Conversion Tools - Feature #6270

annotations-based per-file hints

04/10/2022 09:14 AM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		
Description			

History

#1 - 04/10/2022 09:48 AM - Greg Shah

Today, we have multiple types of hints at the individual source-file level.

- "UAST Hints" are read from a file that is constructed from the source file name (e.g. path/to/some-procedure.p.hints or path/to/SomeClass.cls.hints)
  - preprocessing/parsing/conversion
  - call-graph
- OCX/COM conversion hints are read from a file that is constructed from the source file name (e.g. path/to/some-procedure.p.ext-hints or path/to/SomeClass.cls.ext-hints)

The 4GL syntax provides an annotations facility. The synatx is similar to how it is done in Java, though I don't know of a mechanism to use the annotations (from 4GL code) as cleanly as in Java. Still, it is a mechanism to associate arbitrary "meta" configuration with specific code. An annotation can be placed anywhere a language statement can be placed (at least in our parser and in the code we've encountered). It isn't heavily used in legacy 4GL code today.

It seems to me that we could use this same capability to encode our hints. File-level hints could be placed at the top of the file. Hints that are specific to particular code (e.g. call graph hints) could be placed just in front of that code.

Advantages

- It can keep the hints together with the code that it references. This means that reading and maintaining the code is more natural.
- Creating a direct association between the cfg and the code may be less error prone. The call-graph hints for ambiguous call sites would be much better as inline annotations.
- Might be easier syntax than the XML file.
- For call-graph and OCX hints this may lead to a more compact syntax. For example, perhaps we drop the "hint ID" requirement on the call-graph side.
- When we implement our Eclipse plugin (#3883), we can provide helpers to encode these safely/easily.

Disadvantages

- Requires editing the code. When we are just parsing for the first time, we would have to make code edits which then must be accepted back into the source project. While a customer is still deciding whether or not to shift to FWD, such edits won't be accepted.

We could retain the existing UAST hints mechanism (as an optional source of hints) so that early project work would not require 4GL code edits. With that in place, I don't see too much downside to this.

For the OCX hints, these are things only implemented once a more serious effort in underway. It seems like the annotations approach might be a big benefit there. I especially like the idea that we can move the .wrx (design time and runtime) configuration into these annotations so that we automatically handle all the settings properly (see #4438).

Post here if you see some advantages/disadvantages which I haven't considered or if you have some other comments/thoughts about the idea.

#2 - 04/10/2022 11:08 AM - Tijs Wickardt

I like the idea of using 4GL annotations for FWD runtime hints.
It reduces the chance of an OE developer forgetting to update the FWD hints.
It should be limited only to information that FWD cannot deduce at the conversion.
If somewhere in the future FWD is able to interpret .wrx files (for example) the related annotations should be reported as being deprecated and harmlessly removable.

#3 - 04/10/2022 11:14 AM - Greg Shah

> It should be limited only to information that FWD cannot deduce at the conversion.

This is already the case. We only have hints for such things.

> If somewhere in the future FWD is able to interpret .wrx files (for example) the related annotations should be reported as being deprecated and harmlessly removable.

Due to Marius', Vladimir's and Hynek's work in #4438, we can indeed parse .wrx files. The problem is that it relies upon WIN32 APIs to properly read all the contents. This means the utility can not be run on Linux/UNIX and we do nearly 100% of our conversions on Linux/UNIX. Plus, if we don't migrate the .wrx content into annotations, then you must keep a legacy appbuilder dev environment in order to edit the settings. The annotations resolve this nicely.

#4 - 04/10/2022 12:04 PM - Hynek Cihlar

Greg Shah wrote:

> > It should be limited only to information that FWD cannot deduce at the conversion.
>
> This is already the case. We only have hints for such things.
>
> > If somewhere in the future FWD is able to interpret .wrx files (for example) the related annotations should be reported as being deprecated and harmlessly removable.
>
> Due to Marius', Vladimir's and Hynek's work in #4438, we can indeed parse .wrx files. The problem is that it relies upon WIN32 APIs to properly read all the contents.

And the actual serialized content stored in the wrx files is interpreted and recognized by the individual OCX components. So you need to parse the wrx files on Windows system with the required OCX components installed.

#5 - 04/10/2022 12:15 PM - Tijs Wickardt

Greg Shah wrote:

> > It should be limited only to information that FWD cannot deduce at the conversion.
>
> This is already the case. We only have hints for such things.

Great, that's what I hoped.

> > If somewhere in the future FWD is able to interpret .wrx files (for example) the related annotations should be reported as being deprecated and harmlessly removable.
>
> Due to Marius', Vladimir's and Hynek's work in #4438, we can indeed parse .wrx files. The problem is that it relies upon WIN32 APIs to properly read all the contents.

Excellent. So we can have a single source of truth for OCX: the wrx file.

> This means the utility can not be run on Linux/UNIX and we do nearly 100% of our conversions on Linux/UNIX.

Customers who have an OCX, have Windows. I see two options:

1. provide a build step through RPC to a Windows build machine
2. Use WINE

> Plus, if we don't migrate the .wrx content into annotations, then you must keep a legacy appbuilder dev environment in order to edit the settings. The annotations resolve this nicely.

I see what you mean. But although it might be an extra selling point for FWD customers, it's not a problem that FWD necessarily needs to solve.

So in conclusion: I do agree with using 4GL annotations in favor of hints files. But I'm hesitant about using it for situations that can be handled by the FWD conversion. I'll give it some more thought.

#6 - 04/10/2022 02:16 PM - Tijs Wickardt

Hynek Cihlar wrote:

> Greg Shah wrote:.
>
> > > .. interpret .wrx files (for example) the related annotations should be reported as being deprecated and harmlessly removable.
> >
> > Due to Marius', Vladimir's and Hynek's work in #4438, we can indeed parse .wrx files. The problem is that it relies upon WIN32 APIs to properly read all the contents.
>
> And the actual serialized content stored in the wrx files is interpreted and recognized by the individual OCX components. So you need to parse the wrx files on Windows system with the required OCX components installed.

Thx Hynek.

Has the following (or parts of it) been attempted?

1. Analyse the dll dependency tree on an OE Windows developer workstation
2. Copy the dll dependecy tree (files) to a Linux docker container
3. Install WINE on the docker container
4. Spin up the docker container
5. Do a docker exec to do the FWD OCX parsing

?

#7 - 04/10/2022 02:36 PM - Tijs Wickardt

Note: https://pkgs.alpinelinux.org/package/edge/community/x86/wine

(and winetricks)

Non edge version (stable): https://pkgs.alpinelinux.org/package/v3.15/community/x86_64/wine

#8 - 04/10/2022 07:41 PM - Greg Shah

I don't see the same value in maintaining the .wrx as a resource. It cannot be easily edited, it can only be edited using OpenEdge, it is separate from the source code. I think it is quite a mess.

Instead, we use the legacy system, one time, to run our utility and dump the .wrx contents into an XML format. From there we never need the legacy Windows environment again.

I definitely do not want us to bring along a complex Wine environment with all kinds of binary dependencies on DLLs and OCXes (and unknown licensing!), all so that this very fragile approach can be wired into our conversion process. Any problems will block things. It seems likely to create a mess.

#9 - 04/11/2022 02:41 AM - Tijs Wickardt

Greg Shah wrote:

> I don't see the same value in maintaining the .wrx as a resource. It cannot be easily edited, it can only be edited using OpenEdge, it is separate from the source code. I think it is quite a mess.
>
> Instead, we use the legacy system, one time, to run our utility and dump the .wrx contents into an XML format. From there we never need the legacy Windows environment again.
>
> I definitely do not want us to bring along a complex Wine environment with all kinds of binary dependencies on DLLs and OCXes (and unknown licensing!), all so that this very fragile approach can be wired into our conversion process. Any problems will block things. It seems likely to create a mess.

Downsides to this approach are:

1. Some customers have active development in OpenEdge. Introducing two sources of truth forces a customer to update both the .wrx and and the 4GL annotations. That seems wasteful to me, if FWD actually supports doing this automated, as you indicate (is it finished?). If you automate it by an Eclipse plugin, FWD forces a dependency upon the customer, who might not even be using Eclipse. And if Eclipse is mandatory: the FWD plugin must be active and must not fail, otherwise the resources are out of sync.
2. If the conversion modifies the source files, by adding annotations, those source files must be committed back into the customer VCS. This is very non standard in CI/CD. I see dragons there.

> a complex Wine environment with all kinds of binary dependencies on DLLs and OCXes (and unknown licensing!)

I don't know how complex it will be, but it doesn't sound too hard. The 'mess' is self contained, within a container (see: #6270-6). The customer owns any OCX licenses, so the customer would have to carry any legal issues arising from the image (which the customer should create, by the click of a FWD 'button', approving a warning about legal).
Of which I suspect there are none.
An OCX comes with a developer license, which is the one to use to be able to write the .wrx. It also comes with a runtime license, which is: run the configured OCX . FWD OCX analysis only needs the runtime part.

For FWD to maintain a single source of truth, ánd allow working without AppBuilder (and yes, that product is crappy and unstable), it would have to write the .wrx (not only parse). That sounds unfeasible to me, especially regarding developer licenses (and not to mention maintaining compatibility with OE versions).

We might ask some of our customers what they think before we choose an approach.

I'm not opposed to **any** solution, including manual edits and two sources of truths, but I think it's wise to have more input.

#10 - 04/11/2022 08:17 AM - Greg Shah

Every time we add **any** dependency, we add friction in the process of getting everything running properly. For conversion, today there are few dependencies but it still takes some effort to get running.

Runtime dependencies are more significant, but even relatively simple ones like patching ncurses is a constant source of headaches and effort for both our customers and for GCD. You are suggesting not just adding WINE, but having to maintain a working Windows configuration including an arbitrary set of binaries. We have found customers (including where you work) where the design time licenses are lost and some OCXs are only able to be executed with the runtime binaries. OCX support is a complex mess and adding emulated Windows would not be helpful.

> VMA and possibly other customers have active development in OpenEdge. Introducing two sources of truth forces a customer to update both the .wrx and and the 4GL annotations.

At worst, this is temporary during the conversion project. No one shifts to FWD and plans to stay on OE long term. .wrx files change very infrequently, so if there is some edit during the project, the effort to do it twice is trivial. In most projects, the .wrx files never change during the project. After the shift, there is only a single source of truth and the .wrx files are not used again. This approach is a far lower cost than adding a permanent dependency on WINE and the complexity comes with it. Sorry, I don't see WINE as a viable solution nor do I see the problem it solves for us as very important.

#11 - 04/11/2022 08:48 AM - Tijs Wickardt

No need for apologies, these are all valid points.

> a working Windows configuration

This seems to be a misunderstanding. I meant:
Only the binaries that are needed for unattended FWD parsing. Nothing functional. No developer license configuration if it can be helped.

And of course we don't know until we try. I can think of some pitfalls like you do. Then again:

> nor do I see the problem it solves for us as very important.

Agreed, I feel the same.

I can certainly live with this outcome, and I agree that (if needed) entering the OCX configuration parameters twice (FWD customer developer) isn't expected to generate a lot of work.

So we agree.

#12 - 04/11/2022 08:54 AM - Greg Shah

> > a working Windows configuration
>
> This seems to be a misunderstanding. I meant:
> Only the binaries that are needed for unattended FWD parsing. Nothing functional. No developer license configuration if it can be helped.

Understood, it is a valid point.

To clarify, I meant that to maintain the WINE environment one must have an original Windows functional environment on which it is based. Changes in one must be maintained in the other.

> So we agree.

Nice.

#13 - 04/11/2022 11:25 AM - Tijs Wickardt

Greg Shah wrote:

> To clarify, I meant that to maintain the WINE environment one must have an original Windows functional environment on which it is based. Changes in one must be maintained in the other.

Ah, clear. True (and cumbersome if not automated). I've already moved the idea to the outer fringes of my consciousness.