linkage to the AST can be made
- add the AST details on hover (I plan to create a list of the AST nodes in the graph snippet/server side so that we have enough data to feed the AST details worker without an up-call)

Once the initial rendering (z-order, sizing, positioning...) issues are all resolved, we can test different queries (specific directions and > 0 for the level). I expect to do some work on the initial layout of the snap points for the main screen ("2nd level containers") because today calculates based on the SVG height and width but on a larger graph this won't work well.

Are you testing using Hotel GUI or something else? Can you check in your Hotel GUI hints?

#96 - 06/01/2017 02:13 PM - Constantin Asofiei

Greg Shah wrote:

> I would expect the top-level FILE_RESOURCE blocks to not intersect...

My code that calls the API is requesting the default rootlist (for Hotel GUI this is just start.p, I've recently checked in that file), BOTH directions and 0 levels. This means that there should only be 1 FILE_RESOURCE which is the PROCEDURE_FILE for start.p.

There are three entry points: start.p, emain.p and ehotel.p.

I've implemented a 2 level containment hierarchy so that we do not add links for "contains". Instead, we visually contain call sites inside code blocks and code blocks inside the PROCEDURE_FILE block.

For UPSTREAM mode, how do you decide which node contains which, if there are no OUT @CONTAINS edges added? Latest changes in 11231 adds the CONTAINS edges for these 2-level vertices. For current BOTH mode, these are also resolved in the inner loop.

It does appear that we are incorrectly handling the KW_ON (trigger blocks?) right now. These are being treated as a hybrid of container and a node contained in a code block, with understandably bad results.

For some reason I'm adding a CALLS and CONTAINS link. Only CALLS is needed, I'm fixing it now.

What are you working on right now? I've finished the print jobs so I can back to working on this. My plan is to resolve the following:

I'm working on the reports JS; with 11231 the graph was looking good, except the overlapping of the procedure file nodes.

Are you testing using Hotel GUI or something else? Can you check in your Hotel GUI hints?

Yes, with Hotel GUI. The hints are not yet complete, but the TRPL callgraph rules will resolve almost all of the ADM calls automatically.

#97 - 06/01/2017 02:43 PM - Greg Shah

I just checked in some improvements to text and tooltips.

> There are three entry points: start.p, emain.p and ehotel.p.

I assume you'll check in all your changes to Hotel GUI so these will appear.

BTW, I've added an ant callgraph target to build.xml. You can tweak it if you find something wrong.

> I've implemented a 2 level containment hierarchy so that we do not add links for "contains". Instead, we visually contain call sites inside code blocks and code blocks inside the PROCEDURE_FILE block.
>
> For UPSTREAM mode, how do you decide which node contains which, if there are no OUT @CONTAINS edges added? Latest changes in 11231 adds the CONTAINS edges for these 2-level vertices. For current BOTH mode, these are also resolved in the inner loop.

All nodes which are included in the graph snippet AND which have contains edges, must have those contains edges added to the graph snippet.

The only exception is when we are creating the MORE placeholders. For those cases, we don't care about any edges to that node except for the one from the "current traversal level".

In regard to your upstream question, the idea of upstream or downstream is in regard to other procedure_file vertices that contain nodes that call us (upstream) or that contain nodes that get called by us (downstream). I agree that in either direction, we must build the list of procedure_files by properly traversing the contains link from the call site up to the procedure_file (upstream) or from the entry point up to the procedure_file (downstream). The current approach is not correct.

But to fix it we must handle the TODO on line 355 of CallGraphApi.

> > It does appear that we are incorrectly handling the KW_ON (trigger blocks?) right now. These are being treated as a hybrid of container and a node contained in a code block, with understandably bad results.
>
> For some reason I'm adding a CALLS and CONTAINS link. Only CALLS is needed, I'm fixing it now.

Any scenario that adds contains edges to the snippet will need to have code changes on the client to get things right. We currently assume that each "top level" code block (entry points like external_procedure, internal_procedure, trigger, function) have a single inbound contains link that originates at the procedure_file. We likewise assume that all call sites have a single inbound contains link that originates in one of the top-level code blocks.

Any deviation from that needs to be handled specially.

#98 - 06/01/2017 02:52 PM - Constantin Asofiei

Greg Shah wrote:

> In regard to your upstream question, the idea of upstream or downstream is in regard to other procedure_file vertices that contain nodes that call us (upstream) or that contain nodes that get called by us (downstream). I agree that in either direction, we must build the list of procedure_files by properly traversing the contains link from the call site up to the procedure_file (upstream) or from the entry point up to the procedure_file (downstream). The current approach is not correct.
>
> But to fix it we must handle the TODO on line 355 of CallGraphApi.

There is the cgw.findProcedureFile(Vertex) (called from TRPL) which can be moved to CallGraphHelper.

> Any scenario that adds contains edges to the snippet will need to have code changes on the client to get things right. We currently assume that each "top level" code block (entry points like external_procedure, internal_procedure, trigger, function) have a single inbound contains link that originates at the procedure_file. We likewise assume that all call sites have a single inbound contains link that originates in one of the top-level code blocks.

OK, I'll rework the callgraph TRPL rules under this assumption (only the KW_ON is problematic).

#99 - 06/01/2017 02:56 PM - Constantin Asofiei

Constantin Asofiei wrote:

> > Any scenario that adds contains edges to the snippet will need to have code changes on the client to get things right. We currently assume that each "top level" code block (entry points like external_procedure, internal_procedure, trigger, function) have a single inbound contains link that originates at the procedure_file. We likewise assume that all call sites have a single inbound contains link that originates in one of the top-level code blocks.
>
> OK, I'll rework the callgraph TRPL rules under this assumption (only the KW_ON is problematic).

There was something bugging me about this 2-level approach and it just clicked: triggers can be nested and/or defined in an internal procedure (or function). Making the CONTAINS at the procedure file is not correct, if the trigger is not directly defined in the external program. Even so, the trigger belongs more to the external procedure and not the procedure_file...

#100 - 06/01/2017 03:07 PM - Greg Shah

> There was something bugging me about this 2-level approach and it just clicked: triggers can be nested and/or defined in an internal procedure (or function). Making the CONTAINS at the procedure file is not correct, if the trigger is not directly defined in the external program.

Do we want to fully model this relationship? If so, then we will have to support n-levels of containment and a more complicated implementation on both the JS and server sides.

> Even so, the trigger belongs more to the external procedure and not the procedure_file...

If we model it this way, then if the trigger is contained in an internal_procedure or another trigger it would be contained there and is not associated with the external_procedure.

I think triggers are the only example of a callable code block (a code block with an entry point) that is contained inside another code block. I can't think of any other such case.

#101 - 06/01/2017 03:08 PM - Greg Shah

Also, the KW_ON is a registration of the trigger, but it is not really the trigger itself. Perhaps the distinction does not matter.

#102 - 06/01/2017 03:11 PM - Constantin Asofiei

Greg Shah wrote:

> > There was something bugging me about this 2-level approach and it just clicked: triggers can be nested and/or defined in an internal procedure (or function). Making the CONTAINS at the procedure file is not correct, if the trigger is not directly defined in the external program.
>
> Do we want to fully model this relationship? If so, then we will have to support n-levels of containment and a more complicated implementation on both the JS and server sides.

If we want to properly manage the parent-child relation when triggers are involved, then yes. Is this assumption the reason why you have the 2-level "contains" edge in CallGraphApi (i.e. you assumed that in 2 levels the entire "contains" sub-graph for a procedure file will be loaded)?

> > Even so, the trigger belongs more to the external procedure and not the procedure_file...
>
> If we model it this way, then if the trigger is contained in an internal_procedure or another trigger it would be contained there and is not

> associated with the external_procedure.

I meant triggers defined in the external procedure should be linked via CONTAINS from the external procedure, and not the procedure file vertex.

> I think triggers are the only example of a callable code block (a code block with an entry point) that is contained inside another code block. I can't think of any other such case.

Yes, this is the only case, but is enough to complicate things.

#103 - 06/01/2017 03:18 PM - Greg Shah

> If we want to properly manage the parent-child relation when triggers are involved, then yes.

Let's fix this. I can fix the JS side. I would like you to fix the readGraph() processing. While you are there, please also fix the recursion issue (which we haven't tested yet, but which will be broken the first time levels > 0).

> Is this assumption the reason why you have the 2-level "contains" edge in CallGraphApi (i.e. you assumed that in 2 levels the entire "contains" sub-graph for a procedure file will be loaded)?

I had not been carefully watching your changes for events/triggers, so I wrote this 2-level containment approach without considering the trigger case.

#104 - 06/01/2017 03:19 PM - Constantin Asofiei

Greg Shah wrote:

> > If we want to properly manage the parent-child relation when triggers are involved, then yes.
>
> Let's fix this. I can fix the JS side. I would like you to fix the readGraph() processing. While you are there, please also fix the recursion issue

(which we haven't tested yet, but which will be broken the first time levels > 0).

OK, I'll do it. The start parameter must always include vertices for the procedure files, correct?

#105 - 06/01/2017 03:43 PM - Greg Shah

> The start parameter must always include vertices for the procedure files, correct?

I had written the code so that start would only hold procedure_file nodes. The idea is that it holds those procedure_file nodes that are 1 level away from the current procedure_file node's traversal level.

However, for non-ast nodes, this will not be correct. We should change the code to deal with any FILE_RESOURCE domain, not just procedure_file nodes.

The key here is that the levels of separation concept is being implemented at the topmost container. Any such container that is included in the levels of separation would have all of its contained nodes (and the associated links) included in the snippet. At level 0 in the traversal recursion, any external links are connected to the MORE placeholder.

#106 - 06/01/2017 03:48 PM - Greg Shah

Do you have a list of all possible graph vertex node-type values?

Do you have a list of all possible edge labels?

#107 - 06/01/2017 04:55 PM - Constantin Asofiei

Greg Shah wrote:

> Do you have a list of all possible graph vertex node-type values?

Vertex node-type possible property value:

1. special:
   AMBIGUOUS
2. external
   MISSING
   LIBRARY_PROCEDURE
   PORT_TYPE
   SHARED_LIBRARY
   NATIVE_API
   NATIVE_PROCESS
   NETWORK_CONNECTION
   COM_OBJECT

DDE_SERVER
OCX_CONTROL

3. schema related
   SCHEMA_FILE
   TABLE
   DATABASE
   KW_TAB_TRG
4. OO related
   CLASS_DEF
   INTERFACE_DEF
   DEFINE_PROPERTY_SET
   DEFINE_PROPERTY_GET
   CONSTRUCTOR
   DESTRUCTOR
   METHOD_DEF
   METHOD_INVOCATION
   PROPERTY_INVOCATION
   OBJECT_INVOCATION
5. 4GL program structure
   PROCEDURE_FILE
   EXTERNAL_PROCEDURE
   INTERNAL_PROCEDURE
   FUNCTION
   INCLUDE_FILE
6. event related
   EVENT
   KW_ON
   TRIGGER_BLOCK
   KW_MSG
   KW_PAUSE
   KW_WAIT_FOR
   KW_READKEY
   KW_INSERT
   KW_APPLY
   PROCESS_STATEMENTS
   KW_PRMT_FOR
   KW_SET
   KW_UPDATE
7. call-site related
   KW_SUPER
   FUNCTION_CALL
   KW_DYN_FUNC
   RUN_INT_PROC
   RUN_INT_PROC_ON_SERVER
   RUN_FILENAME
   RUN_VALUE
   RUN_SUPER
   RUN_PORT_TYPE_ON_SERVER
   RUN_PORT_TYPE_VALUE_ON_SERVER
   RUN_FILENAME_ON_SERVER
   RUN_LIBRARY_REF_ON_SERVER
   RUN_VALUE_ON_SERVER
   RUN_LIBRARY_REF
8. other
   DDE_INITIATE
   CREATE_OBJECT
   KW_LOADCTRL
   KW_CONN
   KW_ENABLE_C
9. OS call-sites
   KW_BTOS
   KW_DOS
   KW_OS2
   KW_UNIX
   KW_VMS
   KW_OS_CMD
   KW_CTOS
   INPUT_THRU
   INPUT_OUTPUT_THRU
   OUTPUT_THRU

Do you have a list of all possible edge labels?

Edges label types (all lowercase): contains, includes, references, inherits, implements, override, listens, raises, calls, ambiguous.

#108 - 06/01/2017 05:13 PM - Constantin Asofiei

Greg Shah wrote:

> The start parameter must always include vertices for the procedure files, correct?

I had written the code so that start would only hold procedure_file nodes. The idea is that it holds those procedure_file nodes that are 1 level away from the current procedure_file node's traversal level.

However, for non-ast nodes, this will not be correct. We should change the code to deal with any FILE_RESOURCE domain, not just procedure_file nodes.

In FILE_RESOURCE domain, we have only procedure_file, include_file and schema_file resources. I'll include only these reachable vertices (reachable the contained vertices). I need to know the explicit set of resource domains because determining the root node is different for each domain.

#109 - 06/01/2017 05:21 PM - Greg Shah

I don't think the KW_ON should be linked to a TRIGGER as calls. The ON statement doesn't call the trigger. The trigger is invoked as a callback in response to an event. I think the matching event should have the CALLS link to the trigger.

In my opinion, this is a more intuitive model because one doesn't need to know anything about the ON statement to follow the RAISES link to the EVENT and then follow the CALLS link to the TRIGGER. It is not intuitive to follow the RAISES link to the EVENT and then follow the LISTENS link backwards to the ON statement and then the CALLS link to the TRIGGER.

#110 - 06/01/2017 05:27 PM - Greg Shah

Constantin Asofiei wrote:

> Greg Shah wrote:
>
> > The start parameter must always include vertices for the procedure files, correct?
>
> I had written the code so that start would only hold procedure_file nodes. The idea is that it holds those procedure_file nodes that are 1 level away from the current procedure_file node's traversal level.
>
> However, for non-ast nodes, this will not be correct. We should change the code to deal with any FILE_RESOURCE domain, not just procedure_file nodes.

> In FILE_RESOURCE domain, we have only procedure_file, include_file and schema_file resources. I'll include only these reachable vertices (reachable the contained vertices). I need to know the explicit set of resource domains because determining the root node is different for each domain.

Hmmm. OK, then my idea is not exactly right. The vertices are not limited to the FILE_RESOURCE domain. For example, the next level of traversal might be a top-level container node in NATIVE_PROGRAM_RESOURCE (e.g. a shared library) or an IPC_RESOURCE. I think the key here is that they should NOT be an AST_NODE domain unless there is a case where the top-level container node it itself an AST_NODE. I don't think that is possible.

#111 - 06/01/2017 05:30 PM - Constantin Asofiei

Greg Shah wrote:

> Constantin Asofiei wrote:
>
> > Greg Shah wrote:
> >
> > > The start parameter must always include vertices for the procedure files, correct?
> >
> > I had written the code so that start would only hold procedure_file nodes. The idea is that it holds those procedure_file nodes that are 1 level away from the current procedure_file node's traversal level.
> >
> > However, for non-ast nodes, this will not be correct. We should change the code to deal with any FILE_RESOURCE domain, not just procedure_file nodes.
>
> In FILE_RESOURCE domain, we have only procedure_file, include_file and schema_file resources. I'll include only these reachable vertices (reachable the contained vertices). I need to know the explicit set of resource domains because determining the root node is different for each domain.

Hmmm. OK, then my idea is not exactly right. The vertices are not limited to the FILE_RESOURCE domain. For example, the next level of traversal might be a top-level container node in NATIVE_PROGRAM_RESOURCE (e.g. a shared library) or an IPC_RESOURCE. I think the key here is that they should NOT be an AST_NODE domain unless there is a case where the top-level container node it itself an AST_NODE. I don't think that is possible.

Then I can add to this list any node which has the external=true property. I think the 3 cases above (programs, include, schema) and external nodes (like missing programs, shared library - excluding native APIs, etc) will cover all the cases you have in mind.

#112 - 06/01/2017 05:42 PM - Greg Shah

That makes sense.

#113 - 06/01/2017 05:50 PM - Constantin Asofiei

Greg, you've rolled back some edits I made to swapPointsWorker in callgraph.js - without those edits, there are JS-related errors when drawing the graph.

#114 - 06/01/2017 06:06 PM - Greg Shah

In what revision were the edits?

#115 - 06/01/2017 06:08 PM - Greg Shah

Nevermind. I see them.

#116 - 06/01/2017 06:17 PM - Constantin Asofiei

Greg, see revision 11235 for the readGraph changes, an attempt to improve KW_ON and other misc fixes.

Note that the Hotel GUI graph loads very slow, for the 3 root entry points I mentioned; the reason is that if there are 100s of functions/internal procedures, the procedure_file rectangle gets very large, plus all the links to the "MORE" node...

#117 - 06/01/2017 06:58 PM - Greg Shah

I have my changes for the n-level contains, but it is not working properly. I'm debugging it now, on top of your changes. I've put back in the swapPointsWorker() change you mentioned. I don't know how it was dropped.

I am checking in my code so you can see what I am seeing. I am noticing some things that don't make sense:

Some trigger blocks are not contained at all (from the console log), depth 0 means not contained:

```
calcContainment main sizing loop 3 TRIGGER_BLOCK TRIGGER_BLOCK 4276312 contained 0 depth 0
```

There are many MORE nodes, even for things like CLOSE event, RETURN event and the AMBIGUOUS node. These should never be MORE nodes. The events probably are supposed to be contained in the procedure_file in some way. And the AMBIGUOUS node (and things like MISSING..) should probably never be treated like off-screen resources.

#118 - 06/01/2017 07:29 PM - Greg Shah

I think most of the CONTAINS links are missing from the graph snippet. This is the full list for my Hotel GUI setup (which is only trying to load ./abl/start.p:

```
calcContainment CONTAINS target EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 2146360 source ./abl/start.p PROCEDURE_F
ILE 2142264  callgraph.js:587:16
calcContainment CONTAINS target TRIGGER_BLOCK TRIGGER_BLOCK 1970304 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDU
RE 2146360  callgraph.js:587:16
calcContainment CONTAINS target KW_APPLY KW_APPLY 5615624 source TRIGGER_BLOCK TRIGGER_BLOCK 1970304  callgrap
h.js:587:16
calcContainment CONTAINS target TRIGGER_BLOCK TRIGGER_BLOCK 2015272 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDU
RE 2146360  callgraph.js:587:16
calcContainment CONTAINS target KW_APPLY KW_APPLY 5623896 source TRIGGER_BLOCK TRIGGER_BLOCK 2015272  callgrap
```

```
h.js:587:16
calcContainment CONTAINS target KW_ON KW_ON 2150456 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 2146360  call
graph.js:587:16
calcContainment CONTAINS target KW_ON MORE 2257032 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 2146360  callg
raph.js:587:16
calcContainment CONTAINS target KW_RUN RUN_FILENAME 2654248 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 21463
60  callgraph.js:587:16
calcContainment CONTAINS target KW_RUN RUN_INT_PROC 2658344 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 21463
60  callgraph.js:587:16
calcContainment CONTAINS target KW_APPLY KW_APPLY 2793528 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 2146360
  callgraph.js:587:16
calcContainment CONTAINS target KW_WAIT_FOR KW_WAIT_FOR 2896088 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 2
146360  callgraph.js:587:16
calcContainment CONTAINS target KW_WAIT_FOR KW_WAIT_FOR 2900184 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 2
146360  callgraph.js:587:16
calcContainment CONTAINS target KW_ON MORE 4272216 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 2146360  callg
raph.js:587:16
calcContainment CONTAINS target TRIGGER_BLOCK MORE 4276312 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 214636
0  callgraph.js:587:16
calcContainment CONTAINS target KW_UPDATE KW_UPDATE 5619720 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 21463
60  callgraph.js:587:16
calcContainment CONTAINS target KW_MSG KW_MSG 5623816 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 2146360  ca
llgraph.js:587:16
```

No INTERNAL_PROCEDURE nodes, no FUNCTION nodes and only 1 TRIGGER_BLOCK node contains anything. This seems wrong.

My code is still not right (because even with this, the EXTERNAL_PROCEDURE and PROCEDURE_FILE should have a larger size and contain things, so I am still debugging my code.

#119 - 06/01/2017 07:48 PM - Greg Shah

One more fix is in, but it still isn't correct.

The sizes of the containers are calculated properly, but we never get to draw them right. There are two calls links and one listens link that get their x1/x2/y1/y2 values set to NaN. This happens if there is a problem in the node bounds or if the intersection code is broken. Once this happens, much of the rest of the model will fail to render properly, so I need to find this failure first.

Another note: there are multiple MORE nodes that reference AMBIGUOUS. Shouldn't there just be one AMBIGUOUS super node?

Also, there are 31 initializeObject MORE nodes. It is not clear if these are duplicative or if they are really different target nodes. We need to provide more details in these nodes (not just the MORE node, but all of the call sites so that you can tell what the node is doing be looking at the graph. For example, initializeObject operates on some parameters that are very important... Also the domain and file/resource in which the MORE is contained

would be very helpful.

#120 - 06/01/2017 07:53 PM - Greg Shah

Strangely, the nodes that I am failing on are these:

```
calcContainment CONTAINS target KW_ON MORE 2257032 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 2146360  callg
raph.js:587:16
calcContainment CONTAINS target KW_ON MORE 4272216 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 2146360  callg
raph.js:587:16
calcContainment CONTAINS target TRIGGER_BLOCK MORE 4276312 source EXTERNAL_PROCEDURE EXTERNAL_PROCEDURE 214636
0
```

All of these are MORE nodes that are listed as targets of "contains" links, which should never happen. Anything that is a MORE node cannot be contained in the current procedure_file nor would it be contained elsewhere because then it wouldn't be a MORE.

#121 - 06/01/2017 08:25 PM - Constantin Asofiei

Greg, OK, I'm looking at the issues now.

I've managed to get into a good step with the client-side infrastructure for the callgraph reports; basically, I have:

1. left-side report data (i.e. call to missing targets - functions, procedures, etc);
2. when a row is clicked on the left-side, right-side is populated with the call-sites (per each file)
3. when a row is clicked on the right-side, it gets us to the real code location.

I plan to use the same paradigm for all reports, if it makes sense for them (and you).

#122 - 06/01/2017 08:49 PM - Greg Shah

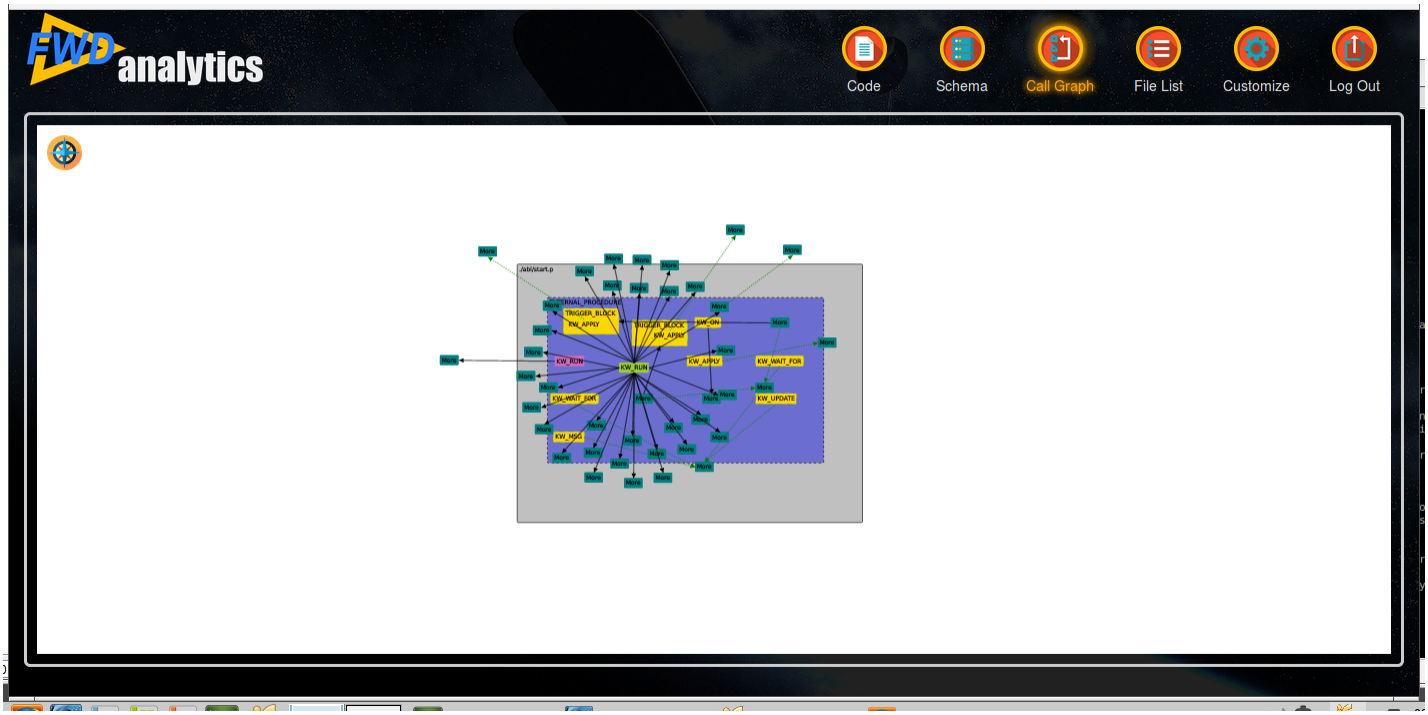
Constantin: your approach sounds pretty good. Can you check it in so I can review it?

I've checked in fixes for my n-level containment. It is working now, through we still have a contained shape is a circle issue, but I've bypassed it temporarily by forcing all shapes to rect. I'll fix that next.

Once you clear some of the other issues I've noted, we can see what other tweaks will be needed. The result is not bad already.

#123 - 06/01/2017 08:51 PM - Greg Shah

- File call_graph_on_hotel_gui_branch_1514a_rev_11239_20170601.png added



#124 - 06/01/2017 09:01 PM - Constantin Asofiei

Greg Shah wrote:

> Constantin: your approach sounds pretty good. Can you check it in so I can review it?

Is committed to rev 11240. Note that the bottom nav-bar look&feel is not that good, but the click works :)

> Once you clear some of the other issues I've noted, we can see what other tweaks will be needed. The result is not bad already.

What I plan to do tomorrow is this:

1. see the KW_ON/TRIGGER_BLOCK support, how it can be reworked. The MORE issue related to KW_ON you noted in a previous commented is related to the KW_ON -CALLS-> TRIGGER_BLOCK links.
2. add more reports
3. for the other MORE cases, we need to determine what "stop" vertex mean: for example, a call-site targeting an external resource might be a 'stop' vertex. Better said, any other node which doesn't have any out edges should not be marked as MORE.

#125 - 06/02/2017 07:07 AM - Greg Shah

> for the other MORE cases, we need to determine what "stop" vertex mean: for example, a call-site targeting an external resource might be a 'stop' vertex.

Any non-local (not in the current resource (the same PROCEDURE_FILE, NATIVE_LIBRARY...) vertex, subject to the exclusion below.

> Better said, any other node which doesn't have any out edges should not be marked as MORE.

Yes, this is an exception. We should just include it since we already are taking space for it, why click on it just to make it re-render it as the proper node?

#126 - 06/02/2017 07:17 AM - Constantin Asofiei

For the RUN initializeObject IN hSmartWindow. case, I think I have a better way than just link to all definitions of initializeObject: for RUN ... IN handleVar cases, collect all external program targets of a RUN ... PERSISTENT SET handleVar statement. Thus, when RUN ... IN handleVar is reached, it will be able to link only to those external programs; otherwise, if nothing was collected (or the target is not a handleVar), fallback to previous way.

#127 - 06/02/2017 07:43 AM - Constantin Asofiei

About the trigger blocks and UI statements raising events: in a previous note you mentioned that the UI statement should have a calls link to the trigger block. This can only be done via hints, and we will have to point to the trigger by its code location, beside the external program where is defined.

#128 - 06/02/2017 09:17 AM - Constantin Asofiei

Greg, what was the reason for the Configuration.forceHome(changes in RootNodeList.java rev 11225? With these changes, I can no longer run callgraph for my testcases/ setup... at least it should check if the path isn't starting with the home, before appending it.

#129 - 06/02/2017 09:30 AM - Greg Shah

> About the trigger blocks and UI statements raising events: in a previous note you mentioned that the UI statement should have a calls link to the trigger block. This can only be done via hints, and we will have to point to the trigger by its code location, beside the external program where is defined.

I was thinking that the call link should be from the EVENT node to the TRIGGER_BLOCK node. Those event definitions that are specific to a given widget will have only a limited number of places that could raise the event.

It seems to me that all downstream places that could raise the related EVENT should have a RAISES link to the nearest upstream instance of that EVENT. When I say upstream/downstream here I mean the specific call stack that includes the top-level code blocks. I realize that for EVENT types that are ANYWHERE, the need to go deeper in the call tree would be there.

The idea here is that one would traverse from something like APPLY or WAIT-FOR over RAISES to an EVENT and from there over CALLS to the TRIGGER_BLOCK. Conceptually this makes sense to a developer. The ON statement can have a LISTENS to the EVENT to show that relationship.

#130 - 06/02/2017 09:33 AM - Greg Shah

Constantin Asofiei wrote:

> Greg, what was the reason for the Configuration.forceHome(changes in RootNodeList.java rev 11225? With these changes, I can no longer run callgraph for my testcases/ setup... at least it should check if the path isn't starting with the home, before appending it.

When loading the rootlist and/or graph database instance from the report server, the current directory is not the same as project home. We have several places in the project (rootlist was one of them) that was implicitly defined relative to the project home (./cfg/rootlist.xml). It could not be loaded from the report server, so I implemented forceHome().

#131 - 06/02/2017 11:25 AM - Greg Shah

I like the reports approach. I will provide a different way to select them. I'm adding some other UI controls to the screen now so that the user can control what is loaded in the graph. One thing that is not clear is what code sets the this.hash member for the menuHandler() to use. Perhaps that is something in the bootstrap framework?

Other things:

- When you click on the call site, we will want it to load the Source/AST view. We will need the AST node id and will have to call some function (of Eric's) that loads down the tree for that file and opens the source/AST view at the given node.
- We probably need a way to get back to the call graph without reloading it.

#132 - 06/02/2017 11:36 AM - Greg Shah

Actually, upon thinking about this, I think we should have a separate menu item. Instead of "Call Graph" we should have "Call Graph Visualization" and "Call Graph Reports". Right now there is no short term plan for integration from the visualization to the reports. It is possible to move the other way and may even be pretty useful. But for now, let's separate this.

#133 - 06/02/2017 03:09 PM - Constantin Asofiei

Greg Shah wrote:

> I like the reports approach. I will provide a different way to select them. I'm adding some other UI controls to the screen now so that the user can control what is loaded in the graph. One thing that is not clear is what code sets the this.hash member for the menuHandler() to use. Perhaps that is something in the bootstrap framework?

Yes, I think that is something in the framework...

- When you click on the call site, we will want it to load the Source/AST view. We will need the AST node id and will have to call some function (of Eric's) that loads down the tree for that file and opens the source/AST view at the given node.

Yes, I'm using Eric's code and it works for some call-sites, but not for others. There's something wrong either in my RPT DB or how I determine the file_map.id for a given filename...

- We probably need a way to get back to the call graph without reloading it.

Even if report is in its own menu, after the AST view we need to allow back to the report. Eric's reports have the same issue.

#134 - 06/02/2017 03:40 PM - Greg Shah

> Even if report is in its own menu, after the AST view we need to allow back to the report. Eric's reports have the same issue.

For now, clicking on the top level menu is enough.

#135 - 06/02/2017 03:55 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Yes, I'm using Eric's code and it works for some call-sites, but not for others. There's something wrong either in my RPT DB or how I determine the file_map.id for a given filename...

Right now, this view doesn't work for schema files. It only works for ABL source code files. This is a known problem. If you're seeing this for ABL files also, read on...

I see you added a static method in ReportApi to get a fid by filename. I didn't really intend it to be used this way, but I don't see a reason it shouldn't work, if the filename is right. Things would be going wrong there, though, if the filename is not exactly as it is stored in the database (it originally comes from the filename stored in the AST).

It would be safer to query the fid using the AST ID of the root of the AST, if that is easily available from the place(s) you need to get the fid.

#136 - 06/02/2017 09:03 PM - Greg Shah

I see some improvements in the graph, though it is still pretty badly broken when all 3 root nodes are loaded. When only start.p is loaded, the problems are much smaller.

At this point I think the dynamic loading of the graph (item 7 below) will have to be deferred. Here are the things that are most important:

1. Finish the CG reports and have them moved to their own tab.

2. Fix the core issues with the graph such that it renders in a way that can be shown to a customer. There are still nodes appearing as MORE which are ON or TRIGGER_BLOCK. The containment is working for start.p but seems broken for the other root nodes.

3. Improve the graph visualization so that the nodes and links are displayed better, with more useful information, shapes, colors... For example, the circle rendering is still broken and many node and link types don't have any color mapping. We need to enhance some of the information displayed for some nodes too.

4. Display the AST details popup when hovering an AST_NODE. This will use Eric's implementation in fwd.ast.showInfoPanel(). We will have to send the node-id down as part of the graph.

5. Link to the source/AST view from the call graph visualization.

6. Traverse deeper in the graph by clicking on a MORE node. Properly reload the graph.

7. User control over the initial graph loading including starting points, direction and traversal levels. I;ve been working on this all day, but the UI is stuck.

If we had the above ready, we would be in good shape. Even if we have to defer items 6 and 7, if we can get the others done by Sunday night, then we are OK.

The conference starts Sunday night and we will be needing to demo to customers starting Monday morning. I also need to work on the presentation which I am giving on Wednesday morning at 9:45am. Having item 6 (and maybe 7) by then would greatly help.

I'll be working on 3, 4 and 5 tomorrow and Sunday.

Constantin: I'm pretty dependent on you for items 1 and 2. Do you think you are close enough to get those closed over the weekend?

#137 - 06/02/2017 09:06 PM - Constantin Asofiei

Greg Shah wrote:

> Constantin: I'm pretty dependent on you for items 1 and 2. Do you think you are close enough to get those closed over the weekend?

Greg, see revision 11253 - I've added ambiguous callsites and external targets reports.

I've moved the reports in a separate section, but the menubar needs more work - currently looks the same as the main menubar, but it should be smaller (and eventually to the right). Need to find how to fix it, if you can do it faster than me, please let me know.

Plan for tomorrow is:

- the dead code report - issue 1 in Greg's previous note
- finish hints for Hotel GUI (encode the event stuff) and other event-related stuff (MORE nodes) - issue 2 in Greg's previous note
- eventually a 'event' report (raised-event-calls-what-trigger).

#138 - 06/02/2017 09:30 PM - Greg Shah

> I've moved the reports in a separate section, but the menubar needs more work - currently looks the same as the main menubar, but it should be smaller (and eventually to the right). Need to find how to fix it, if you can do it faster than me, please let me know.

Sergey: Please look at this tomorrow.

#139 - 06/03/2017 12:57 AM - Eric Faulhaber

Greg Shah wrote:

> Actually, upon thinking about this, I think we should have a separate menu item. Instead of "Call Graph" we should have "Call Graph Visualization" and "Call Graph Reports".

This set of changes has messed up the overall application layout. There are two issues:

- The longer call graph menu item names in the navbar names have added 20px to the height of the navbar. In rev 11254, I made some changes to the CSS and changed the label "Call Graph Visualization" to "Visualize Call Graph" to get this 20px back.
- Using a navbar menu in the call graph reports tab adds 90px (110px before my changes) to the bottom of each screen. This is because the visibility of this element is set to hidden, which affects layout of the mainView div element, regardless of which view is active. Can we find a different mechanism to select the reports on this page?

#140 - 06/03/2017 07:04 AM - Sergey Ivanovskiy

I changed Call Graph Reports navigation menu to be a dropdown menu that is placed over reports views at the right top corner, committed revision 11255.

#141 - 06/03/2017 07:17 AM - Sergey Ivanovskiy

Found that these call graph reports are very heavy and loaded Firefox browser on my computer, I didn't investigated it but it can be "large data unsupported" that is a usual root cause for UI components.

#142 - 06/03/2017 08:50 AM - Sergey Ivanovskiy

I am working to replace global js module initialization with AMD initialization like it was done for the known demo project.

#143 - 06/03/2017 02:21 PM - Greg Shah

Revision 11260 adds the AST details info panel (no annotations right now) on hover over a node that is backed by an AST. It also adds a shift-click to load the source/ast view if the event is on a node that is backed by an AST.

I don't yet include any AST annotations in the display (they are not yet included in the CallSite instances).

Two issues:

1. Many nodes that seem like they should be ASTs are not acting like it. For example, KW_UPDATE, KW_WAIT_FOR... don't show AST details but the EXTERNAL_PROCEDURE and KW_APPLY will do so. I wonder if this is a factor with how the graph is being sent down.

2. Once the source view is loaded, you can't get back to the Visualize CG screen until you have loaded some different menu item.

#144 - 06/03/2017 03:28 PM - Constantin Asofiei

Greg, I've done the dead code report. I'm looking into the callgraph visualisation improvements next.

#145 - 06/03/2017 06:47 PM - Constantin Asofiei

Greg, I've been trying to solve these issues:

1. MORE nodes: the traverseLevel must create MORE nodes only if level 0 is reached and only to PROCEDURE_FILE or SCHEMA_FILE targets. With this kind of MORE implemented, clicking in the browser on such a node will mean that this non-expanded file will now be included as 'root-file' and the graph snippet can be re-generated with the additional file in its root-list (this is not implemented AFAIK, just an idea). Also, there was a problem with the CallGraphLink and CallGraphNode - they needed hashCode and equals to be able to use them in a set...
2. for 1-level deep, it gets us ~4000 nodes and ~11000 links. I think I can reduce the amount of data if I force other kind of edges only and only if that vertex has at least one incoming or outgoing CALLS, RAISES or LISTENS edge
3. non-overlapping root containers (i.e. procedure files) - I've been trying to use a collision detection way at least only for the root containers, but it doesn't work; I think it interferes with some 'snap-to-grid' code, but I can't find where is the problem

I'll commit the code for point 1 above, and continue with point 2, but point 3 is pretty tricky.

#146 - 06/03/2017 08:32 PM - Constantin Asofiei

Greg, I think I've managed to get the callgraph (a real call-graph, not code graph - if you want code, I assume are reports?) to a good representation - see revision 11264; what I've done:

1. reduced the number of nodes and links emitted; as we want to see a callgraph, there is no reason to see code which is never executed - so, from a procfile's entire code, I choose only the tree which has as leaves nodes with calls, raises, listens edges (executed code...).
2. MORE links are emitted file-to-file; if you leave them callsite-to-file, they are in ~4000 for 1-level deep
3. fixed level0 case: these edges are collected and post-processed after the initial snippet is computed. A major problem was if a assumed level0-edge on a DFS path might not be a level0 edge on another path... so, with this change, the callgraph snippet looks a lot better.

What it doesn't look nice is the layout: overlapping, etc... if you can give me some hints, I can take a look tomorrow, too.

The dead files report is finished, too.

#147 - 06/04/2017 12:26 AM - Greg Shah

Constantin: your changes in 11264 do make a huge improvement.

With rev 11265, I add the circle back in as a shape (for More and Event nodes) and there is some safety logic to resolve an issue with MORE nodes that are containers or are themseles contained (both cases are broken). If these cases are found, a FIXME log entry will be seen in the console.

BTW, this is the result even after your changes:

```
FIXME: Node id 1605824 is of type MORE but it is CONTAINED by another node (id 2146408). This should NEVER HAP
PEN!  callgraph.js:2283:16
FIXME: Node id 1056824 is of type MORE but it is CONTAINED by another node (id 1392832). This should NEVER HAP
PEN!  callgraph.js:2283:16
FIXME: Node id 1044672 is of type MORE but it is CONTAINED by another node (id 1392832). This should NEVER HAP
PEN!  callgraph.js:2283:16
FIXME: Node id 524408 is of type MORE but it is CONTAINED by another node (id 1392832). This should NEVER HAPP
EN!  callgraph.js:2283:16
FIXME: Node id 807128 is of type MORE but it is CONTAINED by another node (id 1392832). This should NEVER HAPP
EN!  callgraph.js:2283:16
FIXME: Node id 405608 is of type MORE but it is CONTAINED by another node (id 1392832). This should NEVER HAPP
EN!  callgraph.js:2283:16
FIXME: Node id 565400 is of type MORE but it is CONTAINED by another node (id 1392832). This should NEVER HAPP
EN!  callgraph.js:2283:16
FIXME: Node id 954560 is of type MORE but it is CONTAINED by another node (id 1392832). This should NEVER HAPP
EN!  callgraph.js:2283:16
FIXME: Node id 1044672 is of type MORE but it is CONTAINED by another node (id 1351728). This should NEVER HAP
PEN!  callgraph.js:2283:16
FIXME: Node id 524408 is of type MORE but it is CONTAINED by another node (id 1351728). This should NEVER HAPP
EN!  callgraph.js:2283:16
FIXME: Node id 807128 is of type MORE but it is CONTAINED by another node (id 1351728). This should NEVER HAPP
EN!  callgraph.js:2283:16
FIXME: Node id 954560 is of type MORE but it is CONTAINED by another node (id 1351728). This should NEVER HAPP
EN!  callgraph.js:2283:16
FIXME: Node id 405608 is of type MORE but it is CONTAINED by another node (id 1351728). This should NEVER HAPP
EN!  callgraph.js:2283:16
FIXME: Node id 565400 is of type MORE but it is CONTAINED by another node (id 1351728). This should NEVER HAPP
EN!  callgraph.js:2283:16
FIXME: Node id 1056824 is of type MORE but it is CONTAINED by another node (id 1351728). This should NEVER HAP
PEN!  callgraph.js:2283:16
```

The reverse case is no longer present in the recent build.

In addition, the layout of the top level SVG now handles the case where loading contained nodes needs more space than the viewport allows. With rev 11264 and these changes, the initial graph is much closer to a useful result.

I'll keep going in the morning.

#148 - 06/04/2017 07:34 AM - Greg Shah

I'm working on improving the appearance of nodes and links. I'm adding colors and so forth for all types than can be encountered.

Each node should display with enough information to tell us about what it is doing. For example, the target of a RUN statement should be included in the descriptive text so that viewing the node will make sense. Another example is that a DYNAMIC-FUNCTION call would be better understood by seeing its first parameter.

Is the addition of the node-key enough for this or do we need to augment the data (both stored in the graph and passed down in the graph snippet) to have a standard way to provide that data that can be used to improve the visual representation?

#149 - 06/04/2017 07:43 AM - Constantin Asofiei

Greg Shah wrote:

> Is the addition of the node-key enough for this or do we need to augment the data (both stored in the graph and passed down in the graph snippet) to have a standard way to provide that data that can be used to improve the visual representation?

node-key is not created for all vertex types, only for entry points and external targets. If you want a standard description, then I need to change all call-sites to collect info about their possible target(s).

#150 - 06/04/2017 07:46 AM - Greg Shah

Some other questions:

1. I don't see any "calls" link from an event to a trigger. This is a pretty crucial element of the event processing visualization.

2. There are many nodes without links. For example, in the initial display of ehotel.p, most of the internal procedures are unreferenced. If this is correct (perhaps they are called only via the appserver/js upcalls), then we probably need a way to tell the call graph that fact and to visualize it. It might affect dead code analysis and it certainly is confusing for the user.

3. I'm very excited about the addition of internal procedures and functions to the dead code analysis. How confident are you that the calculated results are correct?

4. In the dead code report, we may need a way to see the whole include filename. Many of them are currently being truncated. Perhaps a tooltip would work?

5. It is not clear why the include files are listed in the right column instead of the left, but that is not the biggest deal.

6. The dead procedure files list shows the filenames twice. Is there some meaning there?

#151 - 06/04/2017 08:11 AM - Constantin Asofiei

Greg Shah wrote:

> With rev 11265, I add the circle back in as a shape (for More and Event nodes) and there is some safety logic to resolve an issue with MORE nodes that are containers or are themseles contained (both cases are broken). If these cases are found, a FIXME log entry will be seen in the console.

MORE nodes should never be contained or containers. How do you determine that they should be contained or containers? Aren't you looking at the edge label - only contains should allow a node to contain another one.

#152 - 06/04/2017 08:25 AM - Constantin Asofiei

Greg Shah wrote:

> Some other questions:
>
> 1. I don't see any "calls" link from an event to a trigger. This is a pretty crucial element of the event processing visualization.

Yes, I need some post-processing done to be able to build these automatically.

> 2. There are many nodes without links. For example, in the initial display of ehotel.p, most of the internal procedures are unreferenced. If this is correct (perhaps they are called only via the appserver/js upcalls), then we probably need a way to tell the call graph that fact and to visualize it. It might affect dead code analysis and it certainly is confusing for the user.

If they are without links, it means the link was hidden via a MORE - the MORE links are from procedure file to another procedure file; if I change it to be from the callsite to the procedure file, there will be too many links.
But there was a problem: I was adding links of kind i.e. this-calls-more and not more-calls-this, which is possible, too. I'll commit this.

> 3. I'm very excited about the addition of internal procedures and functions to the dead code analysis. How confident are you that the calculated results are correct?

Currently, I'm doing a DFS search starting from the known roots (on all OUT edges), and collect the reached entry points (program file, internal procedure, function, trigger). After this, I get the list of all possible entry points and remove the reached ones - whatever is remaining must be dead.

I'll revisit this a little once I have the event call issue resolved.

> 4. In the dead code report, we may need a way to see the whole include filename. Many of them are currently being truncated. Perhaps a tooltip would work?

I'll look into it.

> 5. It is not clear why the include files are listed in the right column instead of the left, but that is not the biggest deal.

The report has two columns, to be able to show the program for a dead function, for example. But for include files and procedure files, only a column is needed... not sure how to make it look better.

> 6. The dead procedure files list shows the filenames twice. Is there some meaning there?

Same as point 5, I should make it look better but not sure how.

#153 - 06/04/2017 09:40 AM - Greg Shah

> MORE nodes should never be contained or containers. How do you determine that they should be contained or containers? Aren't you looking at the edge label - only contains should allow a node to contain another one.

I agree. I am just processing the "contains" links and looking at any source or target nodes that are MORE. See the code in createNodeMap().

#154 - 06/04/2017 10:20 AM - Greg Shah

> MORE nodes: the traverseLevel must create MORE nodes only if level 0 is reached and only to PROCEDURE_FILE or SCHEMA_FILE targets.

I agree that only links (either direction) that connect to something external should be included as MORE. I would prefer this to include other types of external resources (DLL, process, network, web service...), otherwise it may not be possible to ever reach them or see them in most of the graph viewing (since it is likely that there will be a level limit).

> With this kind of MORE implemented, clicking in the browser on such a node will mean that this non-expanded file will now be included as 'root-file' and the graph snippet can be re-generated with the additional file in its root-list (this is not implemented AFAIK, just an idea).

This is similar to the idea I had about traversal. The idea was that when clicked, more would reload the graph from the "perspective" of the MORE node. In effect this would replace the root list, with a new starting list of that specific file being clicked. Your idea is useful too. We can do both by differentiating the clicks (SHIFT-CLICK for one, CTRL-CLICK for the other).

> MORE links are emitted file-to-file; if you leave them callsite-to-file, they are in ~4000 for 1-level deep

This is definitely making a positive difference. The only problem is that we lose information in this approach. The user cannot see that there is a link between some visible node and the MORE.

How about if we mark the node with some kind of flag and a list of the links that are hidden (calls link to xyz, listens link to event w...). Then we can display the node differently or otherwise give the user some indication there is something missing.

We can also potentially load the missing links based on some user action.

> > Is the addition of the node-key enough for this or do we need to augment the data (both stored in the graph and passed down in the graph snippet) to have a standard way to provide that data that can be used to improve the visual representation?

> node-key is not created for all vertex types, only for entry points and external targets. If you want a standard description, then I need to change all call-sites to collect info about their possible target(s).

It seems to me that the graph already encodes this. We just need to gather it up and report it in a useful way.

Perhaps this can be done during graph snippet creation? Why not create a method that examines the outbound links from a call site and encodes a useful description. Since the entry points that are being called have a node-key encoded, perhaps this can be done generically and the result included in the node.

> > 3. I'm very excited about the addition of internal procedures and functions to the dead code analysis. How confident are you that the calculated results are correct?
>
> Currently, I'm doing a DFS search starting from the known roots (on all OUT edges), and collect the reached entry points (program file, internal procedure, function, trigger). After this, I get the list of all possible entry points and remove the reached ones - whatever is remaining must be dead.
>
> I'll revisit this a little once I have the event call issue resolved.

Let's get the visualization finished first. We probably also want to get the traversal working first too.

#155 - 06/04/2017 10:29 AM - Greg Shah

I think EVENT nodes should be contained in the scope (4GL code block) that registers them. Events are not really global at all, even ANYWHERE events. Only code that is deeper on the call stack can raise that event. And more, the event itself will only call the trigger that is in that same scope. The current uncontained event is not modeling this properly.

#156 - 06/04/2017 12:35 PM - Constantin Asofiei

Greg Shah wrote:

> I think EVENT nodes should be contained in the scope (4GL code block) that registers them. Events are not really global at all, even ANYWHERE events. Only code that is deeper on the call stack can raise that event. And more, the event itself will only call the trigger that is in that same scope. The current uncontained event is not modeling this properly.

What if same EVENT name is used in multiple programs? The problem here is that we have runtime behaviour which can't be really simulated in the callgraph: for example, a trigger can be in any persistent procedure which is still alive, and we can't easily determine which one will be targeted by a certain raised event.

#157 - 06/04/2017 12:40 PM - Greg Shah

The first one matched in the current scope is used. So it seems like we could walk up the call graph and look for scopes that contain the matching event.

Are the persistent procedures checked only if we get to the root of the call tree? These would need hints I guess.

For now, I'm looking to get a good initial implementation. If events matching in persistent procedures is not implemented (or if it requires hints which we don't have time to encode for Hotel) that is OK.

#158 - 06/04/2017 12:42 PM - Constantin Asofiei

Greg Shah wrote:

> MORE nodes should never be contained or containers. How do you determine that they should be contained or containers? Aren't you looking at the edge label - only contains should allow a node to contain another one.

I agree. I am just processing the "contains" links and looking at any source or target nodes that are MORE. See the code in createNodeMap().

The links reported by createNodeMap are calls links, not contains... so I'm not sure where the problem is.

#159 - 06/04/2017 12:46 PM - Greg Shah

> The links reported by createNodeMap are calls links, not contains... so I'm not sure where the problem is.

You're right. I needed to only check contains links. There are no additional issues there.

#160 - 06/04/2017 01:33 PM - Greg Shah

Revision 11268 has text labels added to the middle of every link. This should make it much easier to understand the relationships. I hope it doesn't

clutter up the display too much.

Next, I'll add display definitions (color, stroke and stroke-dashing) for the nodes not yet handled.

#161 - 06/04/2017 03:10 PM - Greg Shah

11269 has improved displayProps.

#162 - 06/04/2017 03:32 PM - Constantin Asofiei

Greg, the approach with the events is this:

1. EVENT nodes are created in each context (internal proc, function, external procedure). These are circle and, although I have a 'contains' node from the context to the EVENT, they are not 'contained' in the UI - can you make them contained?
2. listens edges are created from each trigger to the local-context EVENT
3. raises edges are created from each UI stmt to the local-context EVENT
4. EVENT nodes for UI stmt are created in local-context only and only if there is a trigger (in current context or a reachable one) which is executed for that event. So, there will be UI_stmt -RAISES-> EVENT -CALLS-> TRIGGER_BLOCK <-registers- KW_ON -LISTENS-> EVENT. Note that the two EVENT nodes (although with the same event name) may not be in the current context; same for TRIGGER_BLOCK and UI statement.
5. how is a trigger determined to be reachable: always keep first found from, starting with current context:

- currently processed context
- contexts for RUN ... PERSISTENT external program targets which are executed from current context
- up the call-stack (the context for the entry-point which calls currently processed context)

Does this make sense?

#163 - 06/04/2017 03:58 PM - Greg Shah

> can you make them contained?

I can look at it. It has other ramifications but I should be close.

> Note that the two EVENT nodes (although with the same event name) may not be in the current context; same for TRIGGER_BLOCK and UI statement.

I assume that if everything is in the same context there would only be one event node?

> Does this make sense?

I think so.

#164 - 06/04/2017 03:59 PM - Constantin Asofiei

Greg Shah wrote:

> can you make them contained?

I can look at it. It has other ramifications but I should be close.

Although the graph looks OK, the computed snippet has nodes which should be MORE but instead are i.e. TRIGGER_BLOCK from a supposedly MORE external program. After I fix this, I'll commit.

> Note that the two EVENT nodes (although with the same event name) may not be in the current context; same for TRIGGER_BLOCK and UI statement.

I assume that if everything is in the same context there would only be one event node?

Correct.

#165 - 06/04/2017 04:23 PM - Greg Shah

Revision 11270 makes improvements in node text and some tweaks for reducing the space allocated for containers.

After the event improvements are in, please look at:

- the call-site better description problem
- the user can't see that some nodes are linked/hidden more issue

Are there any other critical/core graph problems on the initial display?

My priorities for tonight:

- fix the circles can be contained issue
- shrink the space used by containers (tricky)
- better distribution for non-contained non-container nodes (e.g. more nodes) in the outer svg

#166 - 06/04/2017 05:32 PM - Constantin Asofiei

Greg Shah wrote:
See 1514a revision 11271 for event improvements.

> After the event improvements are in, please look at:
>
> - the call-site better description problem
> - the user can't see that some nodes are linked/hidden more issue

OK

> Are there any other critical/core graph problems on the initial display?

No.

#167 - 06/04/2017 09:23 PM - Constantin Asofiei

Greg, see 1514a revision 11273 for improvements related to the description/tooltip of the call sites - I'm including the names of all targets for a call site, for function, procedure or event calls, so you can see them in the tooltip.

About the MORE nodes: I'm not sure how to solve this; maybe I can enable it only for level > 0?

#168 - 06/05/2017 05:19 AM - Greg Shah

> About the MORE nodes: I'm not sure how to solve this; maybe I can enable it only for level > 0?

I'm not sure. If we do enable this "hide external links" mode, it is very misleading. Much of the code looks disconnected/dead. I have to hover each one to see that it actually connects to something external that is off screen. The entire point of making it visual is lost.

How many procedure_file nodes are there in the level == 1 case? Is it correct that there 4000 nodes/11000 links or are there more nodes/links because we have no hints defined for the project?

One idea:

- send all the links down with the graph snippet
- draw nodes with more links differently (e.g. perhaps with a marker that is an attached star or circle that indicates a hidden link)
- click that marker to expand the link

#169 - 06/05/2017 05:48 AM - Constantin Asofiei

Greg Shah wrote:

> How many procedure_file nodes are there in the level == 1 case? Is it correct that there 4000 nodes/11000 links or are there more nodes/links because we have no hints defined for the project?

procedure_file nodes is limited to 45 files (number of programs in Hotel GUI). The problem is that we have lots of calls to internal procedures/functions/triggers, between programs. With level 1, now I get 2384 nodes and 6673 links. Still some cluttering, but it looks a little better than before. With file-to-file MORE nodes, there are 4476 links.

All this is now done automatically, and the ADM/ADM2 has the most links. I'll try to think of a way to better disambiguate.

> One idea:
>
> - send all the links down with the graph snippet
> - draw nodes with more links differently (e.g. perhaps with a marker that is an attached star or circle that indicates a hidden link)
> - click that marker to expand the link

Hmm.. I think this will look good. And we can go even further than LEVEL 0:

1. for dependencies between external-program, by default use a dependency link between them
2. if you want to dig deeper, use your approach to add the links for a certain node, on demand. Your marker will mean the node has dependencies to another external program and clicking on the marker will show these dependencies explicitly.
3. once a file-to-file link has been 'resolved' (all marker nodes are expanded), then this link can be removed.

This way, the user decides how deep to explore the graph. Also, we can have a way of 'expand all' in the UI.

#170 - 06/05/2017 05:48 AM - Greg Shah

> I'm including the names of all targets for a call site, for function, procedure or event calls, so you can see them in the tooltip.

Many call sites only have 1 target. If there is only 1 target, I want to include this in the displayed node label rather than just the tooltip. If there are multiple targets, then we can give some visible indication of this and let the user use the tooltip to get the full list.

To get this working, we need some consistent way to render the node text and tooltip text. In callgraph.js, both calcNodeText and calcNodeTooltip have hacks to try to deal with the inconsistency of detail in nodes.

We have 4 cases:

- MORE nodes that should display a label of "More" but have good of details in the tooltip
- nodes with a symbolic token name for type that is the same as their text node, in this case we override the label with the more descriptive displayProps[d.type].text but the tooltip is pretty useless, TRIGGER_BLOCK is an example; if we can come up with anything more useful it would be good
- nodes with useful text, these are generally entry points like a FUNCTION which has a name; I've marked such node types with displayProps[d.type].augmentText and we add some useful prefix text to display a better label
- nodes (call sites, mostly) where the symbolic token name for type does not match the token text, KW_RUN vs RUN_INT_PROC is an example; in this case we just display the text, but it is not very useful; I'd really prefer to make the text set to the target here when possible and use the augmentText flag to get a better description

#171 - 06/05/2017 05:54 AM - Greg Shah

> for dependencies between external-program, by default use a dependency link between them

What do you mean by "dependency link"? Is this the "single file to file more"?

One thing that I would like to consider is a heuristic, probably based on a simple threshold, that detects when we should default to showing everything insted of redacting. I like the idea of a UI setting to control the redaction. But it seems to me that below a certain number of links, showing everything is probably OK.

#172 - 06/05/2017 06:05 AM - Constantin Asofiei

Greg Shah wrote:

> > for dependencies between external-program, by default use a dependency link between them
>
> What do you mean by "dependency link"? Is this the "single file to file more"?

I mean, even if both programs are expanded in the current snippet, then we don't create a link for each call between them, just a single link linking the two external programs. The marker at the node will allow the user to show the redacted links. This can reduce the cluttering even more, as is hard to follow a path if there are thousands of links in the UI.

> One thing that I would like to consider is a heuristic, probably based on a simple threshold, that detects when we should default to showing everything insted of redacting. I like the idea of a UI setting to control the redaction. But it seems to me that below a certain number of links, showing everything is probably OK.

Maybe some kind of 'weight' computed using the number of call-sites in an external program and the percentage of links to other programs? Anyway,

I'll think about it.

#173 - 06/05/2017 06:13 AM - Greg Shah

Another thing to consider is that instead of loading entire root list, a user might want to load just a single program. That is what I was working on with the combo box (in the right side toolbox). The idea was to do this:

- enter the "starting point" for the graph to load; choose between "Root List", one of the existing File Lists (Eric calls them "filter profiles" but I'm not a fan of the term) or enter a program name manually (e.g. typing ./abl/start.p.ast)
- select DOWNSTREAM, UPSTREAM or BOTH (BOTH is defaulted)
- select the number of levels, 0 or more (0 is defaulted)
- press "Load Graph" to get started

Over time this might be enhanced to select from some filtering options (e.g. check boxes to control things like whether include files are displayed or whether to include event processing). I haven't figured this part out yet. To some degree we need to keep working our way through this to see what is most useful.

One thing I'm pretty sure of: we will want to implement some form of "summarization level". The idea: the user can choose between 3 levels of collapsed/expansion: FULL, CODEBLOCK, DOMAIN

FULL means there is no summarization, all nodes are rendered at full detail.

CODEBLOCK means that nodes contained in a code block (entry points like internal procedures, functions...) are hidden and only the code block appears. All links are still there except when there now appear to be multiple links between this summarized node and another node, then that link is summarized and the link tooltip may give some more details, Links of different types wouldn't work well in this mode.

DOMAIN means the summarization is at the file or resource level. This is the same idea as CODEBLOCK, but summarized 1 level "higher".

I also have wanted to implement a collapse/expand that can be implemented by the user on a specific container node. For example, CTRL-doubleclick on a node would collapse or expand it on the fly. This would allow the user to implement this same summarization, but dynamically and on specific sections of the graph. I've mentioned this above in "collapse/expand" as a feature and http://mbostock.github.io/d3/talk/20111116/force-collapsible.html is an example of the idea visually.

#174 - 06/05/2017 06:16 AM - Greg Shah

> I mean, even if both programs are expanded in the current snippet, then we don't create a link for each call between them, just a single link linking the two external programs. The marker at the node will allow the user to show the redacted links. This can reduce the cluttering even more, as is hard to follow a path if there are thousands of links in the UI.

Yes, this could work. We might also want to draw the link differently. The width, stroke-dashing, color and label of the link probably needs to be made unambiguous. For example, the width of the link can be 4 or 6 pixels instead of 2, we could have a different ("Aggregated Links") label and unique color.

Maybe some kind of 'weight' computed using the number of call-sites in an external program and the percentage of links to other programs?

This has promise.

#175 - 06/05/2017 11:26 AM - Constantin Asofiei

Greg, I think I found the cause of the NaN's:

1. in rectIntersection, if outside.x == inside.x, then the slope is undefined and this case needs to be treated explicitly
2. I can't find where the cx and cy attributes are set for circle nodes; in circleIntersection, I've changed to get them from target.bounds, but I think there are cases where cx and cy attributes are needed...

#176 - 06/05/2017 02:01 PM - Greg Shah

Code Review Task Branch 1514a Revision 11276

Everything is good except for the changes on line 2117 and 2132. In both cases you have removed the cx/cy references that are needed.

2117 old
:

```
                var sourcePoint = { "x": bounds.cx, "y": bounds.cy };
```

2117 new:

```
                var sourcePoint = { "x": bounds.x, "y": bounds.y };
```

This is the reason that the links to and from rectangles are now attached to the interior instead of the edge.

I also agree that the radius of the circles needs to be added in to both the cx and cy locations so that they snap with their full boundaries inside the container.

#177 - 06/05/2017 02:11 PM - Constantin Asofiei

Greg Shah wrote:

> Code Review Task Branch 1514a Revision 11276
> This is the reason that the links to and from rectangles are now attached to the interior instead of the edge.

Thanks, I've fixed it, now the links are back to normal.

> I also agree that the radius of the circles needs to be added in to both the cx and cy locations so that they snap with their full boundaries inside the container.

Do you have a hint where this is computed?

#178 - 06/05/2017 03:34 PM - Constantin Asofiei

Greg, 1514a rev 11277 has the custom graph loading from note #3277-173 . I'm moving to MORE node expansion next.

#179 - 06/05/2017 04:22 PM - Constantin Asofiei

Greg, 1514a rev 11279 adds SHIFT/CTRL-click on a MORE node which will either load the graph with the program as root or add it to the root list and re-load.

I'll think about the 'hidden MORE link improvements' next.

#180 - 06/06/2017 10:15 AM - Greg Shah

Revision 11281 adds a calculation of the actual minimum width and height of each container. This is the basis for reworking the oversized containers to a much more compact visualization. That rework is not done yet.

The idea is this:

Typically, most programs have a large external procedure container and then many much smaller nodes. The current approach makes a uniform cell size that can handle the largest node. This is way too much usage of space. The new d.minWidth and d.minHeight (where d is the container node), sums the width/height of the largest elements to determine the worst case scenario. In a 3x2 grid with 6 elements, the largest 3 widths would be summed and the largest 2 heights would be summed. This means that we are allocating enough space for the worst case in each dimension. The code that does this is in calcGrid().

The next step is to implement a non-uniform cell size in both the element bounds calculation and for snap-points.

#181 - 06/06/2017 10:28 AM - Greg Shah

> Do you have a hint where this is computed?

This is calculated in calcElementJail() and enforced by the code in enforceBoundingBox(). I suspect that the adjX and adjY processing is not right for circles. The idea there is that we are translating the contained element's coordinates relative to the container for the purposes of the jail calculations and then we need to adjust the position back to the correct relative spot after the decision is made. That is the adjX and adjY purpose. Since we are subtracting the radius from itemX and itemY in calcElementJail() we need to add it back in on the adjustment, I think.

#182 - 06/06/2017 03:08 PM - Constantin Asofiei

Greg, just a quick status: my approach is to 'compact' the rows/columns on the minimum element height/width, from each row/column cell, in ticked. This somehow works, but I don't have it yet stabilized.

For now, I think the approach will be OK (especially that the snap points swap will be allowed). I just need to get it to re-compute the grid after each snap swap.

#183 - 06/06/2017 04:56 PM - Greg Shah

Can MESSAGE and PAUSE really generate the full set of possible user-generated events?

#184 - 06/06/2017 07:57 PM - Constantin Asofiei

Greg Shah wrote:

> Can MESSAGE and PAUSE really generate the full set of possible user-generated events?

Hmm... actually, as I recall, they don't. I'll remove them.

For the container size: it mostly works, but there are cases where the jails are outside of the prison, the contained elements are leaking outside of the container.

#185 - 06/06/2017 10:55 PM - Greg Shah

> For the container size: it mostly works, but there are cases where the jails are outside of the prison, the contained elements are leaking outside of the container.

If it looks better than the current approach, you can check it in. If a user would think it is "more broken" than the current code, then hold off.

What Ive been doing in demos is showing start.p by itself because it doesn't have the size issue. If that looks OK and the others look better, then it may work. I'll explain that that part of the system is still being actively developed.

#186 - 06/06/2017 10:59 PM - Constantin Asofiei

Greg Shah wrote:

> For the container size: it mostly works, but there are cases where the jails are outside of the prison, the contained elements are leaking outside of the container.

> If it looks better than the current approach, you can check it in. If a user would think it is "more broken" than the current code, then hold off.
>
> What Ive been doing in demos is showing start.p by itself because it doesn't have the size issue. If that looks OK and the others look better, then it may work. I'll explain that that part of the system is still being actively developed.

There's only one issue left which I don't understand; how is the matrix computed for the d.contained nodes, via the snapIndex? Because an element with snapIndex 10 should be on col 2 row 1, but is somewhere on col 4... this is not in start.p, is in ehotel.p, but it appears in others too, in certain circumstances.

#187 - 06/06/2017 11:08 PM - Greg Shah

> how is the matrix computed for the d.contained nodes, via the snapIndex? Because an element with snapIndex 10 should be on col 2 row 1, but is somewhere on col 4... this is not in start.p, is in ehotel.p, but it appears in others too, in certain circumstances.

It is defined in the d.container.snapPoints array made in createSnapPoints(). Then we use the snapIndex as an array index to get the calculated point. We then use that point in all the snap point processing.

#188 - 06/06/2017 11:14 PM - Constantin Asofiei

Greg, in revision 11284 is the current approach, with start.p looking OK. My idea was to have a static grid (determined by the snappoints) and compute the min width/height for each column and row, thus adjusting the grid to occupy a minimum area.

If you want to disable it, add a if (true) return; in callgraph.js calcMinGrid:1204.

#189 - 06/07/2017 12:39 AM - Greg Shah

Overall, the changes seem to make things better. Any further improvements will be appreciated. The start.p external proc does look better, but the procedure_file is too big.

If you can clean up the message/pause events, it would help.

The session is at 9:45am GMT-5. I can take changes up to 9am, if they are stable/safe.

#190 - 06/07/2017 05:15 AM - Constantin Asofiei

Greg Shah wrote:

> Overall, the changes seem to make things better. Any further improvements will be appreciated. The start.p external proc does look better, but the procedure_file is too big.
>
> If you can clean up the message/pause events, it would help.

See rev 11285 for MESSAGE and PAUSE plus a small improvement in the width/height of the container nodes.

> The session is at 9:45am GMT-5. I can take changes up to 9am, if they are stable/safe.

I'll try to solve the remainder issue, if I find the cause and is safe, I'll commit it.

#191 - 06/07/2017 08:51 AM - Constantin Asofiei

Greg, in the end the problem was something stupid, I was computing wrong index of the matrix element, in the snapPoints array... instead of multiplying row with columns, I was multiplying row with row.

See revision 11288 - now you can even go level 1 from start.p and it looks nice.

#192 - 06/08/2017 03:20 PM - Greg Shah

I've been thinking about the "hidden more" problem:

- I know that you have greatly reduced the number of links from the event processing. Have you tried making the hidden more links explicit again to see the result?
- We should place a checkbox on the toolbox to control display of these hidden more links.
- I'm thinking that when hidden, we can display a star on each node and hovering the mouse over the star could make the link appear temporarily (or clicking the star could make the link appear permanently).

- If there are multiple hidden links to the same more node, then we could draw the file to file link that that MORE node using a bigger stroke width to show that it is multiple links aggregated together.

#193 - 06/09/2017 07:27 PM - Constantin Asofiei

Greg Shah wrote:

> I've been thinking about the "hidden more" problem:
>
> - I know that you have greatly reduced the number of links from the event processing. Have you tried making the hidden more links explicit again to see the result?
> - We should place a checkbox on the toolbox to control display of these hidden more links.
> - I'm thinking that when hidden, we can display a star on each node and hovering the mouse over the star could make the link appear temporarily (or clicking the star could make the link appear permanently).
> - If there are multiple hidden links to the same more node, then we could draw the file to file link that that MORE node using a bigger stroke width to show that it is multiple links aggregated together.

All these are in 1514a rev 11293. Please review/check.

I chose to just change the visibility of the inter-program links; I think this is the better way, as this will not force the re-layout of the graph (as the links are already there...).

BTW, I think the MORE nodes should attract to each other (maybe in a circle pattern), towards the 'center of mass' of the graph... do we have an explicit force for these nodes now?

#194 - 06/16/2017 01:15 PM - Greg Shah

Constantin: Can you see if you can resolve the .pphints issue recently seen in the customer application? We will be meeting with the customer on Monday to setup the analytics tools with them. If call graph can be run on their project, it would be good. If not, we will provide it later.

#195 - 06/19/2017 04:03 PM - Greg Shah

Does the following correctly explain the status of our call graph v3 report replacements?

View Program Dependencies - does not exist in v3
Missing External Programs - replaced by Missing Targets report (new version is a superset of the old report)
Dead Files - replaced by Dead Code report (new version is a superset of the old report)
External Targets List - does not exist in v3 (External Dependencies seems to be different)
Ambiguous Programs - replaced by Ambiguous Call Sites report (new version is a superset of the old report)
Schema Trigger Procedures - does not exist in v3

#196 - 06/19/2017 04:10 PM - Constantin Asofiei

Greg Shah wrote:

> Does the following correctly explain the status of our call graph v3 report replacements?
>
> View Program Dependencies - does not exist in v3

Correct, will look how it can be added.

> Missing External Programs - replaced by Missing Targets report (new version is a superset of the old report)

Correct

> Dead Files - replaced by Dead Code report (new version is a superset of the old report)

Correct

> External Targets List - does not exist in v3 (External Dependencies seems to be different)

External targets was printing all external vertices; I've changed it to not include the missing vertices (which can only be assumed 4GL code), and can be seen in the Missing Targets report.

> Ambiguous Programs - replaced by Ambiguous Call Sites report (new version is a superset of the old report)

Correct

> Schema Trigger Procedures - does not exist in v3

Will add it.

#197 - 06/20/2017 01:38 PM - Greg Shah

This is the list of features that were planned but are not yet implemented.

1. ~~Ensure that the call graph processing (visualization and reports) will work in a multi-user mode.~~
2. ~~Ensure that the file-list loading works.~~
3. ~~Make a proper legend. The current legend is a starting point, but it is missing:~~
   - ~~A better layout that won't scroll off the screen.~~
   - ~~Entries that show the different link types.~~
   - A heading.
4. Better layout and processing for More nodes.
   - ~~If we currently have multiple More nodes that can refer to the same off screen resource, then these should we represented as a single More node that is connected from multiple places. I'm not sure if this is an issue.~~
   - ~~I would like to see the More nodes on the outside of the graph, not in the middle. The easiest way is to distribute them around the outside of the graph after the container jail sizing and assignment is done. They would still have snap points created and assigned. The idea is that we can evenly distribute them and it should work pretty well since the sizing should not be an issue. If we even enable the swap snap point processing for them, they can naturally move to a perimeter location where they most naturally should exist because of the links involved.~~
   - Modify the More node location distribution to spread them out evenly. These should not use the same jail cell size used for the larger nodes, rather they should be evenly distributed in a rectangle surrounding the container nodes.
5. BUG: Load from file list can leave everything off-screen right now, but if you pan/zoom you see it.
6. I would like to consider a different drawing method for the "thick" links. Perhaps a line that has a thin stroke and some fill would look good. Or a triple or double thin line might work.
7. Improve sizing and initial positioning.
   - Implement different sized cell sizes both at the top level and in container nodes. See #3277-180 for the idea. Constantin's approach was to compact the same somewhat, but the core issue is still there.
   - The largest nodes can be put in the same column or row, based on their largest dimensions. In most cases, at the top level there are some really wide containers that tend to get put side by side, making a really inefficient use of the display and placing much of it off screen. Put these in the same column would make it much better.
8. Add file list support for the CG reports.
9. Provide summarization levels for contained nodes. This would allow containers to be displayed without the contained nodes, this can occur to multiple levels of "collapse" all the way up to the domain resource level. The user would have the ability to collapse and expand specific nodes or to do it for the currently visualized graph snippet. The idea is allow a simpler view of a much larger portion of the graph, with the links still showing the relationships. Then the user can expand parts or all of the graph to see more details.
10. Use the D3 "brush" to implement a kind of magnifier tool that allows a zoomed out graph to have details viewed without having to zoom in.
11. Click on a node and it highlights that node, its immediate links and the nodes on the other end of each link. This could be done by greying out everything else in the graph. The idea is that this helps the user visually isolate the immediate portions of the graph connected to a given node.
12. Add annotations to the AST node details display.
13. Implement complete Hotel GUI callgraph hints.
14. Implement a search capability where you specify criteria and if something is found, it gets added to the snippet or replaces the snippet.
15. Implement filtering where criteria can be specified that includes or excludes matches from the visualized snippet.
16. Provide a UI to edit the root list and to add/delete/edit hints. This would need to allow the re-generation of the graph from the UI as well.

I think we might want to finish items 1 through 5, but I'm looking for feedback on your opinion on the other items. In your work of getting the call graph encoded for an application, which of these items (or other items) are essential or greatly valuable such that the visualization can really help you review your work and get the call graph to be correct?

LE: Items 1 through 4 were resolved well enough for release. The rest of this work will happen in a different task.

#198 - 06/29/2017 06:56 PM - Constantin Asofiei

Sergey, do you have other changes for the File List section? Because currently is not functionally complete.

#199 - 06/29/2017 07:08 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Sergey, do you have other changes for the File List section? Because currently is not functionally complete.

Constantin, I have taken over this feature, but I haven't gotten to it yet.

#200 - 07/17/2017 05:59 AM - Constantin Asofiei

Greg, 1514a revision 11353 adds the dependencies report and issues 1,3,4 from #3277-197 . There is an issue where swapping the snap points goes in a somehow 'infinite loop' with the same two adjacent cells, was trying to solve it but didn't get a good result.

#201 - 07/17/2017 06:45 AM - Greg Shah

After Eric completes the file list support, we can check that it works. Are the reports in good shape at this point?

The only other critical thing here is that you found an issue with running callgraph in the #3297. Is this still an issue?

#202 - 07/17/2017 07:01 AM - Constantin Asofiei

Greg Shah wrote:

> Are the reports in good shape at this point?

Yes.

> The only other critical thing here is that you found an issue with running callgraph in the #3297. Is this still an issue?

No, it was fixed. The problem was that a .p file was used as an include file, too.

#203 - 07/26/2017 11:53 AM - Greg Shah

> After Eric completes the file list support, we can check that it works.

File list support is now included in 1514a. I did a quick check and it is not working for call graph loading.

Would you please fix this?

One other question: have you tested multi-user support (simultaneous access to the CG from 2 or more users)? If it does not work. we need to document it. We don't have to fix it right now, though it might be useful to at least block access and notify the user that it cannot be accessed while another user is in it.

#204 - 07/28/2017 09:01 AM - Constantin Asofiei

Greg Shah wrote:

> After Eric completes the file list support, we can check that it works.

> File list support is now included in 1514a. I did a quick check and it is not working for call graph loading.
>
> Would you please fix this?

OK, working on it, should be easy to fix.

> One other question: have you tested multi-user support (simultaneous access to the CG from 2 or more users)? If it does not work. we need to document it. We don't have to fix it right now, though it might be useful to at least block access and notify the user that it cannot be accessed while another user is in it.

Yes, I've checked multi-user, and the graph DB (for the callgraph drawing and reports) was working properly.

#205 - 07/28/2017 02:34 PM - Greg Shah

Load from file now works in 1514a revision 11395.

#206 - 10/13/2017 08:09 AM - Greg Shah

- Related to Feature #3359: enhance the callgraph to better support programs outside of the current application added

#207 - 01/22/2018 03:27 PM - Greg Shah

Interesting article on TinkerPop3:

http://www.zdnet.com/article/from-graph-to-the-world-pioneering-a-database-virtual-machine/

#208 - 07/19/2018 05:14 PM - Greg Shah

I am thinking about ways to greatly reduce the time needed to get the call graph hints defined. This has surfaced with multiple clients as a blocker because:

- It takes so long to do that it is hard to get development resources allocated.
- It is error prone because it is a completely manual process.
- It is very iterative which extends the overall time frame.

We have considered providing a UI in the Analytics web application that would allow editing of hints. This would reduce syntax errors but it is still a manual process and it does nothing to avoid errors due to omission of call linkages that are needed OR inclusion of call linkages that should not be there. We will build this at some point, but I think we need something more.

I'm thinking that we could implement code that would instrument the 4GL code for an entire application, such that the application can be executed and the actual control flow/call processing of the executed code would save off hints that could be directly used by FWD call graph processing.

- We would probably write some 4GL helper code that would be in an include file.
- Every program would have this included at the beginning.
- We could instrument every indirect call site to capture hints (where possible at the moment of execution). For example, a RUN VALUE(expr) knows the result of the VALUE() in the caller. We would also potentially know the specific hint name to be encoded at this point.
- We could instrument every top level block to capture the linkage to a specific named target for those cases where we cannot know the target until it is invoked.
- The resulting hints should be encoded as a set of <program_name>.hints files that can be easily captured and placed into the FWD project.

The primary limitation of this approach is that it is only as good as the scenarios executed. If the overhead isn't too bad, this could even be executed in production. Worst case, it can be executed in testing.

Constantin: What are your thoughts on this? Do you have any other ideas of how to solve this problem?

#209 - 07/19/2018 05:24 PM - Eric Faulhaber

Would it make sense for the include file to conditionally preprocess in the instrumentation code? That way, the customer could create a build that would have zero impact/risk on production, if they only wanted to run the instrumented code in non-production environments. Also, this would allow one to easily disable instrumentation once the data needed for call graph was collected, in the case this is not an ongoing part of the development cycle.

#210 - 07/19/2018 07:05 PM - Constantin Asofiei

Greg, the idea is interesting. I assume you mean to be able to generate the hints for an app not being run on FWD? If so, I don't see how we can intercept the i.e. RUN statement at 4GL-level, without rewriting the 4GL legacy code. We could do some automated preprocessing in FWD, where we rework the callsites so that the expression (target) is computed and saved in a hint, before executing the call, but we need to ensure this is done only once (to avoid side-effects). And this instrumented 4GL app would be ran from our .cache files, and not from the legacy files.

If you mean to generate the hints with an already converted app running on FWD, this can be done easier directly in FWD, by enhancing ControlFlowOps and other code. But this would require the app to already parse, convert and run in FWD.

#211 - 07/19/2018 07:12 PM - Constantin Asofiei

Also, I don't see how we can compute the hint ID at runtime; the only way to do this is at preprocessing time, where we can count these call sites and mark them with the hint ID.

#212 - 07/19/2018 10:44 PM - Greg Shah

> I assume you mean to be able to generate the hints for an app not being run on FWD?

Yes, in 4GL.

> If so, I don't see how we can intercept the i.e. RUN statement at 4GL-level, without rewriting the 4GL legacy code.

Yes, we wrapper the RUN in some helper function (from the include) that does the real RUN.

> We could do some automated preprocessing in FWD, where we rework the callsites so that the expression (target) is computed and saved in a hint, before executing the call, but we need to ensure this is done only once (to avoid side-effects).

The idea was to parse everything and generate some list of hint definitions that would be edited (instrumented) in the 4GL. We edit the code and then run the result in the 4GL.

> And this instrumented 4GL app would be ran from our .cache files, and not from the legacy files.

Hmmm. I'd like to see if we can edit the original source instead, but if we have to run in the 4GL from the .cache we can do that.

> If you mean to generate the hints with an already converted app running on FWD, this can be done easier directly in FWD, by enhancing ControlFlowOps and other code.

No, I was thinking run in the 4GL.

> But this would require the app to already parse, convert and run in FWD.

Parse and call graph is OK, but convert is not. That is why I'm thinking run in 4GL.

> Also, I don't see how we can compute the hint ID at runtime; the only way to do this is at preprocessing time, where we can count these call sites and mark them with the hint ID.

Yes, we won't calculate it at runtime. We will instrument the 4GL with the right hint IDs in advance.

#213 - 08/23/2018 08:11 AM - Greg Shah

- Related to Feature #3695: instrument 4GL code for automated call-graph hint generation added

#214 - 08/23/2018 09:41 AM - Greg Shah

- Related to Feature #3697: provide a web UI to edit/manage call-graph hints and the root node list added

#215 - 08/23/2018 10:44 AM - Greg Shah

- % Done changed from 0 to 100

- Status changed from WIP to Closed

Items that could not be finished in this scope have been moved to other tasks.

#216 - 08/23/2018 10:44 AM - Greg Shah

- Related to Feature #3699: improve call-graph visualization to make it much more usable added

Files

janusgraph_0.2.0_javadoc.zip	4.31 MB	05/09/2017	Greg Shah
call_graph_visualization_branch_1514a_rev_11188_20170522.png	311 KB	05/22/2017	Greg Shah
call_graph_visualization_branch_1514a_rev_11204_20170526.png	302 KB	05/26/2017	Greg Shah
call_graph_on_hotel_gui_branch_1514a_rev_11239_20170601.png	343 KB	06/02/2017	Greg Shah