Assisting Human Intelligence...!

SGV SARC

Crystal *DataFlow*

Review, Analyse, und Dokumentation von Datenstrukturen

Der Projektbrowser von Crystal C/C++ stellt zu Funktionen jede aus dem Quellcode ermittelbare Information vom Definitionsort, über Aufrufer und gerufene Unterprogramme, bis zu einem Aufrufbaum bereit. Crystal *DataFlow* geht noch einen Schritt weiter und stellt Hierarchie und Aufrufsequenz von Datenobjekten in einem Flowchart ähnlichem Diagramm dar. Der Nutzen wird schnell deutlich, wenn wir zwei ganz alltägliche Anforderungen betrachten.

Anforderung 1
Nehmen wir an, Sie möchten den Datenfluss einer lokalen Variable verfolgen. Nehmen wir weiters an, die lokale Variable wird als Argument in einem Funktionsaufruf verwendet. Damit die Aufgabe ein wenig reizvoller wird, wird das als Argument übergebene Objekt in der Funktion mit einem anderen Namen benutzt.

Wahrscheinlich(st)e Lösung
Man startet eine Suche über das Projekt und geht aus der Ergebnisliste Zeile für Zeile durch und prüft im Quellcode wie das Objekt verwendet wird. Einigermaßen sicher und bewährt, aber leider auch mühsam.

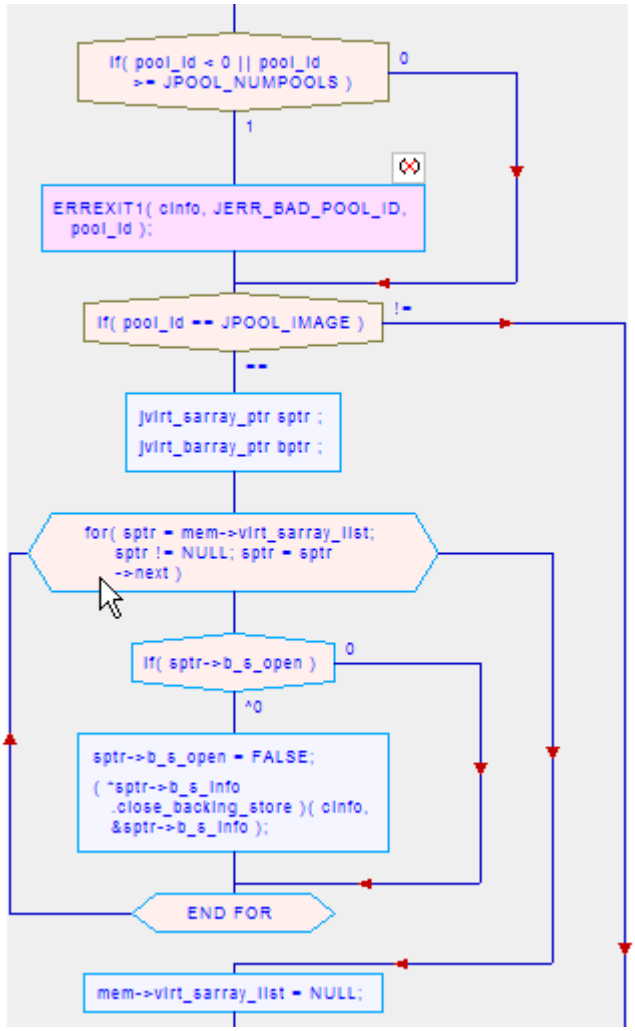
Datenflussdiagramme
Mit Crystal *DataFlow* setzen Sie den Cursor auf die Variable und erhalten in sekundenschnelle ein Diagramm, das alle Anweisungen enthält, welche die Variable benutzen oder ändern und zusätzliche Kontrollstrukturen wie Schleifen, if-else Entscheidungen etc., die sich direkt auf diese Anweisungen auswirken. Und auch das mit einem anderen Namen verwendete Objekt wird zuverlässig erkannt.

Ein Doppelklick auf den Funktionsaufruf expandiert das Diagramm wie im nebenstehenden Screenshot und zeigt wie die korrespondierenden Parameter in dem Funktionsaufruf verwendet werden.

Im Datenflussdiagramm werden alle auf das Objekt referenzierenden Stellen übersichtlich aufgelistet und Funktionsaufrufe können zur Inspektion direkt expandiert werden.

Natürlich gibt es Code Coverage. Man klickt ein Symbol und der entsprechende Abschnitt wird im Editor hervorgehoben angezeigt.

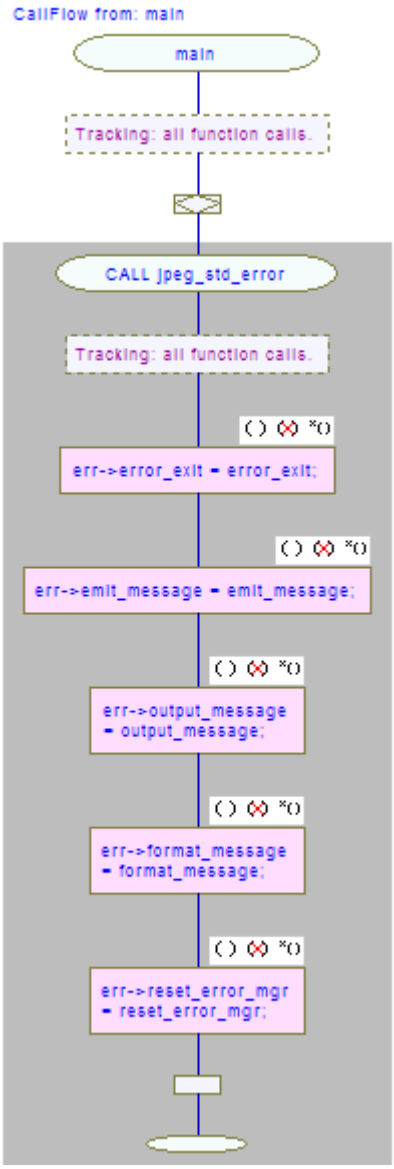
Für die Diskussion mit Kollegen oder eine Dokumentation können die Diagramme gedruckt oder als Grafikdatei exportiert werden. An eine Exportfunktion nach Microsoft VISIO wurde ebenfalls gedacht.



Anforderung 2
Nehmen wir an Sie machen ein Code Review, debuggen einen Softwarefehler oder versuchen sich schnell und effizient in alten Quelldateien zurecht zu finden – Ihre Anforderung wird in jedem Fall die Analyse von Funktionen und deren Aufrufer und die gerufenen Unterprogramme enthalten.

Wahrscheinlich(st)e Lösung
Eine Kombination aus den Möglichkeiten der verwendeten IDE, der existierenden Dokumentation und Stift und Papier.

Aufrufdiagramme
Das Aufrufdiagramm zeigt die Funktionsaufrufe und ein Kontrollflussdiagramm der Funktion. So wird deutlich welche Funktionsaufrufe in Schleifen, then oder else Zweigen einer if-Abfrage usw. enthalten sind. Mögliche Ausführungspfade und Funktionsaufrufe in den Pfaden sind gut nachvollziehbar, weil direkt auch ersichtlich ist wohin sie führen. Funktionsaufrufe können zur Untersuchung der Funktion im Diagramm expandiert werden.



Der nebenstehende Screenshot zeigt einen kleinen Ausschnitt einer Aufrufsequenz einer main mit einer expandierten Funktion und deren Funktionsaufrufen, die wiederum im Diagramm weiter expandiert werden können. Code Coverage und die Anzeige der Typdefinition sind gewährleisten, dass man sehr schnell einen vollständigen Überblick über die Logik einer Funktion erhält.

Besonders wertvoll werden die Funktionsaufrufdiagramme, wenn die Aufrufsequenz durch Preprozessoranweisungen bedingt ist oder wenn durch if-then-else Entscheidungen verschiedene Aufrufpfade durchlaufen werden. Im Diagramm sieht man sofort ob die Sequenz korrekt ist und alle Argumente richtig gesetzt sind.

Modularität
In modernen Projekten ist man gehalten Funktionen möglichst modular zu gestalten um die Wartbarkeit der Software zu gewährleisten. Anstatt einer großen Funktion mit komplizierten Algorithmen bekommt man vielleicht 10 kleine, in sich gut verstehbare Funktionen. Das erleichtert zwar die Analyse im Quellcode, erfordert aber hohes Wissen und viel Zeit um all die Abhängigkeiten gegen zu prüfen.

Auch hierfür wirken Aufrufdiagramme Wunder indem sie die durch Funktionsaufrufe ausgelagerten und die in der Funktion direkt implementierten Aufgaben, in einem einzigen Diagramm zeigen. Damit helfen sie nicht nur die Fragmentierung der Aufgaben in die ausgelagerte Funktionen zu verstehen, sondern helfen auch beim Redesign. Im Ergebnis erhält man ein besseres Softwaredesign und damit robusten, wartbaren Code.

Systematik
Funktionsaufrufdiagramme können zur systematischen Analyse über das ganze Projekt erstellt werden, wobei die erste Ansicht eine symbolische Auflistung der Funktionen enthält. So ist gewährleistet, dass man in der Analyse keine Funktion übersieht und wieder auf Fehlersuche geht.

Druck und Export
Die Diagramme können für einzelne Funktionen oder das ganze Projekt gedruckt, als Bilddatei ausgegeben oder für MS Visio aufbereitet exportiert werden. Alles in nur einem dateiübergreifenden, verbundenen Diagramm. Sie quälen sich nicht mehr durch einzelne Dokumente und suchen die logischen Verknüpfungen. Damit haben sie eine perfekte Grundlage für Diskussionen, Projekt-Analyse und Dokumentation.

Crystal *DataFlow* für C/C++ wird in Crystal *FLOW* und Crystal *REVS* mit unterschiedlichen Leistungsmerkmalen angeboten.

Vertrieb und Produktsupport in D/A/CH

EASYCODE GmbH
Löwenberger Straße 50
D-90475 Nürnberg
Tel:+49-911-99 840-60
Fax:+49-911-99 840-97
e-mail: info@easycode.de
support@easycode.de