EEExplore

Methoden und Werkzeuge zur E/E-Architekturexploration und –Optimierung

Schlussbericht aquintos

BMBF Förderkennzeichen: 01IS08028A

aquintos GmbH
Philipp-Reis-Strasse 1
76131 Karlsruhe
Deutschland

Autor: Dr.-Ing. Clemens Reichmann
Telefon: +49 721 91430-100
Fax: +49 721 91430-100
Email: reichmann@aquintos.com

Inhaltsverzeichnis

1 Ziele

1.1 Aufgabenstellung

Die Aufgabenstellung des Vorhabens EEExplore bestand darin, erstmals eine Werkzeug-gestützte Entwurfsraum-Exploration und Architekturoptimierung für E/E-Systeme zu ermöglichen. Hierzu wurden mehrere innovative, teilweise aufeinander aufbauende Ansätze verfolgt, die weit über den Stand der Technik im Bereich der Architekturentwicklung hinausgehen:

- Eine ganzheitliche Architekturbewertung über modellbasierte Metriken
- eine teilautomatisierte Architekturoptimierung durch eine modellbasierte Entwurfsraumexploration
- die Nutzung von Architektur- Entwurfsmustern zur wissensbasierten Entwicklung

Die aquintos GmbH besitzt als koordinierender Antragsteller, Entwickler und Vermarkter des Produkts PREEvision tiefgehende Einblicke in die durch Kundenwünsche getriebenen Anforderungen aus der Automobilindustrie an die Weiterentwicklung ihres marktführenden Architekturbewertungswerkzeugs. Sie übernahm im Rahmen des Gesamtprojekts die federführende Erarbeitung der Anforderungen aus Sicht der Marktgegebenheiten in den einzelnen Teilthemen.

aquintos legte einen wissenschaftlich-inhaltlichen Projektschwerpunkt auf den Bereich der Metrikmodellierung. Der Antragsteller besaß bereits ein weitgehendes Verständnis bei der Definition von textuellen Metrikbeschreibungen und der Ausführung derselben. Die langjährige Erfahrung bei der Gewinnung von robusten Metamodellen war essentiell, um eine Verknüpfung der zu erarbeitenden Metrikmodellierung mit dem Architekturmodell zu ermöglichen.

In den aufeinander aufbauenden Arbeitspaketen in den Projektthemenfeldern modellbasierte Architekturbewertung auf Basis des Metrikmodells, der Architekturexploration und -optimierung sowie der Gewinnung von Entwurfsmustern für Architekturmodelle lagen die Schwerpunkte der Arbeit der aquintos GmbH jeweils auf folgenden Aspekten:

- Am Markt orientierte Anforderungsanalyse
- Validierung und Bewertung der technisch-wissenschaftlichen Konzepte und Prototypen des Projektpartners FZI
- Erarbeitung benötigter Datenmodelle und Notationen
- Konzeptionelle und praktische Integration der Ergebnisse in das Werkzeug PREEvision, dabei insbesondere auch Beachtung der Benutzerführung und Ergonomie des Softwarewerkzeugs

Die drei Schwerpunkte wurden an Hand von realitätsnahen Modelldaten untersucht. Hierzu stand aquintos in enger Kooperation mit Architekturentwicklern (Fahrzeughersteller, Tier-1-Zulieferer), die im Rahmen eines regelmäßig tagenden industriellen Anwenderkreises in das Projekt eingebunden wurden. Zum Ende des Projektes werden die Methoden und Werkzeugkomponenten zu einem Demonstrator integriert, der eine Evaluation der Projektergebnisse in einem realen Entwicklungsprozess ermöglichte. Als praktikables Anwendungsszenario wurde die Optimierung von Teilsystemen wie dem Türsteuergeräte-Verbund vorgesehen.

1.2 Voraussetzungen der Vorhabensdurchführung

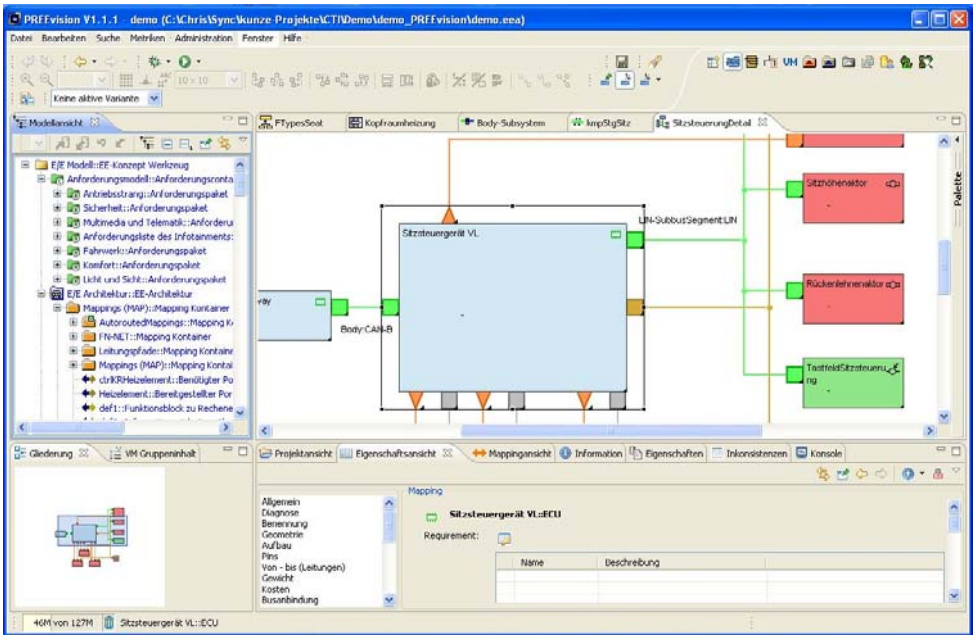


Abb. 1: Architekturentwicklungswerkzeug PREEvision

Als Tool-Hersteller des Modellierungswerkzeugs PREEvision liegt seit 2005 der Fokus auf Themenstellungen der Elektrik/Elektronik-Architektur, Konzeption und Entwicklung.

PREEvision ist ein CASE-Werkzeug zur modellbasierten Entwicklung und Bewertung von Elektrik/Elektronik-Architekturen im Kraftfahrzeug. Unterstützt werden Entwurfsschritte im Bereich des Requirements Engineerings, der Funktionalitätenentwicklung, der Netzwerk- und Komponenten-Architektur, der Bordnetz- und Leitungssatzentwicklung sowie der Topologie. Ein integriertes Variantenmanagement erlaubt die Definition und vergleichende Bewertung von alternativen Architekturkonzepten im Rahmen einer Baureihen- bzw. Plattformentwicklung. PREEvision verfügt über regelbasierte Konsistenzchecker sowie Variantenpropagationskonzepte, die kundenseitig angepasst werden können. Eine große Zahl an Import und Export-Schnittstellen stehen zur Verfügung, darunter werden Industriestandards wie z.B. AUTOSAR unterstützt.

Zu den Kerntechnologien gehört auch die leistungsfähige Modell-zu-Modell-Transformations-Entwicklungsumgebung aquintos.M2M für die Modell-Migration und Synchronisierung zwischen etablierten CASE-Werkzeugen.

Neben dem Produkt PREEvision bietet das Unternehmen auch alle Engineering- und Support-Services an, die im Umfeld von OEM und Supplier für den erfolgreichen Betrieb des Werkzeugs benötigt werden. Hierzu zählt das Consulting im Rahmen von Elektrik/Elektronik-Architekturprojekten. Dabei werden in vielen Fällen langfristig Entwicklungsleistungen in den Entwicklungsabteilungen beim Kunden vor Ort durch Resident Ingenieure der aquintos GmbH betreut.

1.3 Planung und Ablauf des Vorhabens

Zu Beginn des Vorhabens wurde zwischen den beteiligten Partnern ein Zeit- und Arbeitsplan für das gesamte Vorhaben erstellt und ein Kick-Off-Meeting durchgeführt, auf dem unter anderem die Projektkommunikation geplant wurde.

So wurden wöchentlich Projektmeetings durchgeführt, auf denen jeder Partner über den aktuellen Stand seiner Arbeiten berichtete. Parallel wurden zwischen den kooperierenden Partnern regelmäßige Arbeitstreffen in Form von Workshops in den jeweiligen Arbeitspaketen durchgeführt. Im Verlauf des Vorhabens fanden ständig telefonische und Kontakte per E-Mail statt, mit denen eine kontinuierliche Abstimmung und Information über den Bearbeitungsstand erfolgte.

Die wissenschaftlichen Untersuchungen wurden in enger Zusammenarbeit mit dem Projektpartner durchgeführt. Auf diese Weise wurden dem Projektpartner jeweils die aktuellsten Entwicklungsstände der entwickelten Systeme zur Verfügung gestellt und Rückmeldungen bzgl. der Integration in die Weiterentwicklung mit einbezogen. Die erarbeiteten Konzepte wurden in einem Demonstrator umgesetzt.

1.4 Stand der Technik

Der Stand der Technik an den angeknüpft wurde, wurde bereits in den Anlagen zum Antrag des Projekts EEExplore ausführlich dargestellt (Referenzen [2] [3] [4] [5] [7] [8]. Dieser änderte sich bis zum Projektstart nicht.

Im Laufe des Projekts verwendete Fachliteratur konnte nationalen und internationalen Fachdatenbanken entnommen werden. Hierbei wurden hauptsächlich Springerlink[1] und IEEE *Xplore* Digital Library[2] genutzt. Weitere Referenzen sind in Abschnitt 3 angegeben.

1.5 Zusammenarbeit mit anderen Stellen

Im Rahmen des Vorhabens erfolgte eine enge Kooperation mit dem Projektpartner FZI. Es fand ein intensiver Austausch auf wissenschaftlich-technischer und auf Projekt-organisatorischer Ebene statt. Während der Projektbearbeitung erfolgte eine ständige Abstimmung mit dem beteiligten Partner. Hierzu wurden unter Leitung von aquintos/FZI Workshops mit anschließender Diskussionsrunde veranstaltet. Themen der Workshops waren beispielhaft „Optimierungsalgorithmen“, „Klassifizierung von EE-Optimierungsproblemen“, „Erfassung der Gesamtheit von Optimierungsaufgaben in der EE-Architekturmodellierung“, „Vorstellung ausgewählten Optimierungsaktivitäten“, „Hierarchisierung von Bewertungskriterien“, „Konzeption einer gesamtheitlichen Optimierung von Elektrik-Elektronik-Architekturen“ und „Explorationsstrategien“. Durch wöchentliche Projekttreffen wurde der Stand und Fortschritt des Projektes erfasst und die weiteren Projektschritte identifiziert.

Zur Außendarstellung und Öffentlichkeitsarbeit wurde eine Projektseite mit Projektbeschreibung unter http://www.aquintos.eu/index.php?page=285 online gestellt. Die Koordination mit dem industriellen Anwenderkreis wurde von aquintos übernommen.

[1] http://www.springerlink.de/

[2] http://ieeexplore.ieee.org

2 Ergebnisse

2.1 Darstellung der erzielten Ergebnisse

2.1.1 Basisanalyse und Spezifikation (AP1)

Inhalte der Basisanalyse und der Spezifikation waren die grundlegenden Nutzeranforderungen an Architekturmodellierung und -optimierung, sowie die Erstellung eines daraus resultierenden Systemspezifikationsdokuments.

2.1.1.1 Anforderungsanalyse (AP1.1)

Zur Analyse der grundlegenden Nutzeranforderungen an die Architekturmodellierung und -optimierung wurden intern und bei Kunden der aquintos GmbH Forschungsprojekte durchgeführt. Von den Kunden wurde durch langfristige Entwicklungsleistungen von Resident Ingenieuren der aquintos GmbH bei der Arbeit mit PREEvision direktes Feedback aus deren Entwicklungsabteilungen und der Industriellen Anwendung gegeben. Viele Details, Daten und Berechnungen durften beim Kunden nur intern verwendet werden, und konnten somit nicht direkt in PREEvision eingebaut werden. Daher war eine Grundanforderung die Flexibilität und Erweiterbarkeit der Modellierung und Optimierung in PREEvision. Für die Kunden muss es möglich sein, geistiges Eigentum (IP) für sich zu behalten und nicht an einen Toolhersteller weiter zu geben.

Der folgende Auszug fasst die wichtigsten identifizierten Anforderungen zusammen:

- Einer der wichtigsten Punkte war die Wartbarkeit der Modelle und der Optimierungsmethoden. Es musste eine Nutzung der gewonnen Erkenntnisse und Methoden über Jahre hinweg möglich sein.
- Aufgrund der großen Datenmengen, die in einem realen Modell enthalten sind, musste auf eine gute Performanz geachtet werden.
- Eine Erstellung von Bibliotheken, um auf einfache Weise häufig benötigte Teile verfügbar zu machen, sowie eine Einbindung von externem Code sollte ermöglicht werden.
- Von den textuellen Metriken sollte so viel als möglich in einen modellierbaren Teil der Metrik umgesetzt werden, was die Übersicht, das Verständnis und Wartbarkeit verbessert.
- Da Modelle mit verschiedenen Varianten erstellt werden und gerade der Vergleich dieser verschiedenen Varianten sehr interessant ist, musste aus einer Metrik leicht über Varianten iteriert und alle möglichen Varianten-basierten Zugriffe auf das Modell ermöglicht werden.
- Eine Metrik sollte in mehrere Teile gegliedert werden können, um Einzelergebnisse zu erhalten und wiederzuverwenden. Durch diese Anforderungen wurde auch eine leichtere Verständlichkeit einer Metrik erreicht, da einzelnen Teile und der Ablauf als Modell vorliegen und grafisch dargestellt sowie modelliert werden konnten.

Die Anforderungsanalyse wurde weiterhin im Dialog mit den Nutzern von PREEvision durchgeführt, um sie fortlaufend mit der Toolentwicklung abzugleichen.

2.1.1.2 Sytemspezifikation (AP1.2)

Die erfassten Anforderungen wurden spezifiziert und hieraus die Systemspezifikation abgeleitet. Als wichtigste Grundanforderungen für eine Nutzung in Forschung und Entwicklung wurden die folgenden Punkte spezifiziert: Wartbarkeit der Modelle, Performanz, modularer Aufbau, modellbasierter Ansatz, Konfiguration von Varianten. Durch eine grafische Benutzeroberfläche musste eine einfache Bedienung und somit eine hohe Benutzerfreundlichkeit erreicht werden. Es wurden zusätzliche Rahmenbedingungen für den Bereich der Architekturexploration und Optimierung festgehalten. Auf Grundlage dieser Spezifikation konnte die Metrikmodellierung in Abschnitt 2.1.2 realisiert werden, die in einer Pilotphase intern und bei Kunden eingesetzt wurde.

2.1.2 Metrikmodellierung (AP2)

Das Ziel der Metrikmodellierung bestand in der Verbesserung der Möglichkeiten im Werkzeug PREEvision und der Gewinnung sowohl eines graphischen Editors als auch der Automation der Ausführung einzelner Metriken.

2.1.2.1 Untersuchung existierender Metriken (AP2.1)

```
#gets the costs and weights of all components
    totCosts = 0
    totWeight = 0
    for a_s in actor_sensors:
        totCosts += util.getMetric("COSTS",a_s)
        totWeight += util.getMetric("WEIGHT",a_s)

    for c in cus:
        costs = util.getMetric("COSTS",c)
        weight = util.getMetric("WEIGHT",c)
        totWeight += weight
        totCosts += costs
```

Abb. 2: Einfache textuelle Metrik

Die existierende Möglichkeit mit Metriken auf das Architekturmodell beliebig zuzugreifen, erfolgte mit Hilfe einer imperativen textuellen Skriptsprache (siehe Abb. 2). Der skript-basierte, textuelle Zugriff auf das meist grafisch dargestellte Modell war teilweise sehr umständlich und erforderte genaue Kenntnisse des dahinterliegenden Metamodells von PREEvision. Komplexere Metriken wurden daher nur selten geschrieben und konnten auch nur sehr schwer bei Upgrades zu neuen Releases gewartet werden. Die Ergebnisse wurden textuell ausgegeben und mussten vom Benutzer detailliert ausgewertet werden. Durch die textuelle Ausgabe war es weiterhin nicht möglich das Ergebnis oder mögliche Modellobjekte eines Ergebnisses als Eingangsdaten für eine weitere Metrik oder für einen automatisierten Report zu nutzen.

Ausgangspunkt für die Entwicklung von Metriken waren vorhandene Ansätze, die existierende Systeme mit Hardware- und Softwaremetriken auswerten. Für EE spezifische Metriken wurden Kategorien identifiziert, um die Metriken z.B. nach Wiederverwendbarkeit, Komplexität oder die Auswertung der Beziehungen von Modellartefakten einzuteilen. Des Weiteren wurden die betrachteten Modellartefakte, Eingangsparameter und Vorbedingungen für das Ausführen einer Metrik untersucht. Für die Darstellung des Metrik-Ergebnisses wurden die möglichen Verwendungszwecke des Ergebnisses oder Teilergebnisses sowie eine potentielle Verwendbarkeit eines Ergebnisses für weitere Metriken am Beispiel eines FlexRay Bussystems evaluiert. Basierend auf den Erfahrungen mit der grafischen Modellierung von EE-Architekturmodellen in PREEvision sollte auch die Darstellung der

Metrikergebnisse grafisch aufbereitet werden und in Diagrammen dargestellt werden können.

Insgesamt wurden ca. 140 Metriken betrachtet. Die Erfahrungen hierbei zeigten, dass Metrikskripte zu einem Großteil aus Datenakquise bestanden. Die Suche und Navigation im Modell sowie die Bereitstellung der benötigten Daten benötigte somit in etwa 70% des Metrikskriptes. Auf den Algorithmus zur Berechnung und Auswertung der Daten entfielen ca. 20% der Quelltextkonstrukte und für Ausgabe wurden meist nur ca. 10% verwendet. Die Datenakquise sollte deshalb durch Modellabfragen und andere Methoden vereinfacht und möglichst gut wiederverwendbar für den Benutzer werden.

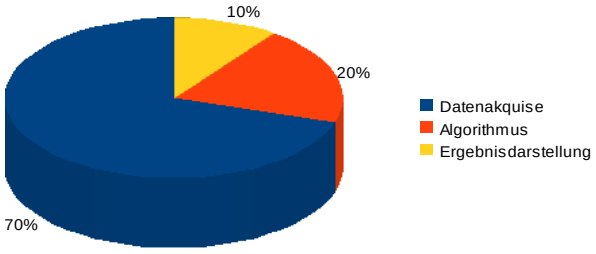


Abb. 3: Codeanteile in einem Metrikskript

Zuvor wurden Metriken in der Skriptsprache Python erstellt. Da PREEvision auf dem in Java implementierten Eclipse-Framework basiert wurde die Ausführung der Python Skripte über Jython realisiert, was zu Geschwindigkeitseinbußen führte. Bei Performanz-Messungen mit anderen Programmiersprachen ergab sich ein Vorteil für in Java geschriebene Metriken von ca. 1000:1 im Vergleich zur Python-Integration von PREEvision. Deshalb sollte neben Python, um kompatibel zu existierende Metriken zu bleiben, auch Java als Programmiersprache in den Metriken Verwendung finden.

2.1.2.2 Entwicklung Metamodell, Notation und Editor (AP2.2)

Zur Realisierung eines graphischen Editors für Metriken musste ein geeignetes Datenmodell für die Metriken geschaffen werden. Das Metrik-Metamodells sollte dabei auf den Grundmetamodellen von PREEvision basieren um eine möglichst enge Bindung der Metamodelle und damit einen einfachen Zugriff auf die Daten sowie eine direkte Integration in PREEvision zu erhalten. Dadurch wird auch bei einem Versionswechsel von PREEvision ermöglicht existierende Metriken eines Kunden direkt in die neue Version zu migrieren.

Die Erkenntnisse der Evaluation aus Abschnitt 2.1.2.1 diente dazu ein Metrik-Metamodell zu entwerfen. Aus den Anforderungen und Zielen der Metriken für EE-Architekturen wurden als Grundelemente der Notation Berechnungsblöcke, Parameterblöcke, Modellabfragen und Ergebnisblöcke herausgearbeitet. Diese Blöcke können verschiedene Zustände und Bedingungen beinhalten und sollten über einen vom Benutzer modellierbaren Datenfluss verbunden werden. Zur Darstellung der Ergebnisse einer Metrik sollte eine spezielle Tabelle entwickelt werden.

Diese herausgearbeiteten verschiedenen Elemente einer Metrik sollten nun in ein Metrik-Metamodell abgebildet werden. Hierzu wurde das Metamodell von PREEvision erweitert. Das Metrik-Metamodell teilt sich hierbei die gleichen Metamodell-Basisartefakte mit dem PREEvision-Metamodell, um eine direkte Integration in das Konzepttool zu ermöglichen.

Zur Eingabe und Modellierung der Metriken sollte nach dem Erstellen des Metamodells ein grafischer Editor entstehen mit dem die Blöcke und Datenfluss komfortabel grafisch modelliert werden können. In Berechnungsblöcken sollte z.B. die Möglichkeit geschaffen werden, um in einer Skriptsprache beliebige Berechnungen durchzuführen, wobei die Eingangsparameter im Skript verfügbar sein sollen und das Ergebnis an den Block und den darauf folgenden Datenfluss weitergegeben werden kann.

Metrik-Metamodell

In Abb. 4 wird der die Kernstruktur des Metrik-Metamodells dargestellt. Manche Blöcke wie z.B. der Berechnungsblock (CalculationBlock) kann, wie über die Vererbungshierarchie zu erkennen ist, als Datenquelle (DataSource) oder als Datensenke (DataTarget) verwendet werden. Zudem gibt es spezielle Ergebnisblöcke (ScaleBlock, LightsBlock, etc.) die nur als Datensenke dienen. Entsprechend kann z.B. ein Parameterblock nur als Datenquelle verwendet werden. Über einen Datenfluss (DataFlow) können dann die jeweiligen Datenquellen und -Senken miteinander verbunden werden.

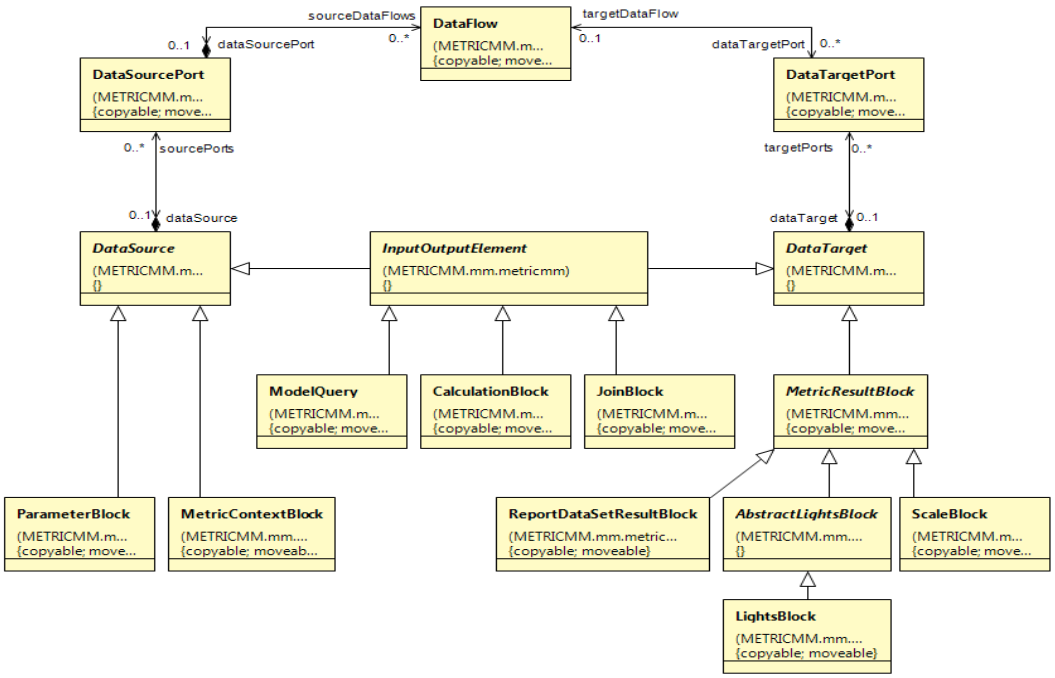


Abb. 4: Das Kern-Metamodell des Metrikeditors

Metrikeditor

Das vorwiegende Einsatzgebiet des Metrikeditors ist die Analyse und Evaluation des aktuell geladenen EE-Architektur-Modells. Die Metrikdiagramme ermöglichen dem Benutzer die Metrik in einer datenflussorientierten Art zu modellieren. Die Blöcke werden hierbei zu unterschiedlichen Aufgaben genutzt: Datenbereitstellungs-, Berechnungs-, Datenverarbeitungs-, algorithmische und Ergebnis-präsentierende Blöcke. Die Berechnungen für die Metriken können in Java oder Python Code geschrieben werden.

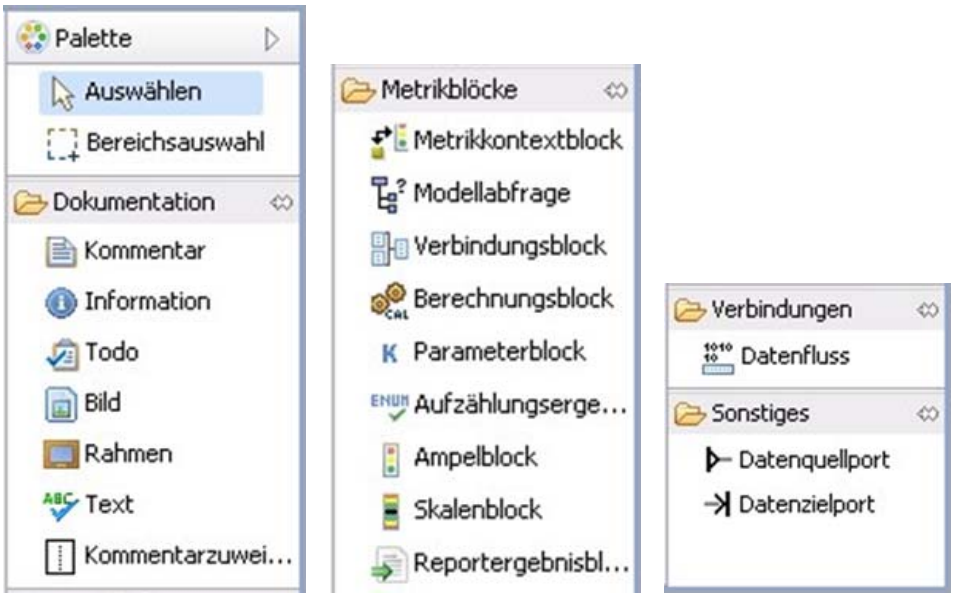


Abb. 5: Metrikeditor-Palette

Im folgenden Abschnitt werden einzelne ausgewählte Blöcke aus der Metrikeditor-Palette beschrieben. Abb. 5 zeigt die Metrikeditor-Palette. Diese gliedert sich in mehrere Kategorien, um hierdurch die Übersicht für den Benutzer zu wahren. Als Kategorien stehen zur Auswahl: Dokumentation, Metrikblöcke, Verbindungen und Sonstiges.

Unter der Kategorie der Metrikblöcke sind folgende Blöcke realisiert worden: Metrikkontextblock, Modellabfrage, Verbindungsblock, Berechnungsblock, Parameterblock, Aufzählungsergebnisblock, Ampelblock, Skalenblock und Reportergebnisblock

Metrikkontextblock

Der Metrikkontextblock (siehe Abb. 6) wird benutzt um Kontext-Modellartefakte als Eingang für Metriken zu setzen.

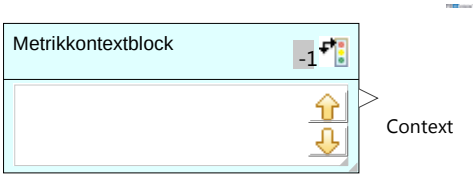


Abb. 6: Metrikkontextblock

Modellabfrageblock

Im Modellabfrageblock (siehe Abb. 7) können mehrere Modellabfragen gruppiert werden. Die Ausführung kann in einem Metrikdiagramm erfolgen, die Ergebnisse sind in dem internen Block ersichtlich. Zur Darstellung von Inhalten, welche die Kapazität der ersichtlichen Liste überschreiten, kann an der Seite des Modellabfrageblocks geblättert werden.

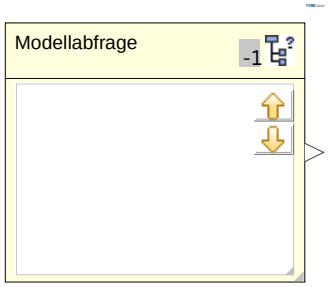


Abb. 7: Modellabfrageblock

Berechnungsblock

Der Berechnungsblock (siehe Abb. 8) ist das zentrale Element in der Metrik. Er ist verantwortlich für die Berechnungen, die in Java- oder Python-Code hinterlegt sind. Des Weiteren kann er einen festen Wert zurückgeben. Welcher Wert zurückgegeben wird, kann mittels zweier Schalter an der Oberseite des Berechnungsblocks eingestellt werden. Der Berechnungsblock kann mehrfache Datenquellen und Datenzielports haben. Für jeden Datenquellport kann eine separate Kalkulation und Ausgang definiert und auf dem Berechnungsblock ausgeführt werden. Jeder Datenquellport wird durch einen internen Block repräsentiert, der das Berechnungsergebnis oder den festen Wert, der für den Port benutzt wird, zeigt.

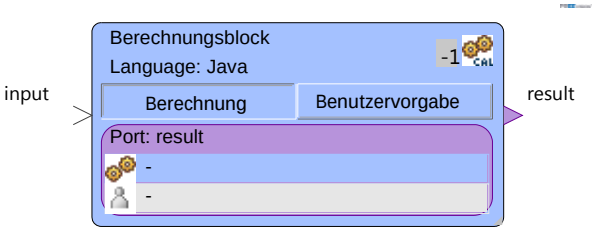


Abb. 8: Berechnungsblock

Parameterblock

Der Parameterblock (siehe Abb. 9) wird genutzt, um feste Eingangsdaten für die Metrik zu setzen.

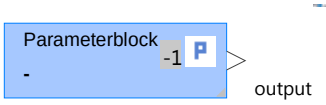


Abb. 9: Parameterblock

Aufzählungsergebnisblock

Die Aufzählungsergebnisblöcke (siehe Abb. 10) werden dazu genutzt um zu überprüfen, ob ein Ergebnis, welches in der Metrik berechnet wurde, ein Teil einer Aufzählung ist. Für den Fall, dass das Ergebnis, welches mit dem Aufzählungsergebnisblock verbunden ist, ein Teil einer Aufzählung ist, wird dies durch eine grüne Umrahmung des Aufzählungsergebnisblocks angezeigt. Ansonsten wird eine rote Umrahmung dargestellt.

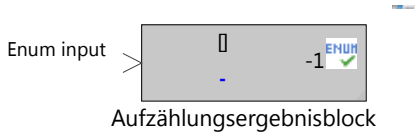


Abb. 10: Aufzählungsergebnisblock

Ergebnisblöcke:

a) Ampelblock

Der Ampelblock (siehe Abb. 11) beinhaltet wie eine Verkehrsampel drei Zustände, mit denen das Ergebnis evaluiert werden kann. Er benutzt drei Datenzielports. Der Eingangsport erhält das Resultat, welches evaluiert werden soll. Die untere und obere Grenzen definieren die Werte bei denen die Ampel von rot beziehungsweise grün nach orange umschaltet.

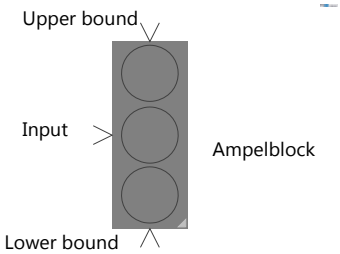


Abb. 11: Ampelblock

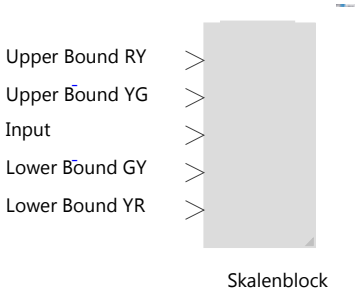


Abb. 12: Skalenblock

b) Skalenblock

Der Skalenblock (siehe Abb. 12) ist ein weiterer Ergebnisblock. Er agiert ähnlich zu dem Ampelblock, kann allerdings das Ergebnis besser evaluieren, da zwei Ampelblöcke vereint wurden.

Reportergebnisblock

Der Reportergebnisblock (siehe Abb. 13) ist ein auf Kundenbedürfnisse anpassbarer Ergebnisblock. Der Benutzer kann so viele zusätzliche Datenquellports wie benötigt anlegen, um ein entsprechende Ergebnisstruktur zu erhalten. Der Reportergebnisblock zielt darauf hin ab, als Ausgabe für einen Report zu dienen.

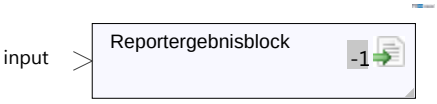


Abb. 13: Reportergebnisblock

Datenfluss:

Unter der Kategorie Verbindungen ist der Datenfluss ersichtlich. Der Datenfluss modelliert den Fluss der Daten zwischen den einzelnen Metrikblöcken. Datenflüsse werden an Ports, die in der Metrikeditor-Palette bereitgestellt werden, angeschlossen.

Datenquellport:

Der Datenquellport wird dazu genutzt, Daten, welche der Block generiert hat bereitzustellen.

Datenzielport:

Der Datenzielport wird dazu genutzt, eingehende Daten von einem Datenfluss anzunehmen und diese für weitere Verarbeitung bereitzustellen.

Darstellungselemente im Metrikeditor

Der Status eines Blockes innerhalb des Metrikeditors wird zusammen mit der Berechnungsreihenfolge oben rechts im Block dargestellt (siehe Abb. 14).

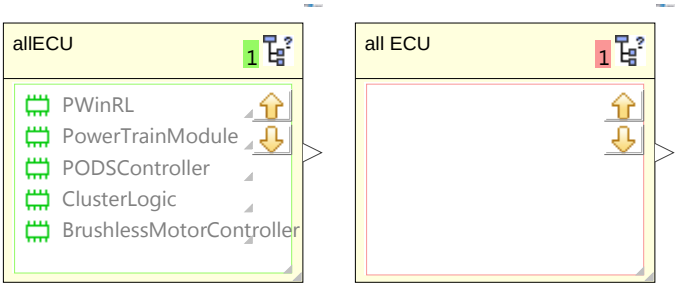


Abb. 14: Ausführungsstatus von Metrikblöcken

Eine „-1“ symbolisiert einen undefinierten Zustand. Nach einer Ausführung der Metrik stellen die angezeigten Zahlen die Berechnungsreihenfolge dar. Wird die Zahl grün hinterlegt war die Berechnung dieses Blockes erfolgreich, bei Problemen werden Informationen dazu in der Informationsansicht ausgegeben und die Zahl rot hinterlegt.

Bei der Selektion eines Blockes oder Datenflusses im Metrikdiagamm werden die jeweiligen Vorgänger und Nachfolger hervorgehoben um schnell einen Überblick der Abhängigkeiten zu erhalten.

Beispiel einer Metrik zur Berechnung und Überprüfung von Leitungslängen mit Reportgenerierung:

Links in der Abb. 15 ist ein Modellabfrage-Block wiringHarnessStates ersichtlich. Hierbei wurden mehrere Modellabfragen gruppiert. Die Ergebnisse der Modellabfragen werden über Datenflüsse als erstes an den Berechnungsblock Wiring Harness States weitergegeben und nachfolgend über die Berechnungsblöcke ExtractLongestWire sowie ExtractShortestWire weiterverarbeitet. Als grafische Darstellung der Ergebnisse wurden zwei Ampelblöcke für das Ergebnis der kürzesten und längsten Leitung genutzt. Durch die Ampelblöcke wird auf einfache Art ersichtlich ob die festgelegten Grenzwerte einhalten wurden. Durch zwei Reportergebnisblöcke wurden die Ergebnisse der Metrik festgehalten und vom Benutzer vordefinierte Dokumente generiert.

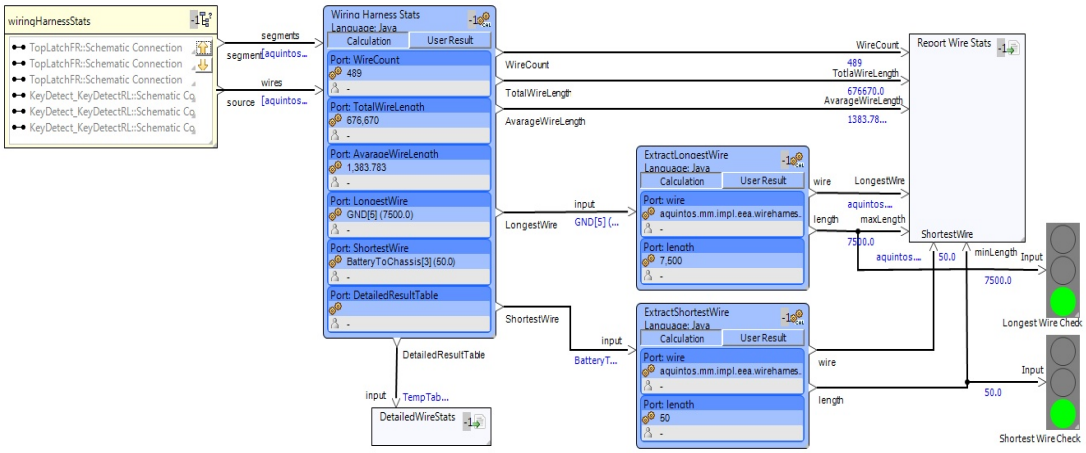


Abb. 15: Beispiel: Metrik zur Berechnung und Überprüfung von Leitungslängen mit Reportgenerierung

2.1.2.3 Entwicklung Automation der Metrikausführung (AP2.3)

Benutzer-Interaktion zur Ausführung der Metriken:

Für den Benutzer stehen mehrerer Möglichkeiten mit den Metriken zu interagieren zur Auswahl. Über das Kontextmenü eines Blocks im Metrikeditor (siehe Abb. 16) können verschiedene Aktionen gewählt werden. So können markierte Metrikelemente zurückgesetzt und/oder neu berechnet werden. Aber auch das Quellcode-Modellelement des aktuell ausgewählten Elements in der Modellansicht kann ausgewählt werden. Ebenso kann das ausgewählte Ergebnis an die Informationsansicht weitergegeben werden.

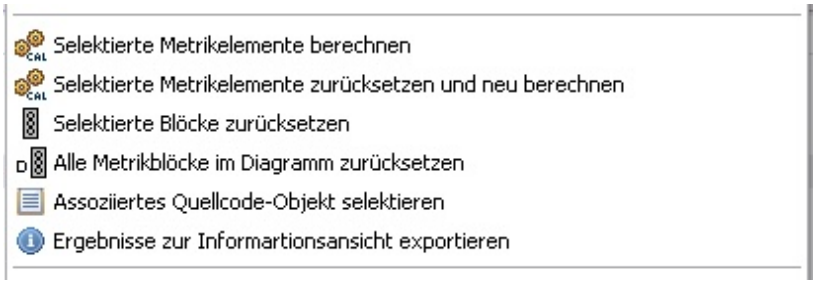


Abb. 16: Kontextmenü zur Ausführung der Metriken

Berechnungsreihenfolge ermitteln

Sind mehrere Berechnungsblöcke miteinander verbunden, so wird automatisiert durch einen Algorithmus analysiert welche Berechnungsreihenfolge eingehalten werden muss. Zyklen können nicht berechnet werden, sondern führen zum Abbruch der Berechnung. Es werden somit die Blöcke zuerst berechnet die von keinem anderen unberechneten Block abhängig sind. Unabhängige Teilergebnisse können gleichzeitig berechnet werden, wodurch auf Multiprozessor-Computern eine bessere Performanz erreicht wird. Nach der Berechnung eines Blocks steht das Ergebnis an seinem Ausgangsport zur Verfügung. Die Berechnung der jeweils abhängigen Blöcke wird so lange durchgeführt bis der Block berechnet ist auf dem der Benutzer die Ausführung gestartet hatte.

2.1.3 Modellbasierte Architekturbewertung (AP3)

Aufbauend auf der Metrikmodellierung in Abschnitt 2.1.2 und den bestehenden Datenmodellen in PREEvision sollten Methoden zur modellbasierten Bewertung ganzer Fahrzeugarchitekturen untersucht und entwickelt werden.

2.1.3.1 Erweiterung Metrikmodellierung um Metrikfusion (AP3.3)

Eine Metrikfusion von mehreren Metrik-Diagrammen zu Berechnung eines Gütemaß konnte mit der CPRM (capped reference point method) [9] erreicht werden. Eine Implementierung der CPRM stellte dabei ein eigenes Metrik-Diagramm dar, welches ein Gütemaß für die Architektur lieferte.

Die CRPM wurde dabei im Metrikeditor durch Berechnungs- und Parameterblöcke umgesetzt. Die Designvariablen wurden dabei über einen Eingangsport entgegengenommen. Die Schwellenwerte Q^u, Q^a, Q^r und Q^n wurden über Parameterblöcke dem Berechnungsblock zur Verfügung gestellt. Dieser stellte dann das berechnete Gütemaß zur Verfügung. Da der Berechnungsblock jedoch eine Whitebox darstellt, ist es wünschenswert, dass die Implementierung geschützt werden kann und vom Benutzer nicht zu modifizieren ist. Dazu wurde der Metrikeditor entsprechend erweitert. Um nicht für jede Speziallösung wie die CPRM einen Typen im Metrik-Metamodell einzuführen werden solche Erweiterungen über benutzerdefinierte Blöcke realisiert, deren Implementierungen über externe Programmmodule (Plugins) erfolgt.

2.1.3.2 Realisierung in Metrikeditor (AP3.4)

Die CRPM Berechnung konnte durch existierenden Blöcke des Metrikeditors wie in Arbeitspaket 2.1.2.2 beschrieben durchgeführt werden. In diesem Bereich wurden noch einige Verbesserungen implementiert. Für die Benutzerdefinierte Blöcke wurde das Metamodell entworfen (siehe Abb. 17).

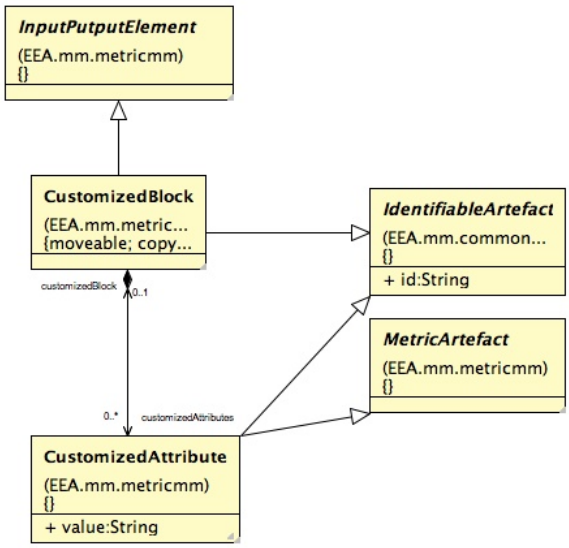


Abb. 17: Metamodellteil des CustomizedBlock

Der externe Programmcode für den Benutzerdefinierten Block wird über den Erweiterungspunkt aquintos.metric.engine.customizedBlock (siehe Abb. 18) als Plug-In integriert.

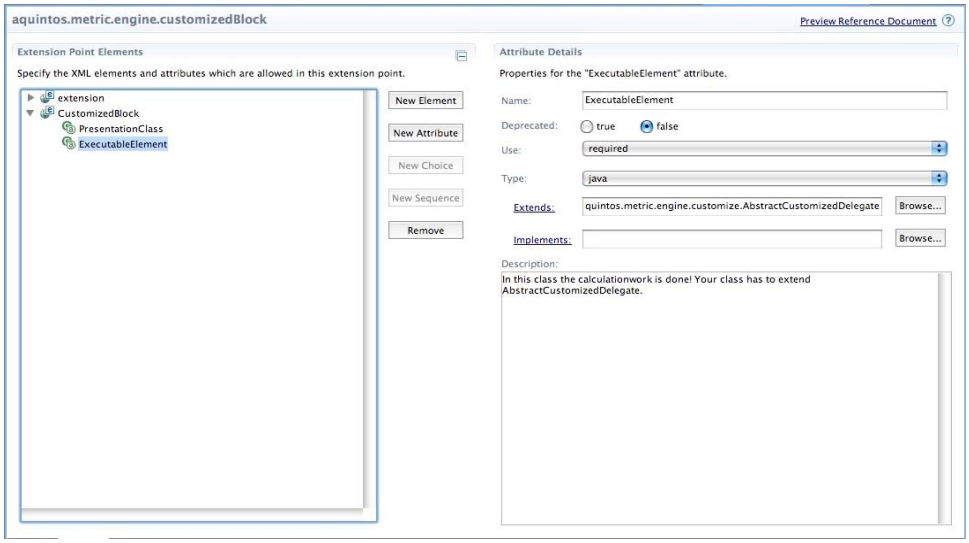


Abb. 18: Erweiterungspunkt des CustomizedBlock

Erweiterung Berechnungsblock

Erweiterungen der Berechnungsblöcke um mehrerer Ausgangsports und entsprechende Highlights wurden umgesetzt, um somit Metriken flexibler und einfacher gestalten zu können.

Der Berechnungsblock (siehe Abb. 19) ist das zentrale Element in einer Metrik. Er ist für Berechnungen verantwortlich, die in Java oder Python Quellcode vorliegen. Des Weiteren kann er einen fixen Wert zurückgeben. Welcher Wert zurückgegeben wird, wird über zwei Schaltflächen, die sich im oberen Bereich des Berechnungsblocks befinden, eingestellt. Der Berechnungsblock kann mehrere Datenquell- und Datenzielports erhalten. Für jeden Datenquellport kann eine separate Berechnung und ein Ausgang festgelegt werden. Repräsentiert wird jeder Datenquellport durch einen internen Block, der die Berechnungsergebnisse oder einen festen Wert, der für diesen Port genutzt wird, anzeigt. Sollten mehrere Datenquellports genutzt werden, müssen sie in der Berechnungsmethode von diesem Block berücksichtigt werden.

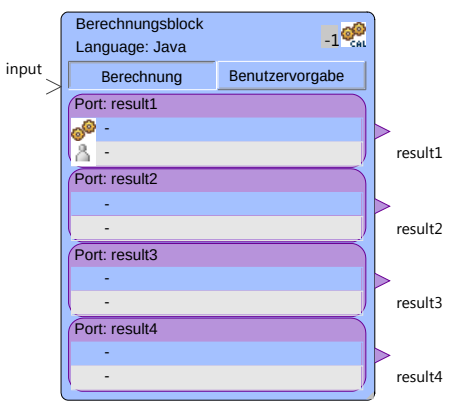


Abb. 19: Erweiterung Berechnungsblock: mehrere Ausgangsports

Die in Abb. 20 gezeigte Beispiel-Metrik enthält einen Berechnungsblock mit mehreren Datenquellports. Die Metrik dient der Berechnung einer Busauslastung. Die zwei Datenquellports sind hierbei FirstBusMax und BusLoadTable.

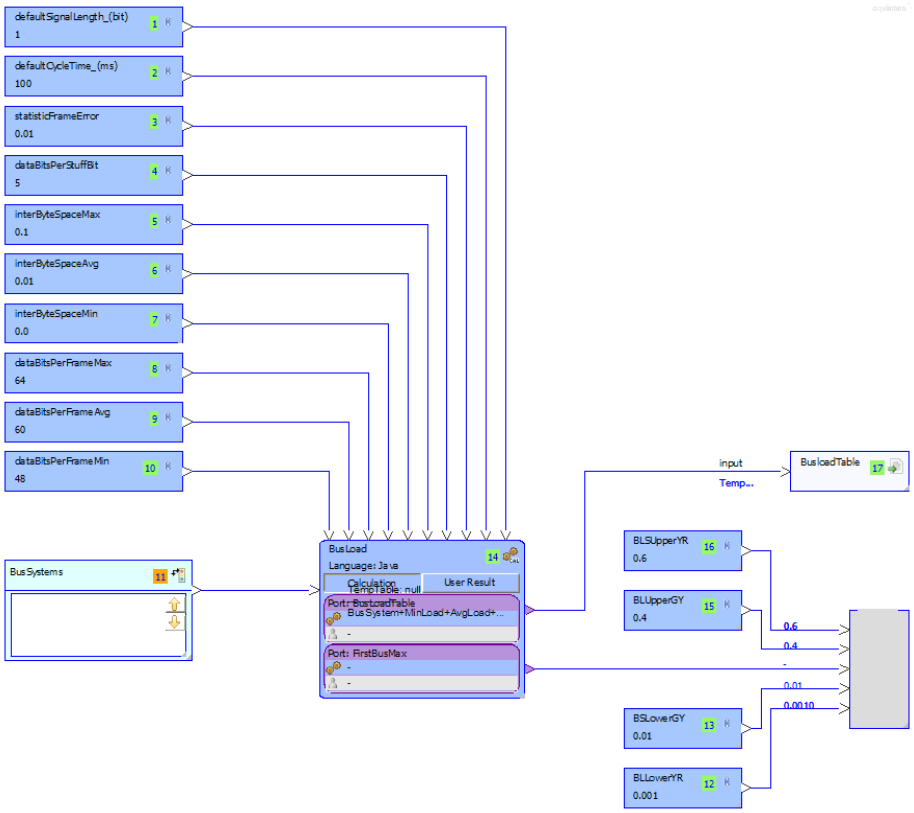


Abb. 20: Beispiel: Berechnungsblock mit mehreren Datenquellports

Die Erweiterung der Metrikmodellierung um eine Metrikfusion ist nicht direkt umsetzbar. Hintergrund ist, dass die Original Equipment Manufacturers (OEMs) eigene Bewertungskriterien festlegen, die zur Beurteilung der Fahrzeugarchitekur dienen. Das Wissen basiert auf Erfahrungen der OEMs aus vorhergehenden Modellreihen. Ziel war daher Methoden und Werkzeuge den Architektur-Entwicklern zur Verfügung zu stellen. Zudem konnte eine Analyse durchgeführt werden, inwiefern einzelne Optimierungsaufgaben sich auf bestimmte Qualitätskennzahlen auswirken. Dies diente wiederum dazu, die Zusammenhänge einer Gesamtgütekennzahl zu erfassen. Gewichtungen der einzelnen Kennzahlen konnten im Zuge dieses Projektes nicht eingebracht werden. Anstelle dessen wurde eine genauere Analyse der Teilaspekte, die von einer Architektur besonders profitieren durchgeführt, siehe hierzu Abschnitt 2.1.4.1.

Zusätzlich wurde die Palette des Metrikeditors um die folgenden Blöcke erweitert, die im Folgenden näher vorgestellt werden: Sortier- und Filterblock, Schleifenblock, Stromabfrageblock, Stromberechnungsblock und Variantenwechselblock. Hiermit stehen dem Benutzer ein Vielzahl von Möglichkeiten zur Verfügung Qualitätskennzahlen seinen Anforderungen entsprechend zu berechnen.

Sortier- und Filterblock

Der Sortier- und Filterblock (siehe Abb. 21) wird genutzt, um Daten innerhalb der Metrik zu sortieren und/oder zu filtern.

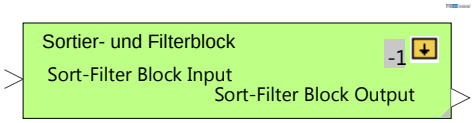


Abb. 21: Sortier- und Filterblock

Schleifenblock

Der Schleifenblock (siehe Abb. 23) wird genutzt, um einen Container zu modellieren, der in einer Schleife ausgeführt wird, solange ein festgelegtes Abbruchkriterium nicht erfüllt ist.

Zudem besteht die Möglichkeit eines manuellen Abbruchs der Iterationen des Schleifenblocks durch das Drücken der ESC-Taste, um somit bei einem fehlerhaften Aufbau oder einer Endlos-Schleife ein hartes Beenden der Metrikausführung zu verhindern.

Der Schleifenblock stellt Datenziel- und Datenquell-Delegationsports zur Verfügung, sowie einen internen Start- und Endport. Zudem besteht die Möglichkeit Schleifenblöcke ineinander zu verschachteln.

Startport für Interne Schleifen

Für Berechnungsblöcke können interne Schleifen erstellt werden. Die internen Schleifen müssen im Berechnungsblock-Quellcode konfiguriert, gestartet und ausgelesen werden. Dies ist der Startport für interne Schleifen. Beim Anlegen eines Startports für interne Schleifen wird ein Dialog-Fenster zur Eingabe des Namens der neuen Schleife geöffnet (siehe Abb. 22)

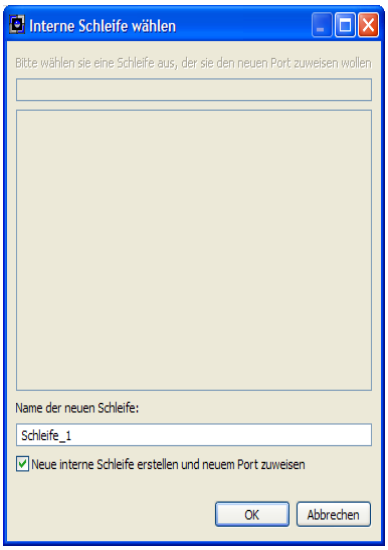


Abb. 22: Auswahl der internen Schleife

Zielport für interne Schleifen:

Der Zielport für interne Schleifen repräsentiert die Daten.

Interner Datenquellport:

Jeder Schleifenblock benötigt exakt einen internen Datenquellport, der den Anfang der Schleife, die durch diesen Block modelliert ist, angibt.

Interner Datenport

Der interne Datenport repräsentiert weitere eingehende Datenports für einen Schleifenblock.

Interner Endport

Der Interner Endport zeigt das Ende eines Zyklus des Schleifenblocks an. Ein Schleifenblock kann mehrere Interne Endports enthalten.

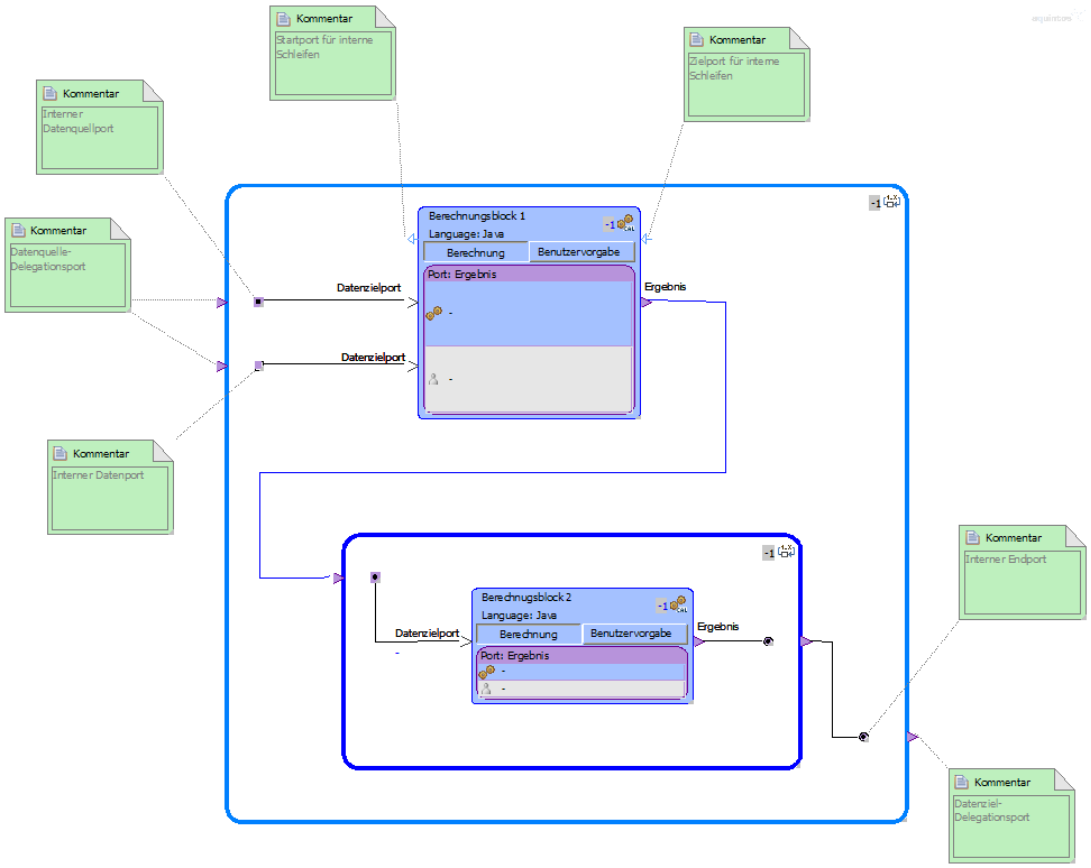


Abb. 23: Beispielhafte Darstellung eines Schleifenblocks

Stromabfrageblock

Der Stromabfrageblock (siehe Abb. 24) wird genutzt, um Elemente zu erhalten, die für den Stromberechnungsblock für die Berechnung des Stroms benötigt werden. Man kann diesen Block benutzen, um die benötigten Pins festzustellen oder um einen eigenen Algorithmus zu implementieren, um wiederum die benötigten Elemente zu erhalten, die als Eingang für den Stromberechnungsblock benötigt werden.

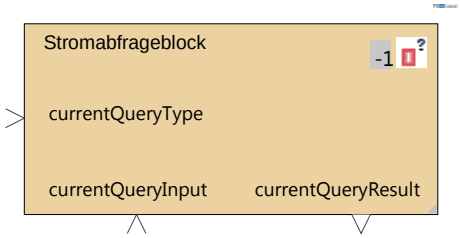
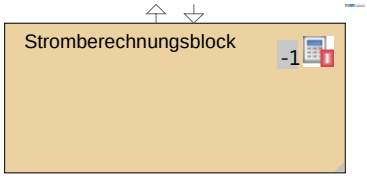


Abb. 24: Stromabfrageblock

Stromberechnungsblock

Der Stromberechnungsblock (siehe Abb. 25) berechnet die Ströme für die Pins, die vom Stromabfrageblock bereitgestellt wurden.

Abb. 25: Stromberechnungsblock



Variantenwechselblock

Mit dem Variantenwechselblock (siehe Abb. 26) können Varianten während der Evaluation einer Metrik gewechselt werden.

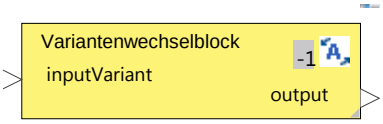


Abb. 26: Variantenwechselblock

2.1.4 Entwurfsraumexploration und Architekturoptimierung (AP4)

Aufbauend auf einer umfassenden Analyse bestehender Forschungsaktivitäten in diesem Umfeld wurden Methoden zur teilautomatisierten Entwurfsraumexploration und Architekturoptimierung entwickelt.

2.1.4.1 Identifikation geeigneter Architekturaspekte (AP4.2)

Parallel zur wissenschaftlich-technischen Basisanalyse wurden die für eine Exploration und Optimierung geeigneten Teilaspekte der gesamten E/E-Fahrzeugarchitektur identifiziert. Eine Auflistung von Aspekten, die besonders von einer Architektur profitieren ist nachfolgend aufgeführt:

- Machbarkeit,
- Kosten
- mechanische Eigenschaften
- Elektrik/Elektronik
- Dokumentation und Qualität

Jede der sechs Kategorien enthält eine Vielzahl an Unterkategorien, die wiederum untergliedert und damit verfeinert werden konnten. Beispielhaft sei dies für die Gesamtkosten anhand der Materialkosten der Komponenten dargestellt.

Aber es gibt auch Modellteile, die während des Optimierungsvorgangs invariant bleiben sollen. Die Entscheidungen werden hierbei wiederum vom OEM getroffen. So kann es für

diesen sowohl aus technischer als auch aus kosten-technischer Sicht günstig sein, zum Beispiel die Anzahl der im Fahrzeug verbauten Steuergeräte festzulegen oder klar zu definieren welche Maße für Bauräume angedacht sind.

2.1.4.2 Prototypische Realisierung für Architekturmodelle (AP4.5)

Eine erstellte Spezifikation für Architekturoptimierung und das Konzept für die ergonomische Benutzerführung bildeten die Grundlage zur Umsetzung der prototypischen Realisierung des Leitungssatzrouters. Der Leitungssatzrouter dient zur Optimierung von Architekturmodellen und wurde dahingehend ausgewählt, da diese Optimierung bei fast jeder Änderung des E/E-Architekturmodells ausgeführt werden muss [10][11][12]. Die Umsetzung des Leitungssatzrouters erfolgte innerhalb des Metrikeditors als Customized Block (siehe Abb. 27), da dieser wie bereits erwähnt bei fast jeder Modelländerung das Leitungssatzrouting angestoßen werden muss. Eine Modelländerung wäre hierbei zum Beispiel das Verschieben einer Komponente innerhalb der Topologie (siehe hierzu auch nachfolgendes Beispiel in Abb. 28).

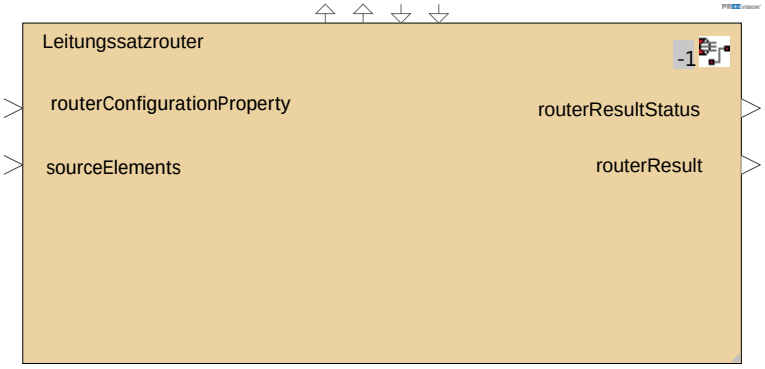


Abb. 27: Customized Block Leitungssatzrouter

Die in der Spezifikation festgelegten Ports wurden wie folgt umgesetzt:

- Datenzielport „routerConfigurationProperty“ als Konfigurationsport
- Datenzielport „sourceElements“ als Eingangsport für zu routende Elemente
- Datenquellport „routerResult“ als Port für die Ergebnisse des Leitungssatzrouters
- Datequellport „routerResultStatus“ als Statusport für die Ausgabe von Fehlermeldungen
- Optionale Ports:
    - Startport für interne Schleifen „routerAlgorithmInit“ zur Initialisierung des Routing-Algorithmus
    - Startport für interne Schleifen „routerEdgeWeightingInit“ zur Initialisierung der Kanten-Gewichtungsfunktionen

Im Nachfolgenden wird eine Optimierungsmetrik zur Aufteilung von Steuergeräten vorgestellt, in welcher der Leitungssatzrouter zum Einsatz kommt:

Eine Optimierungsaktivität innerhalb der E/E-Fahrzeugentwicklung stellt beispielsweise die Aufteilung von Steuergeräten dar. Hierbei wird ein Steuergerät in zwei/mehrere aufgeteilt und beide/alle innerhalb der Topologie neu platziert (siehe hierzu Abb. 28). Durch diesen

Optimierungsschritt können Einsparpotentiale aufgedeckt werden, die sich in den Bewertungskriterien wie Kosten, Gewicht, Leitungslänge, etc. niederschlagen können.

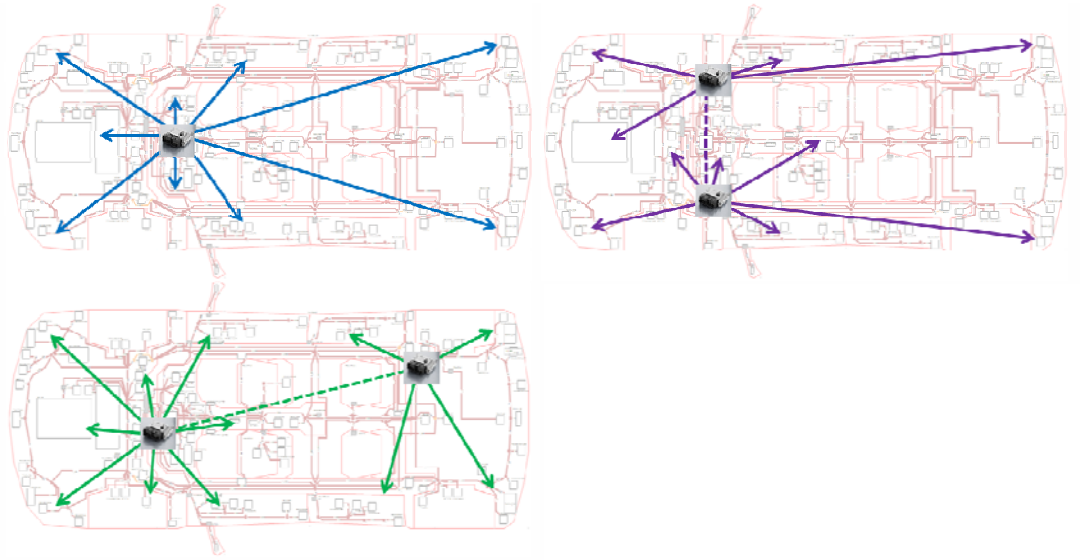


Abb. 28: Unterschiedliche Vernetzungskonzepte eines Fahrzeugs (vereinfacht)

Die hierzu gehörende Optimierungsmetrik gliedert sich in drei große Bereiche, siehe hierzu auch Abb. 29:

- Datenvorbereitung
- Permutationen auf das Modell anwenden, Leitungssatzrouting und Bewertung der Ergebnisse
- Anwendung des besten Ergebnis auf das Modell, Leitungssatzrouting und Nachbereitung

Hierbei ist auch der umgesetzte Customized Block des Leitungssatzrouters ersichtlich.

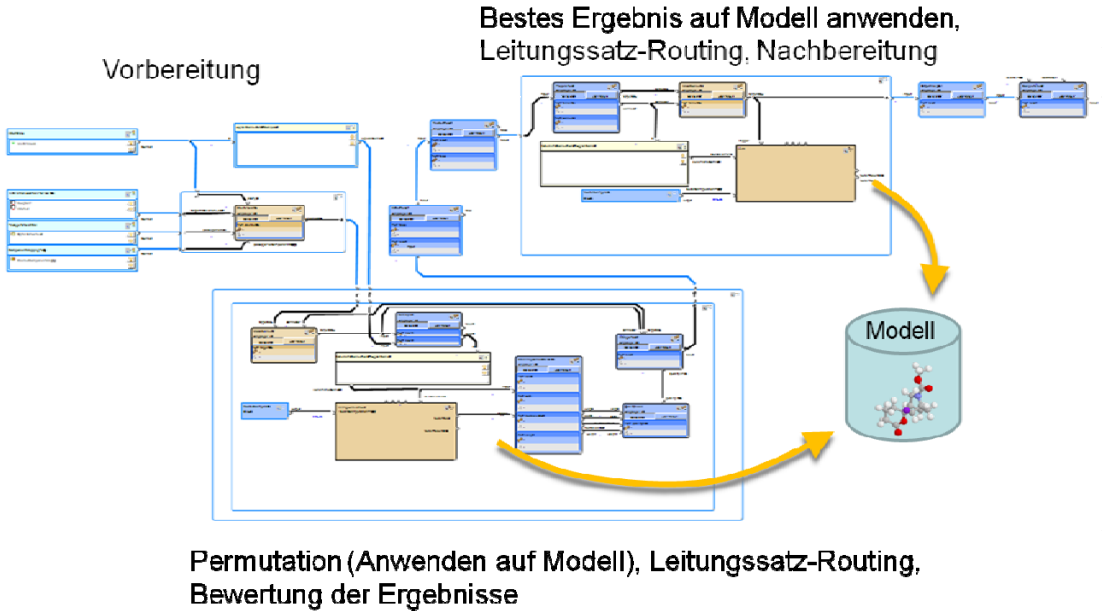


Abb. 29: Optimierungsmetrik zur Aufteilung von Steuergeräten

Des Weiteren wurde ein Signalrouter implementiert. Dieser ist ein automatischer Mechanismus um Kommunikation (Signal-Transmissionen, Frame-Transmissionen und Gateway-Routings) auf Hardwareebene bereitzustellen, basierend auf einer Funktions-Architektur mit einer Verteilung auf Hardwarekomponenten.

Bevor der Signalrouter eine valide Buskommunikation generieren kann, müssen bestimmte Voraussetzungen erfüllt sein. Die Funktionsnetzwerk- (FN), Netzwerk- (NET) und Typenebene (TE) müssen modelliert sein. Zudem müssen Mappings zwischen FN und NET-Ebene angelegt werden.

Der Signalrouter wurde wie bereits der Leitungssatzrouter als Customized Block in den Metrikeditor eingebracht. Abb. 30 zeigt den Customized Block des Signalrouters. Er verfügt wie der Leitungssatzrouter über die Eingänge routerConfigurationProperty und sourceElements sowie die Ausgänge routerResultStatus und routerResult. Startports für interne Schleifen wurden ebenfalls integriert.

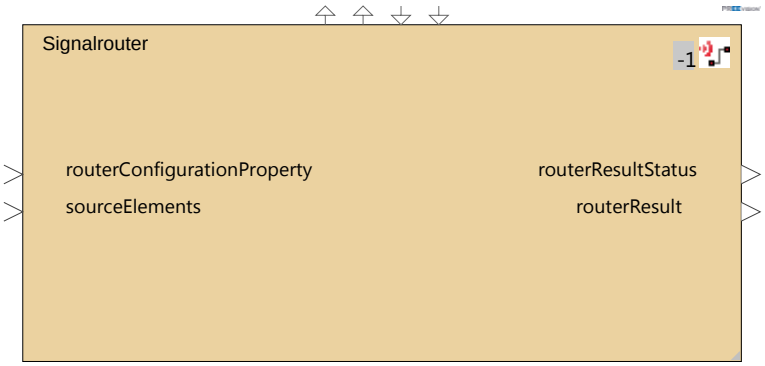


Abb. 30: Customized Block Signalrouter

2.1.5 Architekturpattern (AP5)

Ziel war die Identifizierung von Entwurfsmuster (Pattern), analog zu entsprechenden Methoden im Software Engineering, welche Erfahrungen der Modellierer bezüglich erfolgreicher Lösungsansätze zur Architekturentwicklung formalisieren.

2.1.5.1 Analyse typischer Entwurfsmuster (AP5.1)

Zur Evaluation des Ist-Zustandes wurden typische und bei der Arbeit mit PREEvision genutzte Entwurfsmuster untersucht. Diese Entwurfsmuster stellen bewährte und tragfähig abstrahierte Lösungsansätze bei der Erstellung des Architekturmodells dar. Sie sind Schablonen für wiederkehrende Entwurfsprobleme.

Durch Untersuchungen stellte sich heraus, dass man Unterscheidungen hinsichtlich Entwurfsregeln, die eine Abfolge von Wenn-Dann-Regeln beschreiben, und Templates treffen muss.

Ein Beispiel für eine Wenn-Dann-Regel wäre, dass nach einem abgeschlossenen Leitungssatzrouting eine Platzierung der Trennstellen erfolgen muss. Ein Beispiel für ein Template wäre, dass eine hinzugefügte Trennstelle im Leitungssatz ein Auftrennen der Leitungen impliziert. Als ein weiteres Template kann folgender Punkt angesehen werden: Das Hinzufügen eines weiteren Steuergerätes in die Topologie des Fahrzeuges erfordert eine Anbindung des Steuergerätes an die Leistungsversorgung.

Entwurfsmuster Wiederverwendung

Ein typisches bei der Arbeit mit PREEvision genutztes Entwurfsmuster ist die Wiederverwendung. Das Entwurfsmuster musste ein tragfähiger abstrahierter Lösungsansatz sein, um die Wiederverwendung von Komponenten zu unterstützen. Dieser wurde in Zusammenarbeit mit wichtigen Kunden im Rahmen des industriellen Anwenderkreises erfasst.

Ein Beispiel für das Entwurfsmuster Wiederverwendung ist das Türsteuergerät. Dieses befindet sich sowohl in der Türe auf der Fahrerseite als auch auf der Beifahrerseite, und gegebenenfalls in den Fondtüren. Eine Wiederverwendung soll beim internen Aufbau der ECU vollzogen werden, da hier die wiederkehrenden Strukturmerkmale groß sind. Die Unterschiede der Steuergeräte beruhen lediglich in der unterschiedlichen Konfiguration, zum Beispiel durch Jumper oder Software-Stände, den Sachnummern und zusätzlich in der Verschaltung sowie Vernetzung nach außen. Diese Unterschiede bilden den Kontext der Wiederverwendung.

Um das Entwurfsmuster Wiederverwendung sinnvoll zu verwenden, war eine automatisierte Synchronisation zwischen den wiederverwendeten Einheiten notwendig. Werden an wiederkehrenden Elementen Veränderungen vorgenommen, so werden diese automatisiert mit allen Wiederverwendungen synchronisiert. Ein Beispiel hierfür ist die Änderung eines Bausteins innerhalb der ECU wie den CAN-Transceiver.

2.1.5.2 Untersuchung Unterstützbarkeit im Werkzeug (AP5.4)

Die Definition von wiederverwendbaren Einheiten musste weitgehend automatisiert erfolgen. Das heißt, dass ausgehend von einem Objekt alle möglichen Attribute und Relationen, die Teil der wiederverwendbaren Einheit sind, zu anderen Objekten bekannt sein müssen. Im

Umkehrschluss ergibt sich daraus auch der mögliche Kontext der Wiederverwendung, dabei handelt es sich um die Attribute und Relationen, die nicht zur Wiederverwendung gehören.

Diese Entscheidungen können nicht durch den Benutzer vorgenommen werden, da es sich einerseits um zu viele Relationen und Attribute handelt. Und andererseits wäre damit die Einheitlichkeit wiederverwendbarer Einheiten gewährleistet. Um dieser Anforderung gerecht zu werden, musste die Spezifikation einer wiederverwendbaren Einheit auf den Metadaten des Modells vorgenommen werden. Diese werden formal durch ein Metamodell beschrieben. Die dafür verwendete MOF-Notation musste hierzu um das notwendige Konzept für wiederverwendbare Einheiten erweitert werden. Im Codegenerator, welcher aus dem Metamodell ausführbaren Code generiert, sind ebenfalls Erweiterungen notwendig gewesen, um die Konzepte der Wiederverwendung zu realisieren.

Auf Basis dieser beschriebenen Erweiterungen konnte das Entwurfsmuster Wiederverwendung Einzug in die PREEvision-Software halten. Hierzu waren vor allem Anpassungen der Benutzerführung hinsichtlich von Kreation, Manipulation und Löschung von Modellierungsartefakten vorzunehmen. Ebenso sind wiederverwendbare Einheiten in der graphischen Oberfläche so hervorzuheben, dass diese als solche vom Benutzer erkannt werden. Des Weiteren mussten Navigationshilfen entworfen werden, die es dem Benutzer erlauben, alle Wiederverwendungen eines Elements zu identifizieren und anzusteuern.

2.1.6 Systemintegration und Demonstrator (AP6)

Die Projektergebnisse wurden als Plug-Ins in das Entwurfswerkzeug PREEvision integriert. Zudem wurde ein Demonstrator für die Werkzeug-gestützte Entwurfsraumexploration und Optimierung von E/E-Architekturen erstellt.

2.1.6.1 Integration des Metrikeditors (AP6.1)

Der Hauptnutzen von Metriken ist die Analyse und Evaluation des derzeitig geladenen Architekturmodells [13]. Die Metrikdiagramme sind die zentralen Orte an denen Metriken erstellt und editiert werden. Dies erlaubt dem Benutzer einen Überblick über die modellierte Analyse zu erlangen. Sie erlauben hierbei dem Benutzer die Metriken in einer datenfluss-orientierten Art zu modellieren, wobei unterschiedliche Blöcke über Ein- und Ausgangsport miteinander verbunden werden. Als zweiter Hauptnutzen seien unterstützende Blöcke wie der Reportgenerator erwähnt.

Sowohl Editor als auch Metrikausführung und Ergebnisaufbereitung wurden in das Werkzeug integriert. Die Integration erfolgte ausgehend von der Realisierung des Metrikeditors in Abschnitt 2.1.2.2 und der Metrikausführung in Abschnitt 2.1.2.3, so dass eine Nutzung kommerziell vermarktbarer Qualität verfügbar gemacht wurde.

Übersicht Metrikeditor

In Abb. 31 ist der Metrikeditor ersichtlich. Auf der linken Seite erkennt man die Modellansicht als Baumstruktur aufgebaut, in der Mitte ist der Diagramm-Arbeitsbereich, in dem die Metriken erstellt werden können und auf der rechten Seite ist die für den Metrikeditor spezifische Palette. Die Palette stellt hierbei die in Abschnitt 2.1.3.2 vorgestellten Blöcke zur Verfügung.

Der Diagramm-Arbeitsbereich wird genutzt um den Inhalt des Diagramms anzuzeigen. Dieser benötigt gewöhnlich den meisten Platz im Editor-Arbeitsbereich. Mit den Tools, die in der Palette abgelegt sind, können die Elemente im Diagramm-Arbeitsbereich verändert werden. Der Diagramm-Arbeitsbereich unterstützt durch ein einstellbares Gitternetz, in das die grafischen Elemente beim Platzieren einrasten.

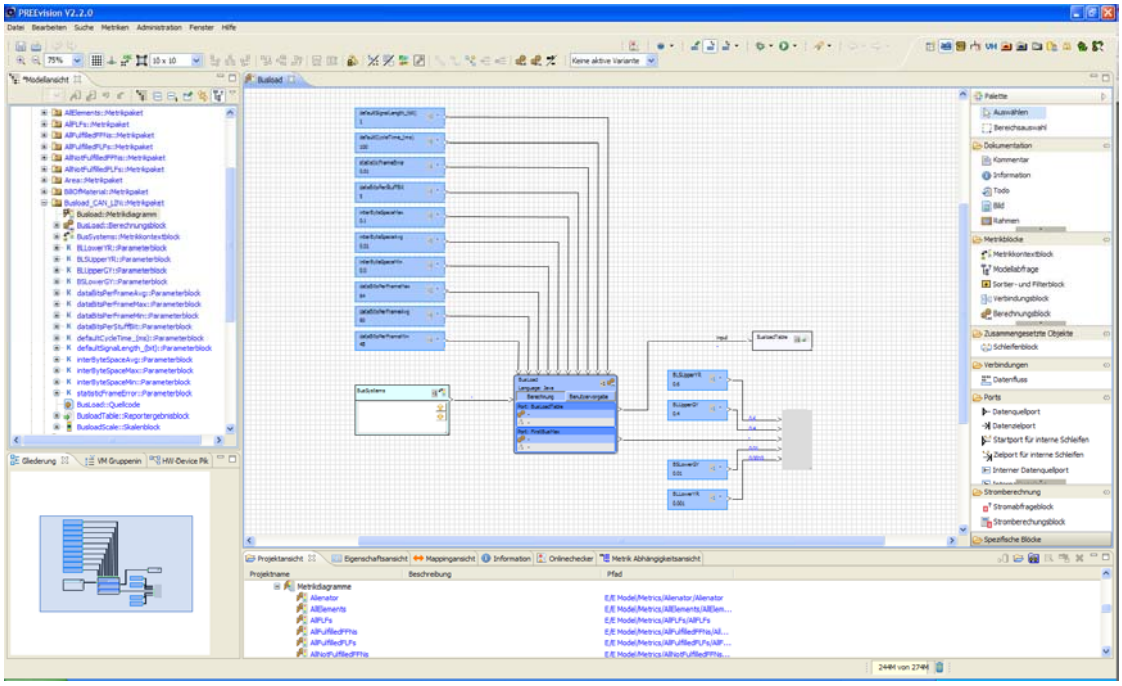


Abb. 31: Metrikeditor

Die Realisierung lehnte sich hierbei an das bisherige Konzept zur Metrikausführung an, so dass gemischt graphisch modellierte und textuell skriptbasierte Metriken eingesetzt werden können.

Die Überprüfung der Integration des Metrikeditors erfolgte über aquintos-spezifische Test, Fallbeschreibungen und der Evaluation von Laufzeitbeschreibungen. Die Integration des Metrikeditors in das Tool PREEvision wurde vollständig vorgenommen und wurde nach den zahlreichen unterzogenen Test zur Pilotphase freigegeben.

Metrikdiagramme

Metrikdiagramme können im Metrikmodell angelegt werden (siehe Abb. 32). Zusätzlich stehen zur Strukturierung Metrikpakte zur Verfügung, die ineinander verschachtelt werden können. Die Erstellung eines Metrikdiagramm erfolgt über ein Kontextmenü auf dem Metrikpaket.

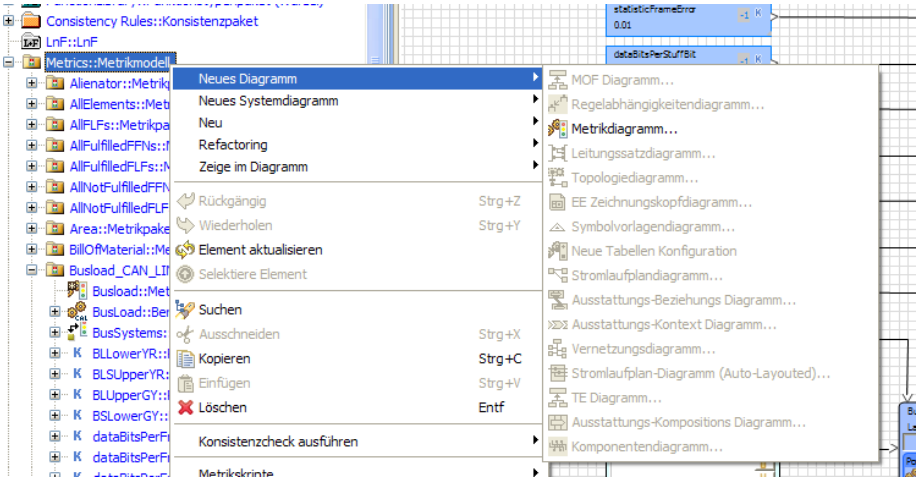


Abb. 32: Erstellung eines Metrikdiagramms

Das Öffnen von Metrikdiagrammen kann zum einen über den in Abb. 32 abgebildeten Modellbaum, zum anderen über die Projektansicht erfolgen.

2.1.6.2 Integration Gesamtmodellgütemaß (AP6.2)

Die erarbeiteten Ergebnisse aus der Untersuchung der Integrierbarkeit in ein Gesamtgütemaß für die Bewertung von E/E-Architekturen erarbeiteten wurde innerhalb der Integration des Gesamtmodellgütemaßes integriert.

Für eine umfassende Bewertung von E/E-Architekturen werden unterschiedliche Kriterien herangezogen. Ausgewählte Kriterien wurden durch eine Metrik abgebildet und lieferten ein entsprechendes Ergebnis, welches ein Teilergebnis der Gesamtbewertung darstellt. Die unterschiedlichen Teilergebnisse wurden in geeigneter Weise miteinander verknüpft (Metrikfusion), so dass am Ende das Gesamtergebnis in Form einer Gütezahl dargestellt werden kann. Zur Ermittlung der Gütezahl wurden mehrere geeignete Methoden recherchiert und analysiert. Die Capped Reference Point Method (CRPM) [9] konnte hierbei im Metrikeditor durch Berechnungs- und Parameterblöcke umgesetzt werden.

Als weiterer Punkt wurde der in Abschnitt 2.1.3.2 vorgestellte Berechnungsblock als Modelloperationsblock modifiziert und ermöglicht es auf dem Modell Operationen durchzuführen. Hierbei findet eine Synchronisation der Zugriffe statt, wodurch eine Transaktionssicherheit gewährleistet wird. Zudem werden über einen Security-Layer die Rechte der Veränderungen am Modell überprüft und kontrolliert. Dies stellt eine gezielte Überwachung von Veränderungen dar.

2.1.6.3 Integration Optimierungsalgorithmen (AP6.3)

Die in Abschnitt 2.1.4.2 erhaltenen Softwarekomponente, welche in der Lage ist, (Teil-) Modelle der E/E-Fahrzeugarchitektur zu optimieren wurde in diesem Arbeitsschritt in PREEvision integriert. Hierbei flossen die in der methodischen Integration ermittelten Konzepte zur geeigneten Benutzerführung ein.

Hierbei lässt sich der Leitungssatzrouter aus der GUI über das Kontextmenü (siehe Abb. 33) ausführen oder als ausführbarer Block innerhalb einer Metrik (siehe Abb. 34) einbringen und ausführen. Diese Funktionalität wurde ebenfalls für den Signalrouter bereitgestellt.

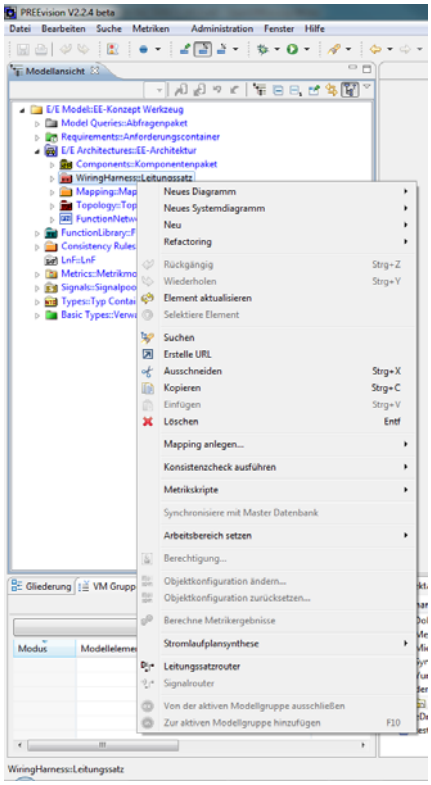


Abb. 33: Aufruf Leitungssatzrouter über Kontextmenü

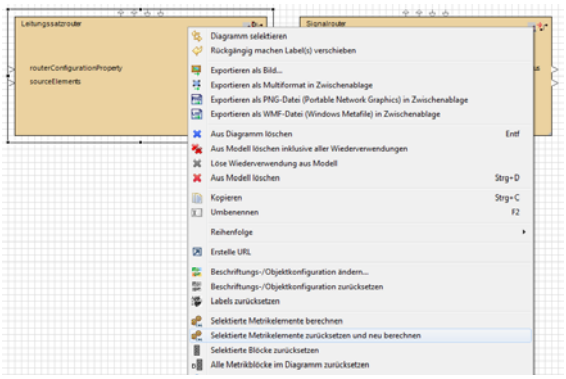


Abb. 34: Aufruf Leitungssatz- oder Signalrouter über Kontextmenü innerhalb des Metrikeditors

2.1.7 Evaluierung (AP7)

2.1.7.1 Systemevaluierung

Für eine Systemevaluierung des entwickelten Metrikeditors wurde dieser inhouse-Tests unter-zogen. Diese entwicklungsbegleitenden Tests während aller Implementierungsschritte stellten eine hohe später verwertbare Qualität der Software-Prototypen sicher.

Dies erfolgte unter anderem auf der Basis von Junit-Tests, welche die Klassen und Methoden automatisiert überprüften. Des Weiteren wurden manuelle Tests zur Verifikation des Metrikeditors ausgeführt.

Zudem befand sich der Metrikeditor bei Kunden bereits in der Pilotphase. Wodurch eine anwendungsorientierte Systemevaluation durchgeführt werden konnte, dadurch dass der Metrikeditor von Entwicklern in einer Produktivumgebung vor Ort getestet wurde.

2.2 Voraussichtlicher Nutzen und Verwertbarkeit

Die Erkenntnisse aus dem Projekt EEExplore hinsichtlich der implementierten Synthesen des Leitungssatz- und Signalrouters sollen in folgenden Aktivitäten genutzt werden, um weitere Synthesen für EEA bereitzustellen. Hierbei ist angedacht Optimierungen für Splice- und Massestellenplatzierungen bereitzustellen. Unter anderem werden hierbei auch die Ergebnisse aus der Integration von Metrikergebnissen in Tabellen, die Einführung neuer Metrikblöcke und die Automatisierung von Optimierungsläufen auf Basis des Metrikeditors genutzt.

Aquintos bereitet die Markteinführung eines Produktes für die oben genannten Optimierungen und Synthesen vor. Dazu gehören zur Zeit Business Development Aktivitäten, um weitere Kundenbedürfnisse zu erfassen und mit den bisherigen Ergebnissen abzustimmen. Ferner werden Businesspläne für das Produkt erstellt und bewertet.

Die bisher erreichten Ergebnisse werden zudem zur Produktreife weiterentwickelt. Dazu gehören Maßnahmen zur Stabilisierung, Performanzsteigerung und zur besseren Bedienbarkeit.

2.3 Fortschritt auf dem Gebiet bei anderen Stellen

Die Arbeiten zur Optimierung von Elektrik-/Elektronik-Architekturen sind vielversprechend fortgeführt worden. Hierdurch konnte sich aquintos im Bereich Synthesen positionieren und Kooperationen mit den Firmen INCHRON GmbH http://www.inchron.com und Symtavision GmbH http://www.symtavision.com aufbauen. Dort geht es vor allem um Timinganalysen, um Randbedingungen und zeitliches Verhalten der Fahrzeugkommunikation zu analysieren, bewerten und optimieren.

2.4 Veröffentlichung der Ergebnisse

2.4.1 Publikationen

Während und nach Ende der Projektlaufzeit wurden verschiedene Veröffentlichungen zu den Ergebnissen und Teilergebnissen auf Fachkonferenzen eingereicht und publiziert. Im Einzelnen waren dies folgende Publikationen:

- Daniel Gebauer, Johannes Matheis, Markus Kühl, Klaus D. Müller-Glaser: *Integrierter, graphisch notierter Ansatz zur Bewertung von Elektrik/Elektronik-Architekturen im Fahrzeug*, HDT (Haus der Technik), 2009 [13]
- Nico Adler, Daniel Gebauer, Clemens Reichmann, Klaus D. Müller-Glaser: *Modellbasierte Erfassung von Optimierungsaktivitäten als Grundlage zur Systemoptimierung von Elektrik-/Elektronik-Architekturen*, 14. MBMV-Workshop (Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen 2011), Oldenburg, 2011

2.4.2 Presseartikel

Es wurden unterschiedliche Business Development Unterlagen für den Kundenkreis erstellt. Diese wurden aber nicht der Presse veröffentlicht.

3 Literatur

[1] PREEvision-Produktbeschreibung: http://www.aquintos.info/

[2] AUTOSAR Standard, http://www.autosar.org

[3] European Technology Platform for Embedded Intelligence and Systems (ARTEMIS) Strategic Research Agenda (SRA), 2006, http://www.artemis-sra.eu/downloads/SRA_MARS_2006.pdf

[4] European Technology Platform for Embedded Intelligence and Systems (ARTEMIS) Strategic Research Agenda: Design Methods & Tools, März 2006, http://www.artemis-sra.eu/downloads/RAPPORT_DMT.pdf

[5] EU-Projekt EAST-EEA (Embedded Electronic Architecture) , 2004, www.east-eea.net

[6] EU-Projekt ATESST (Advancing Traffic Efficiency and Safety through Software Technology), www.atesst.org

[7] AADL Standard Info site, http://www.aadl.info/

[8] M. Kühl, C. Reichmann: „Modell-basierte Entwicklung von E/E-Architekturen“, in: Automobil-Elektronik, MI-Verlag, Oktober 2007, http://imperia.mi-verlag.de/imperia/md/content/ai/ae/fachartikel/ael/2007/05/ael07_05_022.pdf

[9] Lars Burgstahler, An enhanced Reference Point Method for Evaluation of Network Performance Aspects, Universität Stuttgart, Institut für Kommunikationsnetze und Rechnersysteme;

[10] Automobilelektronik, Eine Einführung für Ingenieure, 3. Auflage, Konrad Reif, Vieweg/Teubner, 2009, ISBN-13: 978-3-8348-0446-4

[11] Autoelektrik – Autoelektronik, Systeme und Komponenten, 4. Auflage, Bosch, Vieweg Verlag, 2007, ISBN 978-3-528-23872-8

[12] Handbuch Kraftfahrzeugelektronik, Grundlagen, Komponenten, Systeme, Anwendungen, 1. Auflage, Siemens VDO, Wallentowitz / Rief (Hrsg.), Vieweg Verlag, 2006, ISBN-13:978-3-528-03971-4

[13] Daniel Gebauer, Johannes Matheis, Markus Kühl, Klaus D. Müller-Glaser: Integrierter, graphisch notierter Ansatz zur Bewertung von Elektrik/elektronik-Architekturen im Fahrzeug, HDT (Haus der Technik), 2009