Code-Review EDP2 API

ÜBERSICHT

Die Software Experiment Data Persistency & Presentation (EDP2) dient der geordneten Ablage sowie Präsentation von im Rahmen von gezielten Experimenten erhobenen Messwerten. Der Entwurf von EDP2 ist darauf ausgelegt durch eine möglichst große Flexibilität eine hohe Verwendbarkeit in unterschiedlichen Szenarien zu bieten. Hierzu gehören beispielsweise die Unterstützung von Messungen mit einer Anzahl von derzeit bis zu 2^{32} Messwerten sowie mehrdimensionale Messungen.

DATENMODELL

In EDP2 gibt es 3 verschiedene Kategorien von Daten:

- *Repositories* kapseln den Zugriff auf eine EDP2 Persistenzeinheit. Diese kann eine beliebige Menge von Metadaten und Messungen erhalten.
- *Metadaten* beschreiben sowohl den Aufbau und die Aufrufparameter von Experimenten als auch Daten-, Persistenzformate.
- *Messungen* sind eine geordnete Folge der in EDP2 verwalteten Messungen pro Messstrecke

Der Zugriff auf die Daten ist in den einzelnen Kategorien technisch jeweils durch DAOs realisiert. Die einzelnen Kategorien werden im Folgenden näher erläutert. Im SVN sind die Modelle unter svn://i43pc13.ipd.uka.de/code/Palladio.RAS-Models/EDP2.emx zu finden. Hier kann auch die detaillierte Dokumentation zu den einzelnen Modellelementen eingesehen werden.

REPOSITORIES

Repositories sind einzelne Persistenzeinheiten in EDP2. Deren Aufbau und der Zugriff auf die darin liegenden Daten werden im Folgenden näher erläutert. Abbildung 1 enthält die dazu passende Veranschaulichung des dahinterliegenden Modells.

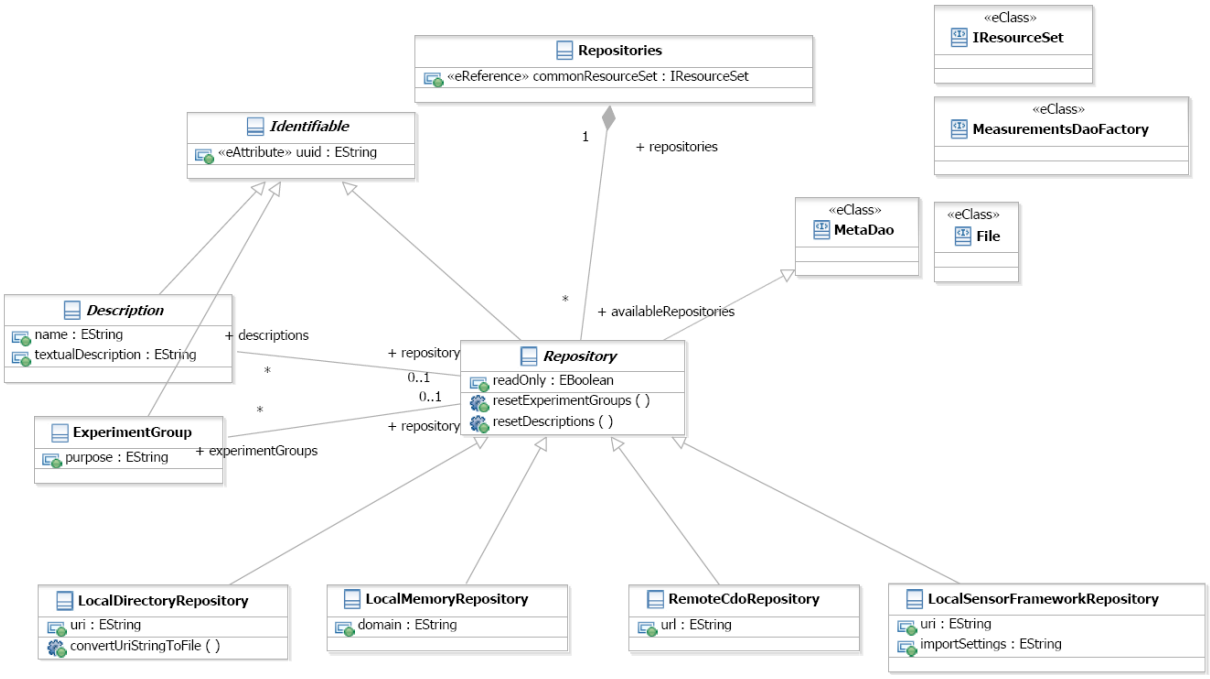

Abbildung 1: Repository-Modell

Ein abstraktes `Repository` (Bildmitte) enthält eine Menge von Experimenten sowie Metrikdefinitionen, die gemeinsam eine Persistenzeinheit bilden. Die Persistierung der dazugehörigen Modellelemente `ExperimentGroup` und `Description` wird je nach Speicher unterschiedlich gehandhabt. Derzeit gibt es 4 Realisierungen(Unten im Bild):

1. `LocalDirectoryRepository` sorgt für die Persistierung innerhalb eines Verzeichnisses auf dem lokalen Dateisystem,
2. `LocalMemoryRepository` sorgt für den Zugriff auf im Arbeitsspeicher befindliche Persistenzeinheiten,
3. `RemoteCdoRepository` für den Zugriff auf entfernt über CDO angesteuerte Persistenzeinheiten und
4. `LocalSensorFrameworkRepository` für den Zugriff auf im lokalen Dateisystem liegende Messungen des SensorFramework.

Zur einfachen Verwaltung mehrerer Repositories und von Querverweisen zwischen den Repositories (bspw. um Metrikdefinitionen wiederzuverwenden) dient das Modellelement `Repositories`.

METADATEN ZU EXPERIMENTEN

Die Ablage von Messdaten in EDP2 erfolgt anhand eines eigenen Metamodells, welches an dieses Dokument angehängt ist. Es besteht aus drei Bereichen: Einer für den Experimentaufbau (`ExperimentsView`), die Definition von Messungen/-strecken (`MeasureDefinitionView`) und für die Metadaten zu Messwerten (`MeasurementsView`). Die Modellelemente der einzelnen Bereiche werden im Folgenden kurz vorgestellt:

EXPERIMENTSVIEW

Bei der gezielten Durchführung von Experimenten steht ein Bewertungsziel gemeinsam für mehrere Experimente fest. Dies kann beispielsweise die Bewertung von 3 Architekturalternativen hinsichtlich der besten Alternative hinsichtlich Performance-Optimierung sein. Alternativen könnten sein A1) Aktuelle Architektur, A2)

Einführung eines Caches, A3) Parallele Verarbeitung von Anfragen. Das gemeinsame Bewertungsziel wird durch das Modellelement `ExperimentGroup` modelliert. Eine für Dritte verständliche Beschreibung des Ziels sollte als `purpose` angegeben werden. Für jede der Alternativen wäre ein eigener Experimentaufbau (`ExperimentSetting`) mit den jeweiligen Architekturen zu definieren. Innerhalb eines Experimentaufbaus werden die einzelnen Messstrecken (`Measure`) referenziert. Dies ermöglicht allen Experimenten gemeinsame Messtrecken, bspw. unveränderte Teile der Architektur, auch als solche zu definieren und diese bei einer späteren Auswertung als identische Strecke in unterschiedlichen Alternativen auswählen zu können. Ein einzelnes Experiment kann nach vollständiger Definition der Messstrecken, also seinem Aufbau, beliebig oft durchgeführt werden. Eine einzelne Durchführung eines Experiments entspricht einem `ExperimentRun`. Die für eine Messstrecke erhobenen Messwerte werden mit Hilfe des Modellelements `Measurement` referenziert.

MEASUREDEFINITIONVIEW, DESCRIPTIONSVIEW

Die Definition einer Messstrecke (`Measure`) enthält Informationen über das beobachtete Objekt, die Metrik (`metric`) mit der die Messwerte erfasst werden und wie Messwerte zu persistieren sind. Ein Objekt kann beispielsweise im Architekturfall eine Software-Komponente oder bei einer Geschwindigkeitsmessung ein Auto sein. Es gibt zwei unterschiedliche Arten von Messstrecken. Die einen beschreiben nominale (`NominalMeasure`), die anderen ordinale Messwerte (`OrdinalMeasure`). Für nominale Messwerte müssen alle möglichen Messwerte zusätzlich als `CategoryIdentifier` modelliert werden. Eine Persistierung von Messwerten (`PersistenceKindOptions`) kann entweder binär (`BinaryPreferred`) oder im XML-Format (`JSXmlPreferred`) stattfinden.

Die Beschreibung von Metriken erfolgt mit dem Modellelement `Metric-Description`. Mehrere Metrikdefinitionen können als Menge von `Descriptions` gespeichert werden. Eine Metrikdefinition kann entweder ein- (`BaseMetricDescription`) oder mehrdimensional (`MetricSetDescription`) sein. Im mehrdimensionalen Fall muss für jede Dimension eine Metrik angegeben werden. Im eindimensionalen Fall muss festgelegt werden in welchem Zahlenformat (`captureType`), mit welcher Skala (`scale`), von welchem Datentyp (`dataType`), mit welchen Eigenschaften bezüglich Monotonie (`monotonic`) und in welcher Standardeinheit (`defaultUnit`) Messwerte erfasst und persistiert werden.

MEASUREMENTSVIEW

Die Referenzierung von Messwerten zu einer Messstrecke erfolgt durch das Element `Measurement`. Die Messung innerhalb eines Experiments kann in verschiedenen Bereichen erfolgen. Beispiele hierfür sind eine Aufwärmphase und eine eingeschwungene Phase. Jede Messstrecke in einer Experimentdurchführung erhält entsprechend `MeasurementRange` Elemente. Hängen die Messphasen mit der der Zeit zusammen kann diese zusätzlich in `startTime` und `endTime` vermerkt werden. In einem einzelnen Bereich können die Messwerte direkt (`RawMeasurements`) und/oder aggregiert (`AggregatedMeasurements`) gespeichert werden. Aggregierte Messungen werden derzeit nicht vollständig unterstützt und hier deswegen nicht beschrieben. Direkt zur Persisistierung durchgereichte Messwerte werden pro Dimension der in der Messstrecke angegebenen Metrik mit Hilfe einer `DataSeries` referenziert. Für jede `DataSeries` kann es, je nach Typ der Dimension, nominale (`NominalStatistics`), ordinale (`OrdinalStatistics`), interval (`IntervalStatistics`) oder ratio-Statistiken (`RatioStatistics`) geben. Je nach festgelegter Persistierung werden die Messwerte im Format für nominale (`NominalMeasurements`), binäre (`DoubleBinaryMeasurments`, `LongBinaryMeasurements`) oder XML (`JSXmlMeasurements`) Messwerte gespeichert. Bei binären Messwerten ist zusätzlich die Einheit anzugeben in der Persistiert wird, da diese unter Umständen erst verlustbehaftet umgerechnet werden muss. Der Zugriff und die Persistierung von Messwerten selbst erfolgt über ein DAO, welches über den Identifier `valuesUuid` angefordert werden kann.

MESSUNGEN

Nominale Messungen werden über das Modellelement ObservedNominalMeasurements (unten im Anhang MeasurementsView) als normales EMF-Modell verwaltet und gespeichert. Jeder Messwert (`ObservedCategory`) verweist dabei auf einen der vordefinierten möglichen Messwerte (`CategoryIdentifier`).

Binäre Messungen werden entsprechend ihrer binären Byte-Repräsentation in Java nach IEEE Standard seriell verwaltet und gespeichert. Zum effizienten Zugriff eines solchen Arrays auf einem Datenträger unterstützt die entsprechende Implementierung Kacheln.

SYSTEMEIGENSCHAFTEN

Aggregierte Messungen werden derzeit nicht unterstützt.

Die Implementierung von Repositories ist derzeit nur für `LocalDirectoryRepository vollständig.`