CANTATA

Quali sono le novità di Cantata 9.0?

Cantata 9.0, disponibile da ottobre 2019, è una nuova importante release che offre nuove e significative funzionalità tra cui il supporto per il Test Driven Development (TDD) e il supporto AutoTest per C ++. Questo documento descrive in dettaglio le modifiche più importanti in Cantata versione 9.0.

Introduzione

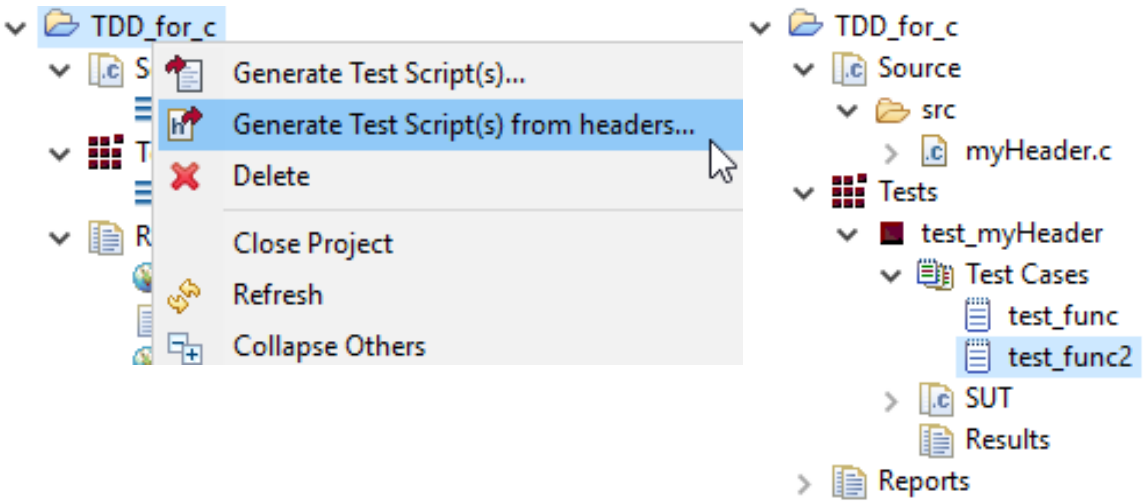
Cantata 9.0, disponibile da ottobre 2019, è una nuova importante release che offre nuove funzionalità avanzate. Questa versione introduce il supporto per il Test Driven Development (TDD) e migliora la funzione all'avanguardia dell'AutoTest, consentendo ora la generazione automatica di test per il codice C ++. Cantata 9.0 contiene anche molti altri piccoli miglioramenti e correzioni. La serie completa di modifiche è documentata nelle Note di Rilascio che tengono traccia di tutte le modifiche in Cantata dalla versione 4.1. Le modifiche più importanti sono evidenziate nelle sezioni seguenti.

Supporto del processo di "Test Driven Development

Il supporto per il Test Driven Development (TDD) è ora implementato in Cantata. TDD è un metodo di sviluppo per il quale vengono creati i test unitari in prima istanza e rispetto ai quali viene implementato il codice sorgente. Questa agile tecnica può essere utilizzata per scrivere codice "più pulito", poichè l'attenzione è focalizzata sullo sviluppo di casi di test a partire dai requisiti, piuttosto che iniziare direttamente scrivendo il codice sorgente.

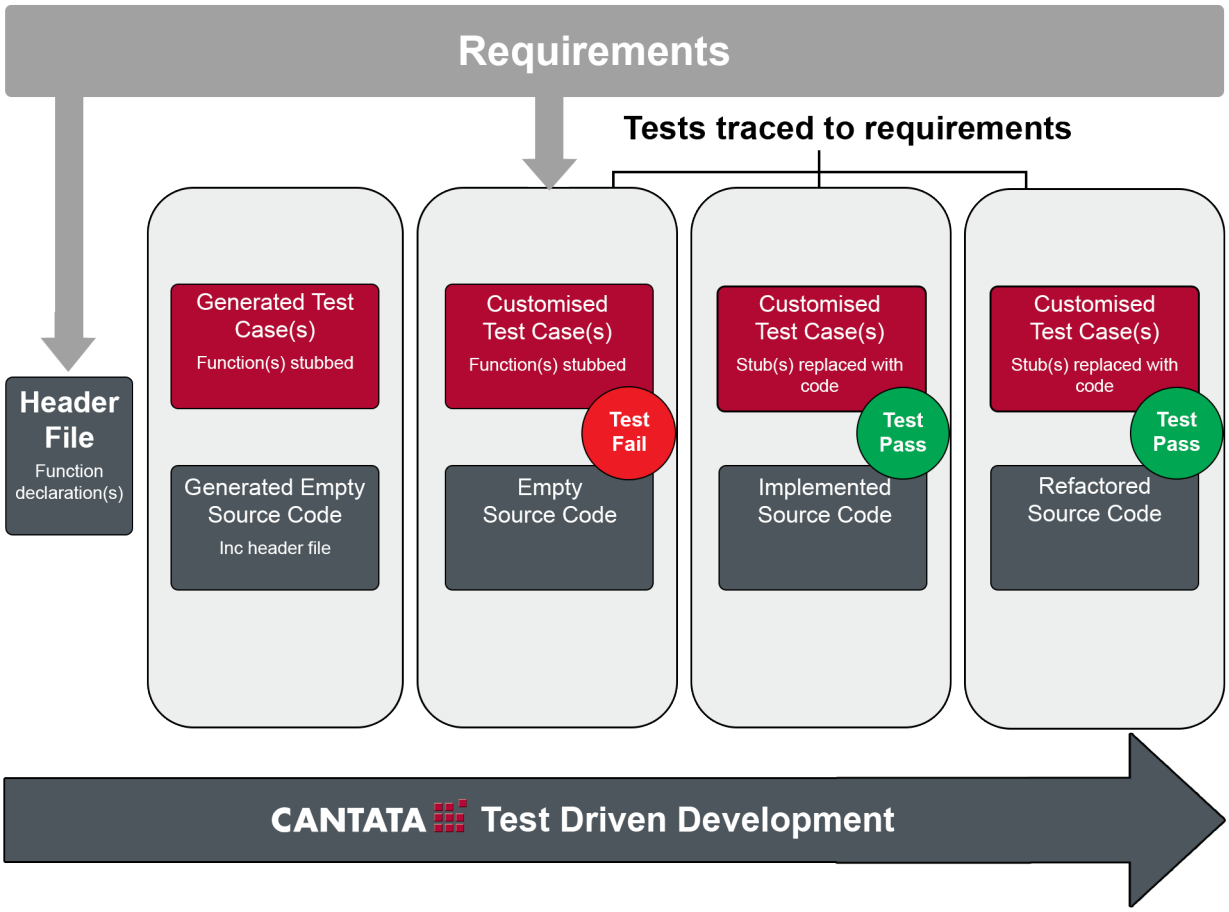
Cantata 9.0 offre nuove funzionalità per supportare il processo TDD e per facilitare la produzione di test per il codice sorgente che non è stato ancora scritto.

La funzione TDD può essere utilizzata per generare casi di test non appena sono disponibili i prototipi di funzione all'interno dei file di intestazione. Viene creato un progetto Cantata standard, per fornire l'ambiente di test e tutti i report utilizzati come prova di certificazione e come previsto da Cantata stesso. L'uso di Cantata per il processo TDD migliora il tradizionale approccio di test unitario in Black Box, garantendo nello stesso tempo l'accesso a funzioni di test complete in White Box, poiché Cantata può chiamare direttamente elementi di codice interni incapsulati come dati / funzioni privati e statici.



Generazione di script TDD nella finestra Cantata Test Explorer

Gli script di test vengono generati automaticamente con un caso di test per ciascun prototipo di funzione dichiarato nel file di intestazione. Questi casi di test possono essere usati come prima base per la verifica ed evitano la necessità di inserire manualmente le informazioni sui prototipi di funzioni. Ogni dato globale dichiarato nel file di intestazione viene rilevato automaticamente e verificato. La funzione TDD di Cantata genera file sorgenti vuoti, cosi che una volta implementati i corpi funzione in una seconda fase, i test possono essere rapidamente ricostruiti ed eseguiti per testare il nuovo codice.



La funzione di Traccia in Cantata può essere utilizzata in combinazione con Cantata TDD per migliorare lo sviluppo di casi di test in base ai requisiti funzionali. I requisiti importati possono essere associati ai casi di test durante la loro progettazione. I collegamenti tra casi di test, codice e requisiti funzionali sono una garanzia di chiarezza e semplificano il refactoring del codice.

Gli stub di simulazione vengono generati per tutte le funzioni il cui il corpo non è stato ancora modificato, ciò consente l'esecuzione dei casi di test anche se l'intero codice sorgente non è ancora disponibile o addirittura scritto. Questi test preliminari probabilmente falliranno fino all'implementazione del codice appropriato. Una volta che il codice è stato scritto, questi stub possono essere rimossi o convertiti automaticamente in Cantata wrapper (per intercettare e controllare le chiamate a oggetti / funzioni) utilizzando l'analisi delle modifiche del codice di Cantata. Al superamento dei test, è possibile eseguire il refactoring del codice sorgente e i test vengono automaticamente riprodotti per garantire la non-regressione.

AutoTest per Codice C++

La funzionalità AutoTest di Cantata è stata estesa al support del codice C ++. AutoTest genera automaticamente dei test unitari per il codice sorgente C ++, che possono essere utilizzati su progetti nuovi o esistenti, rendendo più semplice:

- Configurare la generazione automatica dei test
  - scegliere come saranno esercitati i rami di esecuzione, attraverso la selezione di diverse regole di copertura del codice.
  - configurare il tipo di test, attraverso le diverse opzioni di generazione per stub / wrapper, controlli dei dati, etc.
- Generare script di test per ottenere una copertura del codice del 100%
- Sostituire gli spazi vuoti di codice, una volta eseguiti i casi di test dei requisiti funzionali, attivando l'aggiunta automatica di casi di test generati da AutoTest per funzionalità e metodi non ancora coperti.
- Creare una base di Test di Unità per scopi di non regressione.
- Effettuare un upgrade da altri tool di test a Cantata

AutoTest analizza il sorgente C ++ per determinare tutti i possibili percorsi di esecuzione in base a metriche di copertura del codice predefinite (come il 100% dei punti di ingresso della funzione, il 100% delle istruzioni, il 100% delle decisioni, o il 100% MC / DC). Un algoritmo crea i vettori di test per eseguire i diversi rami logici, utilizzando le funzionalità avanzate di test White Box di Cantata che controllano dati, parametri e interfacce di chiamata delle funzioni.

Cantata Test Results Summary

Project: *AutoTest Multi File Demo* **Overall Result: Pass**

Summary Information

Hostname	Matt	**Cantata version**	v9.0
Summary generated	Jan 17, 2019, 11:30 AM	**Time elapsed during test run**	16 seconds

Build Summary		**Results Summary**		**Coverage Summary**	
Total tests	4	**PASSED**	4	**Entry point (E)**	100%
Compile attempted (files)	8	**FAILED**	0	**Statement (S)**	100%
Link attempted (tests)	4	**Total checks**	233	**Decision (D)**	100%
Execute attempted (tests)	4	**Total checks failed**	0	**Call-return (C)**	100%
		Total script errors/call failures	0	**MC/DC - masking (M)**	100%
				MC/DC - unique cause (U)	100%

I vettori di test controllano l'esecuzione del codice e controllano i parametri scambiati tra metodi, i valori dei dati globali, l'ordine delle chiamate e i valori di ritorno delle funzioni. I casi generati da AutoTest sono modificabili allo stesso modo dei casi generati dall'utente. Ogni caso di test, quando viene creato, include una descrizione del percorso logico esercitato nel codice, che semplifica gli sforzi di manutenzione e la tracciabilità verso i requisiti funzionali.

AutoTest in Cantata 9.0 supporta I seguenti elementi di C++ 03 e versioni precedenti:

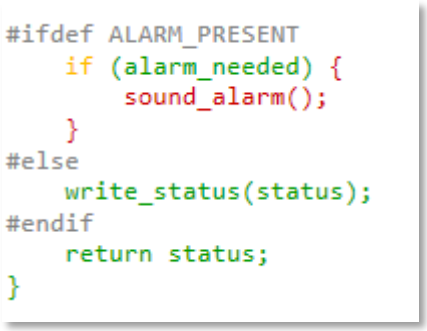
- Classi di base C++ astratte e concrete.
- Meccanismi di Sovraccarico ed Ereditarietà
- Spazi dei Nomi e Classi
- Gestione delle Eccezioni
- Modelli se istanziati nel codice in prova.
- Basi miste di codice C e C++

Cantata AutoTest può essere utilizzato in determinate condizioni d'uso con il codice C ++ contenente elementi sintattici/semantici/di versioni diversi da quelli sopra elencati. I test per il codice C ++ supportato verranno auto-generati e potrebbero essere creati test aggiuntivi per raggiungere il livello desiderato di copertura del codice.

Miglioramenti nella copertura del codice

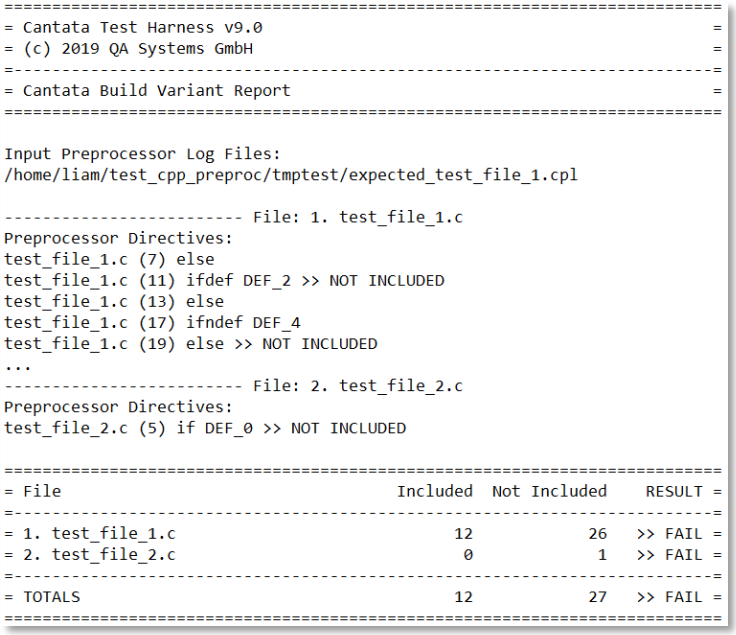
La copertura del codice per le varianti di configurazione del software

La copertura del codice per le varianti di configurazione del software C / C ++ è stata implementata per supportare l'aggregazione dei dati di copertura raccolti attraverso più processi di compilazione. Cantata analizza a tal fine le macro utilizzate dal preprocessore (#define) per identificare varianti del processo di costruzione del codice. Il Visualizzatore grafico di Cantata per la copertura del codice ora può aggregare i dati per lo stesso codice sorgente su più esecuzioni corrispondenti alle varianti di build. È inoltre possibile generare un rapporto testuale per riassumere le direttive del preprocessore che sono state utilizzate (o ignorate). Questo rapporto può essere utilizzato in un processo di certificazione per varianti del codice sorgente.



```
#ifdef ALARM_PRESENT
    if (alarm_needed) {
        sound_alarm();
    }
#else
    write_status(status);
#endif
    return status;
}
```

Sopra : Visualizzatore grafico di copertura che mostra il codice eseguito per diverse varianti della macro del preprocessore denominata « Allarme presente »



```
================================================================================
= Cantata Test Harness v9.0                                                    =
= (c) 2019 QA Systems GmbH                                                     =
================================================================================
= Cantata Build Variant Report                                                 =
================================================================================

Input Preprocessor Log Files:
/home/liam/test_cpp_preproc/tmptest/expected_test_file_1.cpl

------------------------- File: 1. test_file_1.c
Preprocessor Directives:
test_file_1.c (7) else
test_file_1.c (11) ifdef DEF_2 >> NOT INCLUDED
test_file_1.c (13) else
test_file_1.c (17) ifndef DEF_4
test_file_1.c (19) else >> NOT INCLUDED
...
------------------------- File: 2. test_file_2.c
Preprocessor Directives:
test_file_2.c (5) if DEF_0 >> NOT INCLUDED

================================================================================
= File                                  Included  Not Included    RESULT =
--------------------------------------------------------------------------------
= 1. test_file_1.c                            12            26   >> FAIL =
= 2. test_file_2.c                             0             1   >> FAIL =
--------------------------------------------------------------------------------
= TOTALS                                      12            27   >> FAIL =
================================================================================
```

A destra: rapport testuale certificabile

Il rapporto sulla copertura differita è certificabile

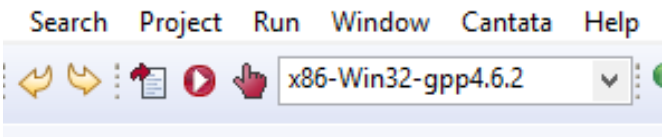
Il meccanismo di analisi differita di Cantata trasferisce i dati di copertura raccolti dall'ambiente di esecuzione al computer host per eseguire controlli di copertura del codice e generare i report corrispondenti. Questo meccanismo è molto utile per target con memoria insufficiente. L'utility Cantata CPPGETCOV, utilizzata a questo scopo, fa ora parte degli elementi certificati dall'ente indipendente SGS-TÜV SAAR GmbH. Ciò implica che il rapporto in formato .ctr ricavato con questa utility può essere utilizzato per ottenere i crediti di certificazione appropriati.

Risultati della copertura in formato HTML

I risultati della copertura del codice ora possono essere generati in HTML a partire dai dati disponibili nel visualizzatore grafico di Cantata per la copertura del codice.

Annulla e Ripristina

Tutte le modifiche apportate agli script di test attraverso le diverse viste di Cantata in Eclipse (ad esempio Test Explorer, Test Case Editor o Test Properties) vengono ora salvate automaticamente. La funzionalità Annulla / Ripristina pertanto può essere aggiunta alle viste di Cantata per correggere e ripetere le azioni dell'utente, che può decidere di utilizzare i pulsanti della GUI, nonché le scorciatoie da tastiera: ctrl + Z (annulla) oppure ctrl + Y (ripristina).



Aggiornamento delle piattaforme supportate

Come con qualsiasi nuova versione di Cantata, le piattaforme supportate vengono aggiornate ed espanse. Cantata, mediante Eclipse (Built-on-Eclipse®), è integrato con i principali ambienti di sviluppo, nonchè le toolchain disponibili come plug-in (Eclipse-Ready®). Cantata 9.0 è integrato con la versione Eclipse 2018-09 (Eclipse 4.9) ed è anche installabile come plug-in, Eclipse-Ready® dalla versione Juno (4.2) fino alla versione 2018-09 (4.9), Ciò offre ai nostri clienti l'accesso a un ricco ecosistema di strumenti integrati (ad esempio catene per ALM, CI e SCM).

Il supporto per i compilatori GNU GCC e g ++ è stato aggiornato alla versione 8.2 su Windows e 8.3 su Linux.

Il supporto per i vari costrutti di linguaggio C ++ 11 e C++14 è stato esteso anche in Cantata 9.0. Per un elenco dei diversi costrutti C ++ 11 e C ++ 14 (con collegamenti ipertestuali a descrizioni dettagliate fornite da open-std.org), contattare sales@qa-systems.com.