Software Plattform Embedded Systems 2020

Verteilte Entwicklung

31. März 2010

Ziele / Anforderungen

Unterstützung verteilter Modellierung

- Persistenzmechanismus, der paralleles Arbeiten erlaubt und ggf. unterstützt
- Konsistenzsicherung von Daten nach parallelen Änderungen
- Prozessunterstützung

Verbindung zu Modellen

- Offline
  - Vorteile: unabhängiges Arbeiten auf verteilten Kopien, Netzunabhängigkeit, Änderungen lokal rückgängig machen
  - Nachteile: Problem der Integration von Kopien (Merge)
- Online
  - Vorteile: unmittelbare Integration von Änderungen
  - Nachteile: Netzabhängigkeit, Auflösung Undo-Redo-Abhängigkeiten

Locking-Strategie

- Pessimistisches Locking: Konflikte durch Locks verhindern vor Änderung
  - Vorteile: keine Merge-Konflikte
  - Nachteile: paralleles Arbeiten auf gleichem Modell nicht möglich
- Optimistisches Locking: Konflikte durch Merge beheben nach Änderung
  - Vorteile: paralleles Arbeiten auf gleichem Modell
  - Nachteile: Merge-Konflikte möglich

Modularisierung

Nutzt die Modularisierung der Modellierungs-sprache etwas für die Verteilte Entwicklung?

- Locking-Granularität: Modul
- Write-Lock auf die editierten Module
- (Live) Read-Access auf die abhängigen Module
- Einbau von Modularisierungskonstrukten in Modellierungssprache

Konsistenzsicherung

Sollen Modelle konsistent sein, wenn man sie committed?

- Konsistenz sichern $\Rightarrow$ mehr Konflikte
- Konflikte vermeiden $\Rightarrow$ Inkonsistenz

Begriff der Konsistenz

- Konsistent bezüglich den Constraints der Modellierungssprache
- Wenn ein Modell konsistent ist, dann kann Code daraus generiert werden
- Konsistenz verlangt auch ein gewisses Maß an syntaktischer Vollständigkeit
- Unterschiedliche Schwere von Verletzungen der Konsistenzbedingungen

Konsistenzsicherung

Arten von Konsistenzbedingungen

- Invariant (immer gültig)
- Variant (gültig abhängig von Prozessfortschritt, Quality Gates, Prozessunterstützung erforderlich)

Arten der Konsistenzsicherung

- konstruktiv: Operationen, die von einem konsistenten Zustand zum nächsten transformieren
- analytisch: Bedingungen, die regelmäßig überprüft werden

Versionierung

Granularität

- einzelnes Modellelement
- Modellmodul

Arten von Versionen

- Trunk: Hauptversion
- Tagging: Auszeichnen von bestimmten Versionen (z.B. Releases), kein Merge zum Trunk
- Branches: Experimente, Merge zum Trunk erwünscht