Conversion Tools - Feature #1519

Feature # 1511 (New): Reporting v3

report improvements

09/10/2012 12:20 PM - Greg Shah

Status:	New	**Start date:**	09/10/2012
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	40.00 hours
Target version:	Reporting 3.0		
billable:	No	**version:**	
vendor_id:	GCD		

Description

History

#1 - 09/26/2012 08:09 AM - Greg Shah

- Target version set to Milestone 2

- Estimated time set to 40.00

#2 - 10/21/2012 09:34 AM - Greg Shah

Add specific reports for:

- dynamic/unspecified EXTENT usage
- EXTENT parameter usage
- detect when the same HANDLE is assigned different kinds of resources (e.g. the same HANDLE var is used for a widget and for a socket)
- ~~incompatible PROGRAM-NAME usage~~

#3 - 10/21/2012 09:36 AM - Greg Shah

Refine reports that are too broad today. For example, reports such as "potential sockets usage" include language statements, attributes, methods... that are multi-modal (may or may not be referencing sockets). Add more sophisticated rules to differentiate the real usage from non-usage.

The DOM XML, SAX XML and SOAP reports need refining/addition of all attributes and methods.

#4 - 11/21/2012 09:09 AM - Greg Shah

- Target version changed from Milestone 2 to 25

#5 - 03/20/2014 11:32 AM - Greg Shah

Dynamic extent and extent parameter reports are already done (they were done before now).

The incompatible program-name usage report is no longer needed because our implementation is now fully compatible.

#6 - 03/20/2014 11:32 AM - Greg Shah

A report listing hard coded absolute filenames would be useful.

#7 - 09/07/2015 02:28 PM - Greg Shah

ADM/ADM2 provides procedures named constructObject and init-object. Both of these are valid call-graph references to other source files. Add a report for these constructs, where the buckets are the target procedure names (first parameter of these calls) and the details list the invocation locations.

Some further details as provided by a customer:

```
ADM and ADM2 are frameworks that use (many) persistently running procedures to build up GUI screens.  The main
 classes of objects are containers (windows, dialogs and frames), browsers and viewers.  In ADM there is the c
oncept of a data provider, but we rarely use them (browsers and viewers can supply their own data (and contain
ers actually!)), but in ADM2 we also use data providers (SDOs) extensively.

In particular, with tabbed frames, each tab normally contains at least one object, and embedded tabs (a tabbed
 frame within a tabbed frame) can create multiple objects within objects.  We have instances in GUI of 3 layer
s of tabbed form - I even think 4 might be the record!

So in both ADM and ADM2, in order to determine the procedure dependency tree, we need to know the parent/child
 container relationships.  This can be determined by looking for certain patterns in the parent (container) ob
jects:

ADM

init-object - value of first parameter

e.g. RUN init-object IN THIS PROCEDURE ( <newline>
INPUT 'somefolder.w':U ,

ADM2

constructObject - first entry (CHR(3) separated) of value of first parameter

e.g. RUN constructObject ( <newline>
             INPUT  'another.w<CHR(3)>....'
```

In the above examples, somefolder.w or another.w would be bucket categories.

For call-graph, the constructObject / init-object stuff was implemented using code in pattern/customer_specific_call_graph.rules. This is the core logic:

```
      <rule>(type == prog.int_proc or type == prog.filename) and
            (this.text.equalsIgnoreCase("init-object") or
             this.text.equalsIgnoreCase("constructObject"))

         <action>warn = true</action>

         <rule>parent.downPath("LPARENS/PARAMETER/EXPRESSION/STRING")
            <!-- determine the first parameter -->
            <action>chRef = this.parent.getImmediateChild(prog.lparens, null)</action>
            <action>chRef = chRef.getImmediateChild(prog.parameter, null)</action>
            <!-- double-check the EXPRESSION/STRING downpath for the first parameter, as the checked
                 parent path might be via a different child -->
            <rule>chRef.downPath("EXPRESSION/STRING")
               <!-- there is a string literal as first parameter, do not warn -->
               <action>warn = false</action>

               <action>chRef = chRef.getImmediateChild(prog.expression, null)</action>
               <action>chRef = chRef.getImmediateChild(prog.string, null)</action>
               <action>target = ecw.progressToJavaString(chRef.text)</action>
```

This code only gets called during call graph processing. The logic itself could be turned into a report with a little work.

#8 - 04/15/2016 09:40 AM - Greg Shah

Add a report for the PROFILER system handle.

In OO code, one can have variable references that refer to something in the inheritance hierarchy. These references appear in the built-in global var usage report and should be removed.

Malformed symbols by type (frame, stream, label, proc...).

Malformed frame names in UI reports.

In the assign style lang stmt report, NO-ERROR appears as a statement.

Browse editing mode (DEF_BROWSE/KW_DISP/KW_ENABLE).

Validation processing in browse.

OS programs referenced (parse actual text for shell commands and pull out the program names).

Temp-table field options (e.g. XML-NODE-TYPE, SERIALIZE-HIDDEN) are showing up on the widget options report.

#9 - 04/15/2016 09:42 AM - Greg Shah

From Eric on 04/14/2016:

Bugs and proposed enhancements for reporting:

1. The fields by type schema report is broken (the links, that is) for metadata fields.

2. Need gap analysis for meta data tables and fields.

3. Need a "Potential client-side WHERE clauses" report. The UDFs in WHERE clauses report is close, but it's missing a key factor: the detection of UDF parameters which reference fields in the record phrase's buffer (i.e., the "current" buffer). A UDF itself in a where clause may well just convert to a substitution parameter, if it doesn't reference the current buffer. But if the UDF has to execute against each record as part of a predicate filter, it means we're bringing each record back and testing it on the P2J server.

4. ProDataSet Usage report is missing some attributes and methods (this is an incomplete list):

```
attributes
        fill-mode
    methods
        read-json
        read-xml
        write-xml
        attach-data-source
        fill
        empty-dataset
```

5. Some way to differentiate or group polymorphic methods by the type of handle they operate on. This would quickly show us, for example, all the read-xml() methods operating on a buffer handle, all those operating on a temp-table handle, and all those operating on a ProDataSet handle.

#10 - 11/16/2016 01:34 PM - Greg Shah

- Target version changed from 25 to Reporting 3.0