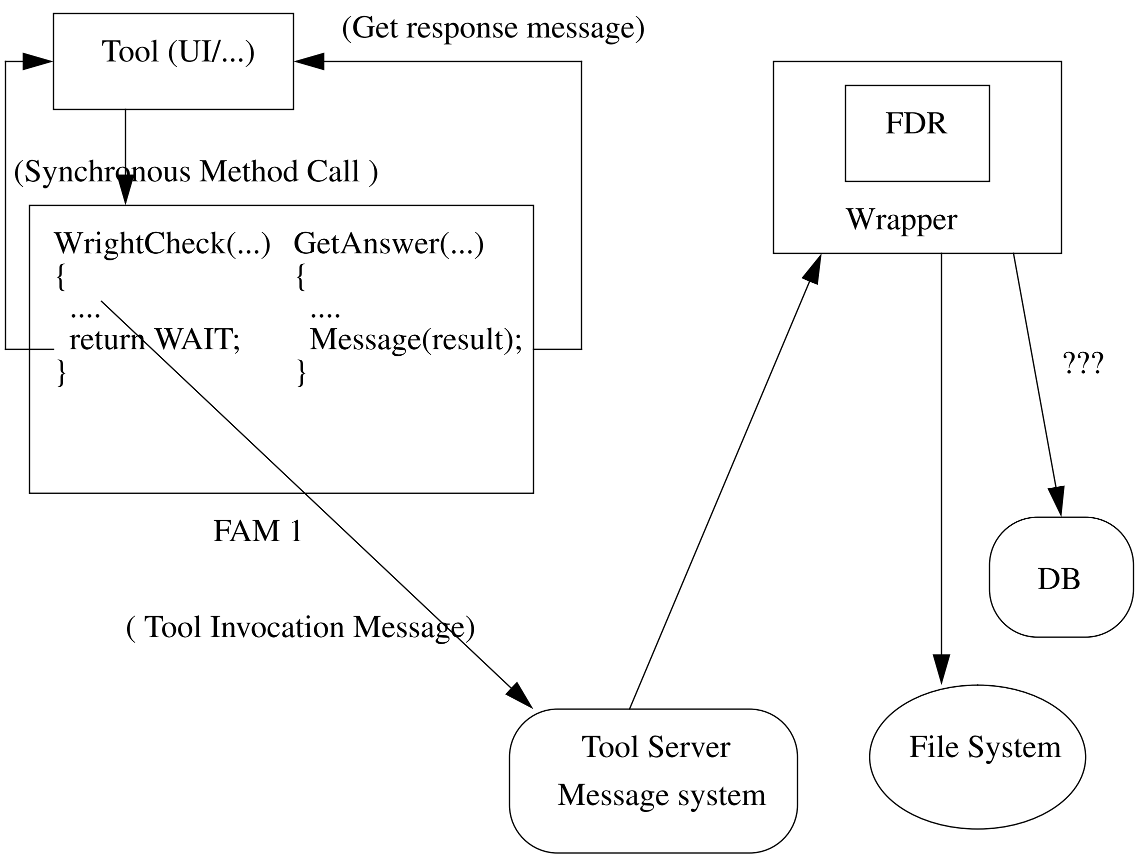
Asynchronous Connector To WRIGHT checker

Zhenyu Wang, Apr. 14

1. General Diagram



2. Explanations

All Following Tools refer to FAM-aware tools such as UI.

What the tool knows?

- FAM API ***WrightCheck(PortID, RoleID, void * OtherData).***
- Standard Message ***M_WrightResultBack(int Result, void * OtherData)***.

What the tool should do?

Call WrightChecker() with appropriate parameters and wait for M-WrightResultBack in its main event loop.

What is Otherdata?

OtherData is an auxiliary data chunk, basically tool programmer can put anything he want here. And those data will be passed back intact with result message. Typical use of this is to keep a Sequence ID here, because the tool may has requested multiple checking before the first answer come back, then it need to know what is the corresponding request of this answer and take actions e.g. lighten port and role to show incompatibility. It should be notes that the UI-Instance ID should also be put in here so the tool server knows where to send the result back. (If you keep UI-Instance ID field to be -1, then tool-server broadcast the result message, so the caller UI can decide whether to share the result itself in this way)

What the FAM knows?

- It support API WrightCheck(...).
- It register a hidden event handler GetAnswer() for the event M_WrightResultBack.

What does FAM do in WrightCheck()?

1. Search Result Cache in DB to see it there exists a clean copy of last checking result on the same port and role. If so, send a M_WrightResultBack immediately and return.
2. Search DB and retrieve the representation relevant to WRIGHT-Spec.(Typically this may be a external file reference kept in some slot of the object)
3. Compose a Tool Invocation Message in standard format:
   Tool_Class: WRIGHT_CHECKER
   Tool_Op: Port_Role_Check
   Tool_Data :
   Port_Val_Type, Port_Val,
   Role_Val_Type, Role_Val,
   void * OtherData

(Port_Val_Type specify what kind of value we are passing in Port_Val, could be: 0-ACME description, 1-File Reference, 2-WRIGHT-Spec..., so is Role_Val_Type)

4. Send the message to tool_server
5. Use some external locking mechanism to lock the port and role under check.
6. return WAIT

What does FAM do in GetAnswer()?

The method got called automatically when the FAM receive M_WrightResultBack message, it will release lock on port and role, put the new result into DB cache, then send the message back to UI.

What the tool server knows?

- It is coupled with event systems.
- It keep track of all tool-instance ID and all registered event handler.
- It has knowledge of tool-config info.

What the tool server do?

- Upon receiving the tool-invocation message, it will check if one Wright-Checker is also running, if not start one, otherwise direct the message to the current background Wright-Checker.
- Upon receiving the M_WrightResultBack message, it check the OtherData to determine the Caller tool ID and route the message back to the FAM of that tool so the GetAnswer will get called.

What the wrapper knows?

This depends on how do we implement of WrightCheck() method If the WrightCheck will do all chores such as read external file, parse ACME to get Wright-spec, then the wrapper only need to know how to feed the WRIGHT-spec into FDR. Another extreme case is we have a very lazy WrightCheck(), it just pass PortID and RoleID, then the wrapper has to look into File system and DB. Perhaps we will take a middle way approach as I described above.

What the wrapper do?

Keep request queue, format input data and feed data into FDR, translate output, compose M_WrightResultBack message and send it back to tool_server.

How about non-FAM aware tool?

It is easy for them to use WRIGHT checker as long as it supports events.