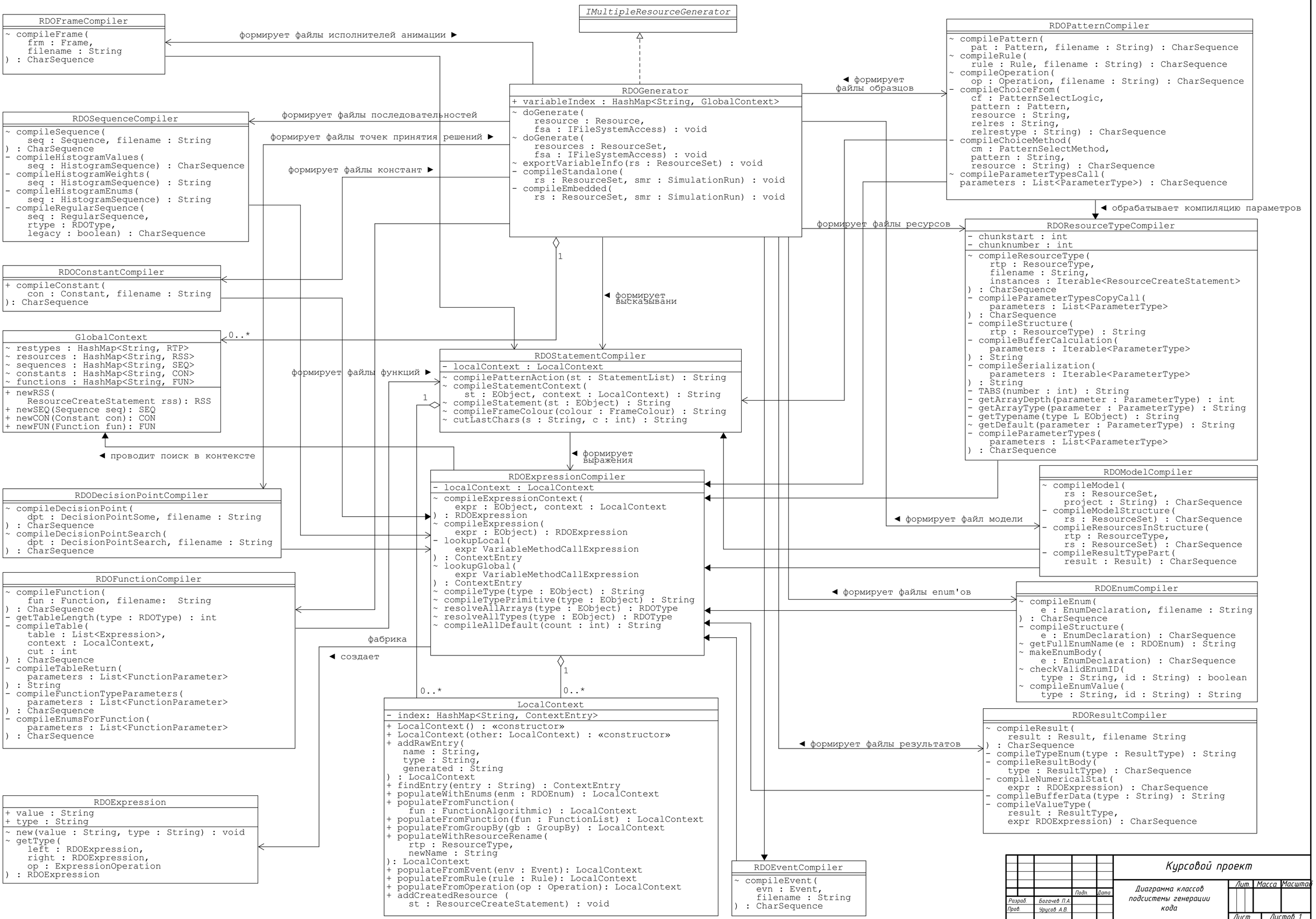
IMultipleResourceGenerator
RDOFrameCompiler
~ compileFrame(
frm : Frame,
filename : String
) : CharSequence
формирует файлы исполнителей анимации ▶
RDOGenerator
+ variableIndex : HashMap<String, GlobalContext>
~ doGenerate(
resource : Resource,
fsa : IFileSystemAccess) : void
~ doGenerate(
resources : ResourceSet,
fsa : IFileSystemAccess) : void
~ exportVariableInfo(rs : ResourceSet) : void
- compileStandalone(
rs : ResourceSet, smr : SimulationRun) : void
- compileEmbedded(
rs : ResourceSet, smr : SimulationRun) : void
◀ формирует файлы образцов
RDOPatternCompiler
~ compilePattern(
pat : Pattern, filename : String) : CharSequence
~ compileRule(
rule : Rule, filename : String) : CharSequence
~ compileOperation(
op : Operation, filename : String) : CharSequence
- compileChoiceFrom(
cf : PatternSelectLogic,
pattern : Pattern,
resource : String,
relres : String,
relrestype : String) : CharSequence
- compileChoiceMethod(
cm : PatternSelectMethod,
pattern : String,
resource : String) : CharSequence
~ compileParameterTypesCall(
parameters : List<ParameterType>) : CharSequence
RDOSequenceCompiler
~ compileSequence(
seq : Sequence, filename : String
) : CharSequence
- compileHistogramValues(
seq : HistogramSequence) : CharSequence
- compileHistogramWeights(
seq : HistogramSequence) : String
- compileHistogramEnums(
seq : HistogramSequence) : String
- compileRegularSequence(
seq : RegularSequence,
rtype : RDOType,
legacy : boolean) : CharSequence
формирует файлы последовательностей
формирует файлы точек принятия решений ▶
формирует файлы констант ▶
◀ обрабатывает компиляцию параметров
формирует файлы ресурсов
RDOResourceTypeCompiler
- chunkstart : int
- chunknumber : int
~ compileResourceType(
rtp : ResourceType,
filename : String,
instances : Iterable<ResourceCreateStatement>
) : CharSequence
- compileParameterTypesCopyCall(
parameters : List<ParameterType>
) : CharSequence
- compileStructure(
rtp : ResourceType) : String
- compileBufferCalculation(
parameters : Iterable<ParameterType>
) : String
- compileSerialization(
parameters : Iterable<ParameterType>
) : String
- TABS(number : int) : String
- getArrayDepth(parameter : ParameterType) : int
- getArrayType(parameter : ParameterType) : String
- getTypename(type L EObject) : String
~ getDefault(parameter : ParameterType) : String
- compileParameterTypes(
parameters : List<ParameterType>
) : CharSequence
RDOConstantCompiler
+ compileConstant(
con : Constant, filename : String
): CharSequence
1
◀ формирует высказывани
0..*
GlobalContext
~ restypes : HashMap<String, RTP>
~ resources : HashMap<String, RSS>
~ sequences : HashMap<String, SEQ>
~ constants : HashMap<String, CON>
~ functions : HashMap<String, FUN>
+ newRSS(
ResourceCreateStatement rss): RSS
+ newSEQ(Sequence seq): SEQ
+ newCON(Constant con): CON
+ newFUN(Function fun): FUN
формирует файлы функций ▶
1
RDOStatementCompiler
- localContext : LocalContext
~ compilePatternAction(st : StatementList) : String
~ compileStatementContext(
st : EObject, context : LocalContext) : String
~ compileStatement(st : EObject) : String
~ compileFrameColour(colour : FrameColour) : String
~ cutLastChars(s : String, c : int) : String
◀ проводит поиск в контексте
◀ формирует выражения
RDOModelCompiler
~ compileModel(
rs : ResourceSet,
project : String) : CharSequence
- compileModelStructure(
rs : ResourceSet) : CharSequence
- compileResourcesInStructure(
rtp : ResourceType,
rs : ResourceSet) : CharSequence
- compileResultTypePart(
result : Result) : CharSequence
◀ формирует файл модели
RDODecisionPointCompiler
~ compileDecisionPoint(
dpt : DecisionPointSome, filename : String
) : CharSequence
~ compileDecisionPointSearch(
dpt : DecisionPointSearch, filename : String
) : CharSequence
RDOExpressionCompiler
- localContext : LocalContext
~ compileExpressionContext(
expr : EObject, context : LocalContext
) : RDOExpression
~ compileExpression(
expr : EObject) : RDOExpression
- lookupLocal(
expr VariableMethodCallExpression
) : ContextEntry
~ lookupGlobal(
expr VariableMethodCallExpression
) : ContextEntry
~ compileType(type : EObject) : String
~ compileTypePrimitive(type : EObject) : String
~ resolveAllArrays(type : EObject) : RDOType
~ resolveAllTypes(type : EObject) : RDOType
~ compileAllDefault(count : int) : String
RDOFunctionCompiler
~ compileFunction(
fun : Function, filename: String
) : CharSequence
- getTableLength(type : RDOType) : int
- compileTable(
table : List<Expression>,
context : LocalContext,
cut : int
) : CharSequence
- compileTableReturn(
parameters : List<FunctionParameter>
) : String
- compileFunctionTypeParameters(
parameters : List<FunctionParameter>
) : CharSequence
- compileEnumsForFunction(
parameters : List<FunctionParameter>
) : CharSequence
фабрика
◀ создает
◀ формирует файлы enum'ов
RDOEnumCompiler
~ compileEnum(
e : EnumDeclaration, filename : String
) : CharSequence
- compileStructure(
e : EnumDeclaration) : CharSequence
~ getFullEnumName(e : RDOEnum) : String
~ makeEnumBody(
e : EnumDeclaration) : CharSequence
~ checkValidEnumID(
type : String, id : String) : boolean
~ compileEnumValue(
type : String, id : String) : String
1
0..*
0..*
LocalContext
- index: HashMap<String, ContextEntry>
+ LocalContext() : «constructor»
+ LocalContext(other: LocalContext) : «constructor»
+ addRawEntry(
name : String,
type : String,
generated : String
) : LocalContext
+ findEntry(entry : String) : ContextEntry
+ populateWithEnums(enm : RDOEnum) : LocalContext
+ populateFromFunction(
fun : FunctionAlgorithmic) : LocalContext
+ populateFromFunction(fun : FunctionList) : LocalContext
+ populateFromGroupBy(gb : GroupBy) : LocalContext
+ populateWithResourceRename(
rtp : ResourceType,
newName : String
): LocalContext
+ populateFromEvent(env : Event): LocalContext
+ populateFromRule(rule : Rule): LocalContext
+ populateFromOperation(op : Operation): LocalContext
+ addCreatedResource (
st : ResourceCreateStatement) : void
◀ формирует файлы результатов
RDOResultCompiler
~ compileResult(
result : Result, filename String
) : CharSequence
- compileTypeEnum(type : ResultType) : String
- compileResultBody(
type : ResultType) : CharSequence
- compileNumericalStat(
expr : RDOExpression) : CharSequence
- compileBufferData(type : String) : String
- compileValueType(
result : ResultType,
expr RDOExpression) : CharSequence
RDOExpression
+ value : String
+ type : String
~ new(value : String, type : String) : void
~ getType(
left : RDOExpression,
right : RDOExpression,
op : ExpressionOperation
) : RDOExpression
RDOEventCompiler
~ compileEvent(
evn : Event,
filename : String
) : CharSequence
Курсовой проект
Диаграмма классов подсистемы генерации кода
Лит. Масса Масштаб
Подп. Дата
Разраб. Богачев П.А.
Пров. Урусов А.В.
Лист Листов 1
МГТУ им. Н.Э.Баумана
Кафедра РК9
группа РК9-102