LangDev Meeting 2018, Amsterdam

PORTING THE WHOLE LWB TO SWIFT/IOS SWIFT GENERATORS

Riccardo Solmi, Whole Factory, Italy

THE WHOLE PLATFORM IS AT THE STATE OF THE ART

- DLSs covering almost every aspect of language definition
- Visual notations designed for gesture based interactions
- DSLs for evolution, testing and deployment
 - multiple versions, instance migration, software product lines

... BUT WE ARE UNDERGOING A SLOW EVOLUTION

- The Whole Platform is not where we want it to be
- Implementation is far behind our vision
- New features are struggling to become pervasive

TOO MUCH INNOVATION BRAKES: UNDERLYING TECHNOLOGIES

- Java is no longer Write once and Run Everywhere
 - (iOS, Windows 10 UWP)
- GEF 3 is no longer developed and is outdated
 - (Draw 2D, poor Multitouch, non composable UI, Java)
- Eclipse is too much big and complex for too little
 - (Classic IDE, Java)

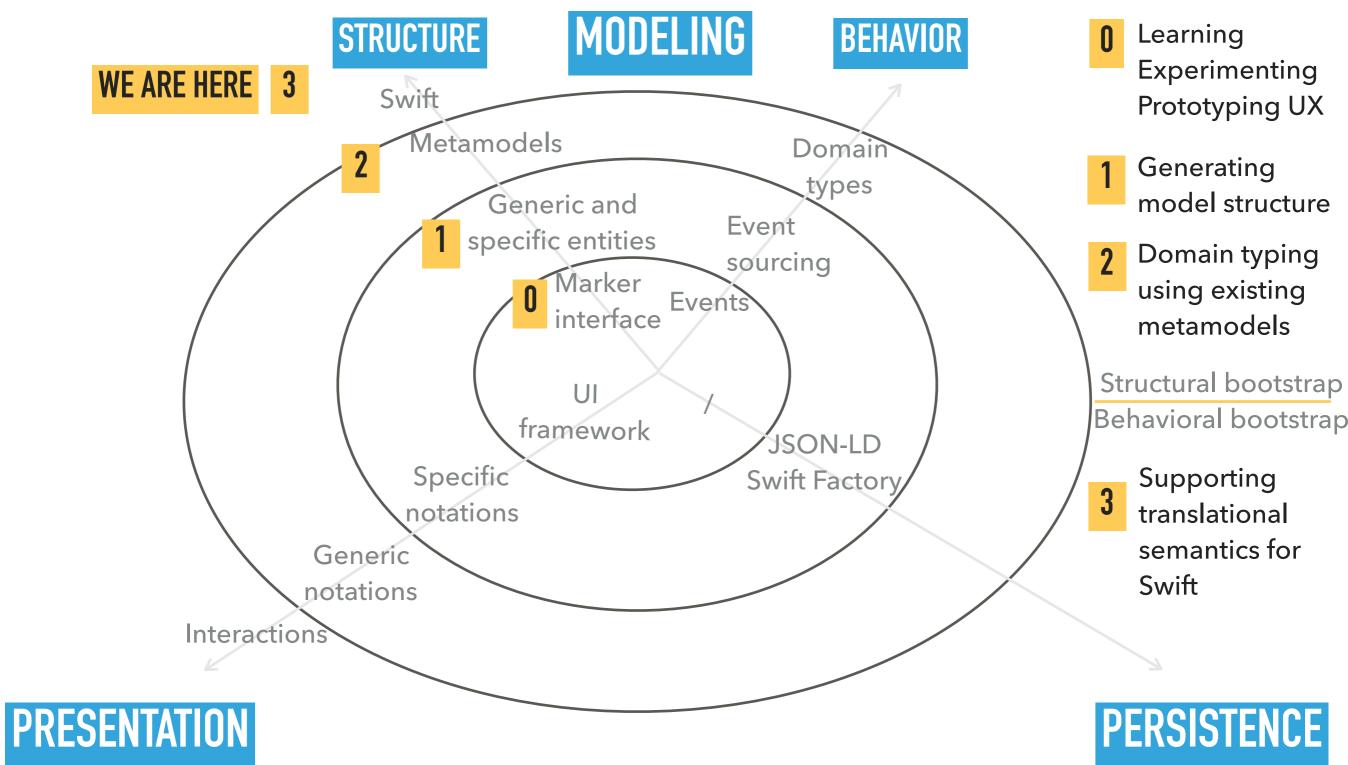
TOO MUCH INNOVATION BRAKES: WHOLE FRAMEWORK

- Duality: framework level <-> domain level
 - Duplication, encoding, constrained domain innovation
- Framework APIs in well-established but wrong places
 - Modeling, events, reflection, and typing
- Framework evolution is inherently slower than domain level evolution

SO WE NEED TO START OVER AGAIN?

- No, domain level code and generated code account for 98% of the code base
- We decided to reuse the domain level and to redesign and rewrite the framework level
- Swift + iOS are the technologies chosen for the porting
- Existing Eclipse based workbench has been used to accelerate the bootstrapping process

OVERVIEW OF THE PORTING PROCESS: FIRST ITERATIONS

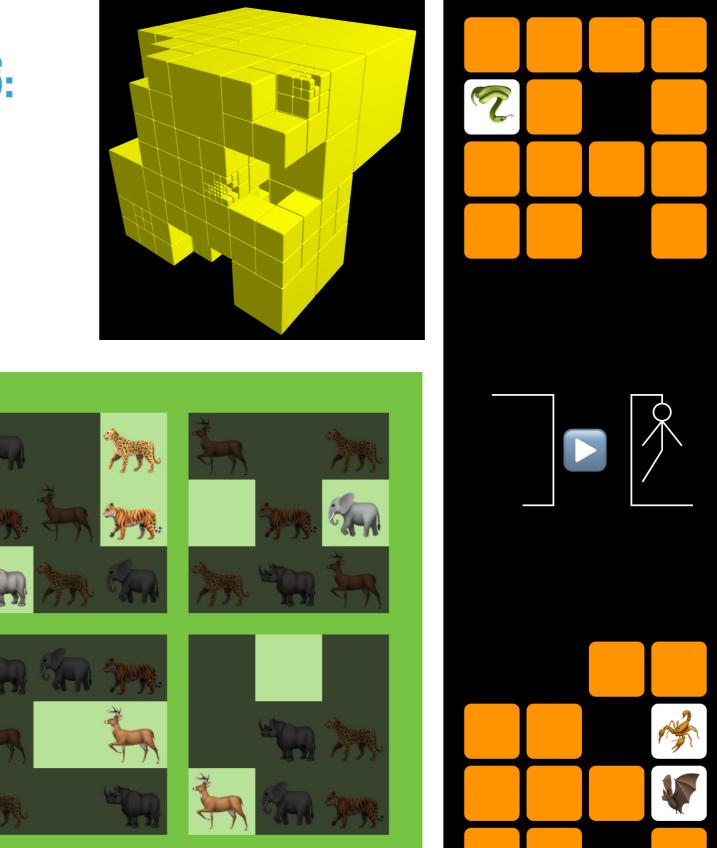


ITERATION 0

FIRST ITERATION PROTOTYPES: SIMPLE GAMES

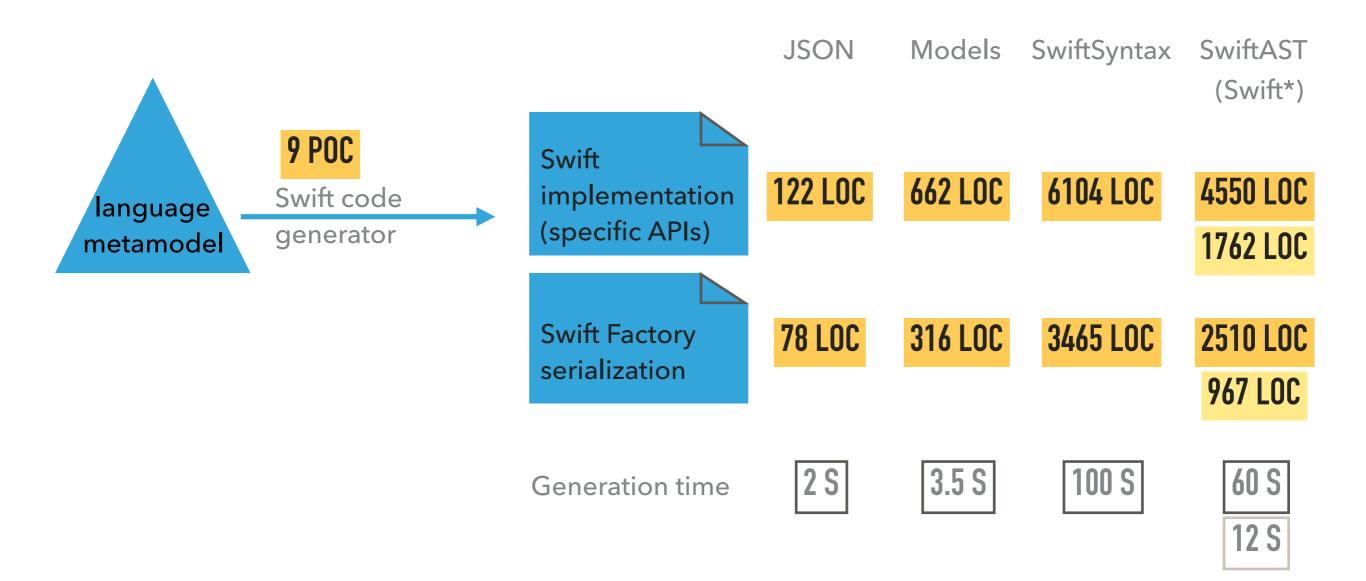


1





GENERATING LANGUAGE IMPLEMENTATION (STRUCTURE)



* Metamodel hand written starting from the official grammar

ITERATION 1

MODELS ON IOS

specific notation

U	RI whole:	whole:org.whole.lang.json:JSONModel							
Namespa	ce <mark>org.w</mark> h	org.whole.lang.json							
Model Nar	ne JSOI	JSON							
Versi	on Resolv	ver							
Foreign Typ	es Supert	rtypesOf							
		types Valu	ie						
	foreigr	Type Any	Туре						
			71						
Supertypes	Entity	Structure							
	Value	Modifiers	Feature	Туре					
Value	Object	_							
	Pair	Modifiers	Feature	Туре					
		L.	name	Name					
			value	Value					
L.	Name	String							
Value	Array								
Value	String	String							
Value	Decimal	double							
Value	Int	long							
Value	Bool	boolean							
Value	Null	Modifiers	ers Feature Type						

Namespace org.whole.lang.sv Model Name Swift					
Version Resolver					
Foreign Types					
Suportupos	Entity	Structure			
Supertypes	Entity	Structure			
	CodeBlock	L.	Es structure - T		
	Statement		Feature T		
Statement	Declaration		Feature T	уре	
Declaration	ProtocolDeclaration	Modifiers			Туре
		ш. 	name		Name
		optional	inheritedPr		Protocols
		-	declaration	าร	ProtocolMemberDeclarations
	ProtocolMemberDeclarations	L.	-		
-	ProtocolMemberDeclaration		Feature T		
ProtocolMemberDeclaration	ProtocolPropertyDeclaration	Modifiers		Туре	
		-	name	Name	
		optional	type		notation
		optional	hasSetter	Boolear	
ProtocolMemberDeclaration	ProtocolMethodDeclaration	Modifiers	Feature		Туре
		.	name		Name
		optional	genericPar	ameters	GenericParameters
		ш."	signature		FunctionSignature
Declaration	MemberDeclaration	Modifiers	Feature T	уре	
ClassMemberDeclaration					
EnumerationMemberDeclaration					
MemberDeclaration	EnumerationDeclaration	Modifiers	Feature		Туре
			name		Name
		optional	genericPar	ameters	GenericParameters
		optional	adoptedPro	otocols	Protocols
		_	declaration	าร	EnumerationMemberDeclaration
	EnumerationMemberDeclarations	_			
	EnumerationMemberDeclaration	Modifiers	Feature T	уре	
MemberDeclaration	StructureDeclaration	Modifiers	Feature		Туре
			name		Name
		optional	genericPar	ameters	GenericParameters
		optional	adoptedPro	otocols	Protocols
		_	declaration	าร	StructureMemberDeclarations
	StructureMemberDeclarations				
MemberDeclaration	ClassDeclaration	Modifiers	Feature		Туре
			name		Name
		optional	genericPar	ameters	GenericParameters
		optional	superclass		Typeldentifier
		optional	adoptedPro	otocols	Protocols
		_	declaration	าร	ClassMemberDeclarations
	ClassMemberDeclarations	_			
_	ClassMemberDeclaration	Modifiers	Feature T	уре	
MemberDeclaration	FunctionDeclaration	Modifiers	Feature		Туре
		optional	modifiers		FunctionModifiers
			name		Name
		optional	genericPar	ameters	GenericParameters
			signature		FunctionSignature
		_	body		CodeBlock
	FunctionSignature	Modifiers	Feature	Tv	ре
	-	optional	parameters		rameters
		optional	throwsMod		nowsModifier
		optional	returnType		pe
	ThrowsModifier		, clainippe	· y	F -

URI whole:org.whole.lang.swift:SwiftModel

Namespace org.whole.lang.swift

PERSISTENCE

- To bootstrap the platform by loading the metamodels
- > XML Builder (too much complex for the bootstrap)
- JSON-LD (unordered -> cannot bootstrap metamodels)
- Swift Factory

ITERATION 1

SWIFT FACTORY PERSISTENCE

No parser needed

 Suitable for embedding domain templates in Swift code

1 import Foundation 2 3 public func createSwiftASTModel(context: MoldingContext) -> Models_Model { let modelsEF = ModelsFactory(context: context) 4 5 let commonsEF = CommonsFactory(context: context) return modelsEF.Model(6 name: modelsEF.SimpleName("SwiftAST"), 7 8 typeRelations: commonsEF.Resolver(), 9 declarations: modelsEF.ModelDeclarations(10 modelsEF.SimpleEntity(11 modifiers: modelsEF.EntityModifiers(modelsEF.EntityModifier(.abstract)), 12 name: modelsEF.SimpleName("Syntax"), 13 types: modelsEF.Types(), features: modelsEF.Features()), 14 modelsEF.SimpleEntity(15 modifiers: modelsEF.EntityModifiers(modelsEF.EntityModifier(.abstract)), 16 17 name: modelsEF.SimpleName("Decl"), 18 types: modelsEF.Types(modelsEF.SimpleName("Syntax")), 19 features: modelsEF.Features()), 20 modelsEF.SimpleEntity(21 modifiers: modelsEF.EntityModifiers(modelsEF.EntityModifier(.abstract)), 22 name: modelsEF.SimpleName("Expr"), 23 types: modelsEF.Types(modelsEF.SimpleName("Syntax")), 24 features: modelsEF.Features()), modelsEF.SimpleEntity(25 modifiers: modelsEF.EntityModifiers(modelsEF.EntityModifier(.abstract)), 26 27 name: modelsEF.SimpleName("Pattern"), 28 types: modelsEF.Types(modelsEF.SimpleName("Syntax")), 29 features: modelsEF.Features()), modelsEF.SimpleEntity(30 modifiers: modelsEF.EntityModifiers(modelsEF.EntityModifier(.abstract)), 31 32 name: modelsEF.SimpleName("Stmt"), 33 types: modelsEF.Types(modelsEF.SimpleName("Syntax")), features: modelsEF.Features()), 34 modelsEF.SimpleEntity(35 modifiers: modelsEF.EntityModifiers(modelsEF.EntityModifier(.abstract)), 36 37 name: modelsEF.SimpleName("Type"), types: modelsEF.Types(modelsEF.SimpleName("Syntax")), 38 features: modelsEF.Features()), 39 modelsEF.SimpleEntity(40 modifiers: modelsEF.EntityModifiers(), 41 42 name: modelsEF.SimpleName("Token"), types: modelsEF.Types(modelsEF.SimpleName("Syntax")), 43 features: modelsEF.Features(44 modelsEF.Feature(45 modifiers: modelsEF.FeatureModifiers(), 46 47 type: modelsEF.SimpleName("TokenKind"), 48 name: modelsEF.SimpleName("kind"), 49 oppositeName: commonsEF.Resolver()), modelsEF.Feature(50 modifiers: modelsEF.FeatureModifiers(modelsEF.FeatureModifier(.option 51 52 type: modelsFF SimpleName("Text")



GENERIC NOTATIONS

- "Table" variant
- Dark theme

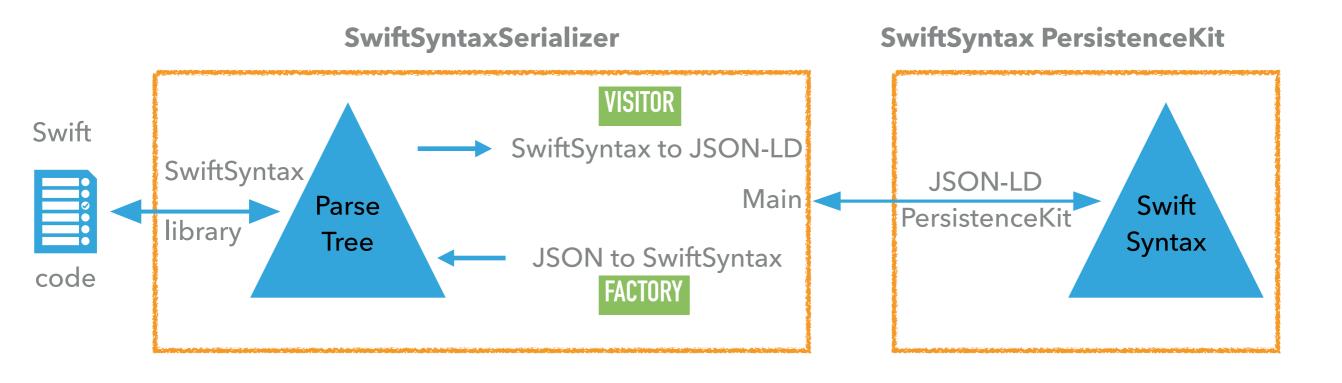
lodel						
name	JSON					
declarations	SimpleEnt	ity				
	features	_				
	name	Value				
	types	_				
	modifiers	abstract				
	Composite	eEntity				
		modifiers				
		name	Obje	ect		
		types	Valu	le		
		ntModifiers	orde			
	comp	onentType	Pair			
	SimpleEnt	ity				
	features	Feature				
		na	ame	name		
		oppositeNa	ame	Resolver		
			type	Name		
		modi	fiers			
		Feature				
		na	ame	value		
		oppositeNa	ame	Resolver		
		1	type	Value		
		modi	fiers			
	name	Pair				
	types					
	modifiers	_				
	DataEntity	,				
	types	_				
	name	Name				
	dataType	String				
	modifiers	_				
	Composite	Entity				
		modifiers	L			
		name	Arra	у		
		types	Valu	Ie		
	componer	ntModifiers	orde	ered		

SWIFT INTEGRATION THROUGH REVERSE ENGINEERING

- Add concrete syntax level Swift persistence to Eclipse backed by official Swift parser
 - Using a fully automated generative process
 - Suitable for source editing scenarios
- Add AST level Swift persistence to Eclipse
 - Chained to, and initially derived by the syntax level
 - Refined to better support generative scenarios

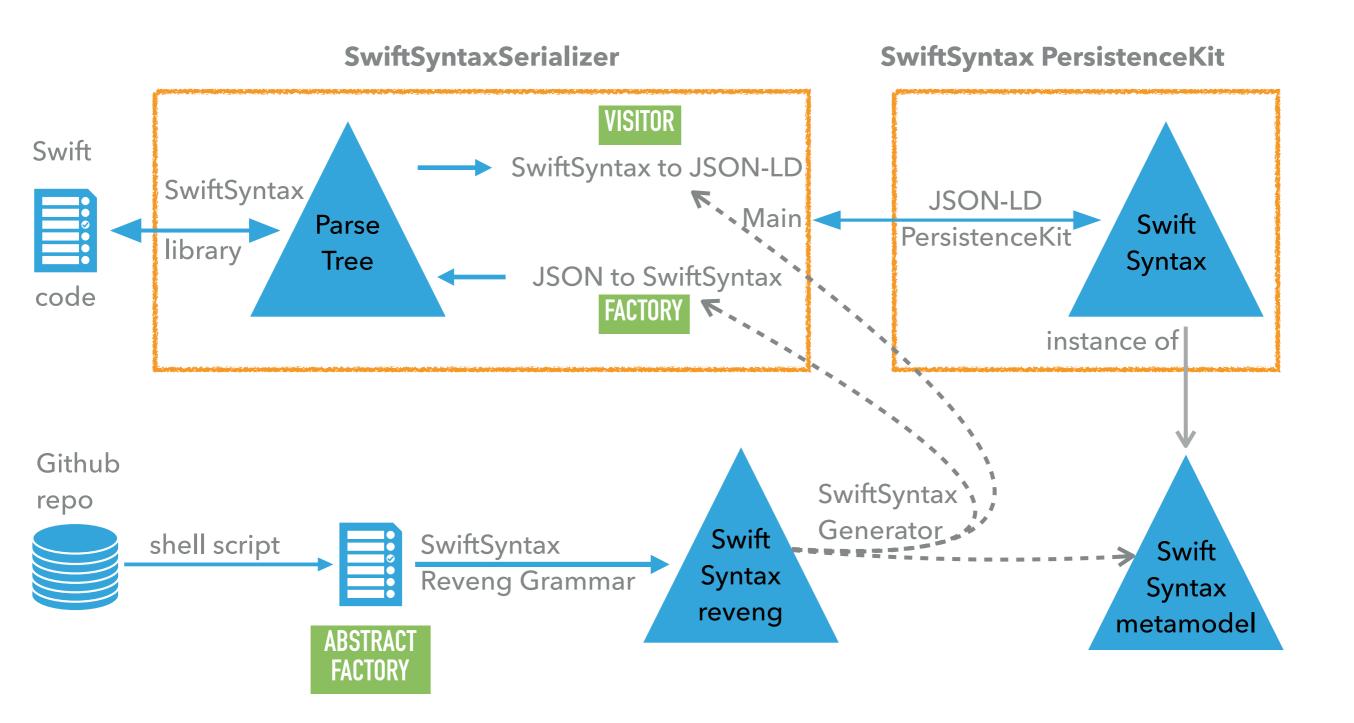


SWIFTSYNTAX PERSISTENCE



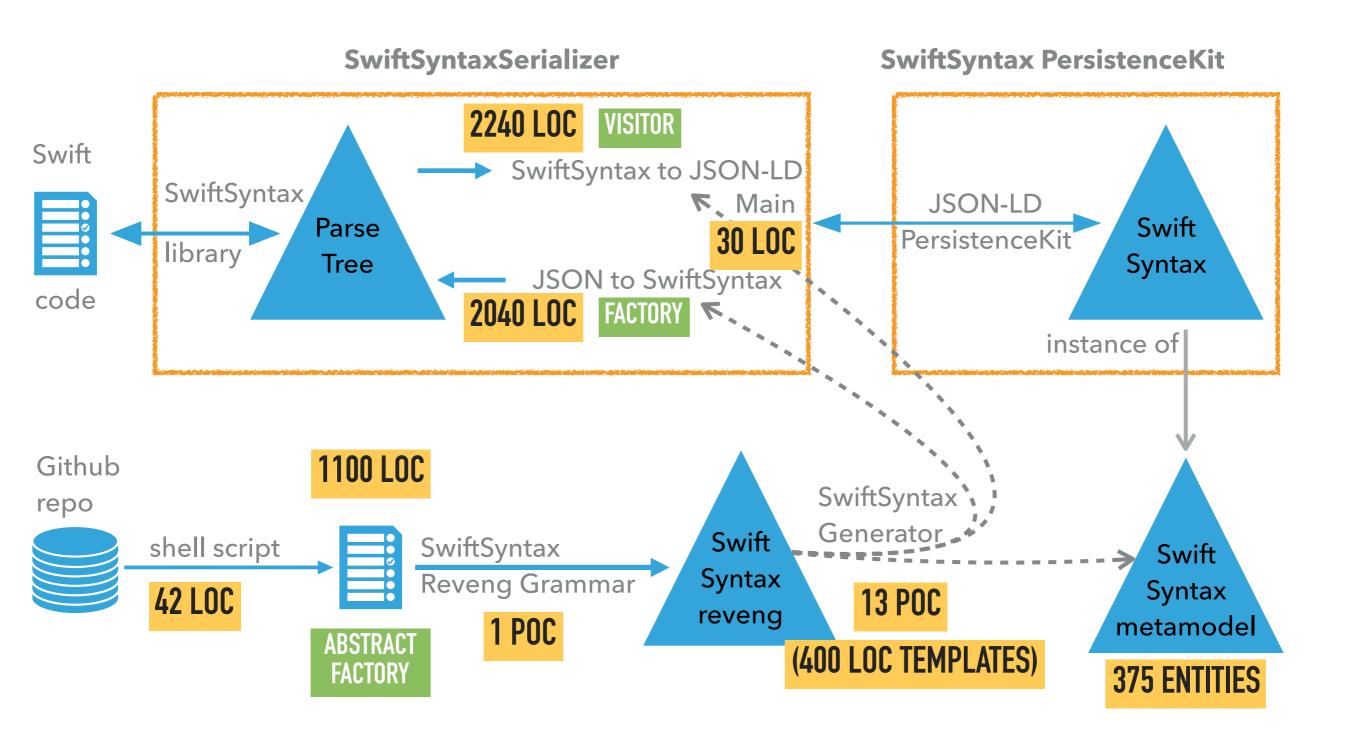


SWIFTSYNTAX PERSISTENCE





SWIFTSYNTAX PERSISTENCE





SWIFT SYNTAX EXAMPLE

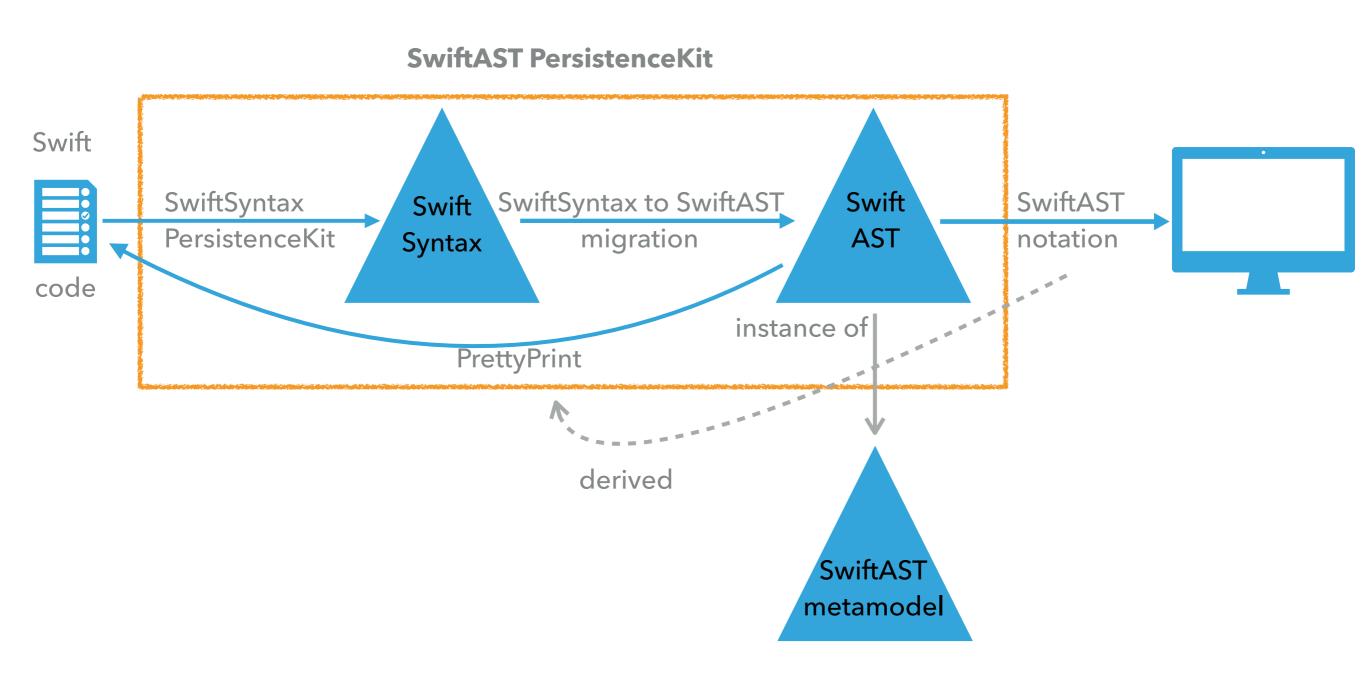
var optionalString: String? = "Hello"
print(optionalString == nil)

var optionalName: String? = "John Appleseed"
var greeting = "Hello!"
if let name = optionalName {
 greeting = "Hello, \(name)"
}

ırceFile 🔻												
ements 🛛 🖻 CodeBl	BlockItemList											
item												se
	ableDecl 🔻											
	attributes modifiers											
letOr\	rVarKeyword	🖃 Token 🕨										
		kino trailingTrivia	d varKeyword a tt var									
	bindings	E PatternBind										
		pattern	0	typeAnno	tation		initi	initializer			trailingComma	
		🗆 IdentifierPa	attern 🔻	⊡ TypeAr	nnotation 🔻			🖃 InitializerClause 🔻				
		identifier	🖃 Token 🕨	colon	🖃 Token 🕨		e	equal	🖃 Token 🕨			
			kind identifier text optionalString	kind c trailingTrivia text :				kind equal trailingTrivia text =				
				type				value	□ StringLiteralExpr ▼			
						🖃 SimpleTypeldentifier 🍸			stringLiteral 🖃 Token 🕨			
						name Token kind ide text Str	entifier		kind stringLitera text "Hello"	I		
					questionMark							
						kind postfixQuestionMark trailingTrivia text ?						
+ Funct	tionCallExpr 🕨				I I							
	ableDecl 🔻											
fToken 🛨 Token												



SWIFTAST PERSISTENCE





SWIFT AST EXAMPLE

var optionalString: String? = "Hello"
print(optionalString == nil)

var optionalName: String? = "John Appleseed"
var greeting = "Hello!"
if let name = optionalName {
 greeting = "Hello, \(name)"
}

var optionalString : String ? = "Hello"

BinaryOperatorExpr { Token { Token { kind : spacedBinaryOperator nil kind : spacedBinaryOperator nil kind : spacedBinaryOperator nil text : == } var optionalName : String "? = "John Appleseed" " var greeting : " = "Hello!" " if " let name : " = optionalName " { Hello, greeting " AssignmentExpr \(name ")



					let commonsEF = CommonsFactory(context: context)						
				return modelsEF.Model(
					name: modelsEF.SimpleName("Models"),						
CWIET V	ςτ εγλΜDΙ	E (9)		typeRelations: modelsEF.TypeRelations(),							
JWIFIA	ST EXAMPI	LL (<i>L)</i>		declarations: modelsEF.ModelDeclarations(
				modelsEF.SimpleEntity(
import 🛛 🖓 Foundation					modifiers: modelsEF.EntityModifiers(),						
public func createModelsModel	□ (□ context □ : MoldingContext □ □ = □) □ -> N	Models_Model 😐 🛛 {			name: modelsEF.SimpleName("Model"),						
I let modelsEF : I = Mode	elsFactory = (context : context =) = =			types: modelsEF.Types(),							
	mmonsFactory = (context : context =) = =				features: modelsEF.Features(
ſ		`			modelsEF.Feature(
	name : modelsEF	,									
	typeRelations : modelsEF \square . TypeRelations \square (\square)	((
			modifiers : modelsEF 🛛 . EntityMod								
			name : modelsEF u . SimpleName types : modelsEF u . Types u (u) u	()							
			types - modelsEP Types - (-) -	ſ							
					modifiers : modelsEF = . FeatureModifiers = (=) = type : modelsEF = . SimpleName = (= : "SimpleName") =						
				- : modelsEF Feature -	name : modelsEF = . SimpleName = [= : "name"] =						
					oppositeName : commonsEF = . Resolver = (=) =						
					modifiers : modelsEF = . FeatureModifiers = = : modelsEF = . FeatureModifier = = = . optional = = = = = = = = = = = = = = = = = = =						
				□ : modelsEF □ . Feature □	type : modelsEF						
					name : modelsEF = . SimpleName = (= : "typeRelations") =						
					coppositeName : commonsEF □ . Resolver □ (□) □						
					modifiers : modelsEF = . FeatureModifiers = (=) =						
				- : modelsEF Feature -	type : modelsEF						
		□ : modelsEF □ . SimpleEntity □			oppositeName : commonsEF = . Resolver = (=) =						
			features : modelsEF 🛛 . Features 🗆		modifiers : modelsEF = . FeatureModifiers = (=)						
				□ : modelsEF □ . Feature □	type : modelsEF = . SimpleName = (= : "Namespace") =						
					name : modelsEF = . SimpleName = [= : "namespace"] =						
					oppositeName : commonsEF = . Resolver = (=) =						
					modifiers : modelsEF . FeatureModifiers . FeatureModifier . FeatureModifier . optional						
			□ : modelsEF □ . Feature □	type : modelsEF							
					name : modelsEF = . SimpleName = (= : "version") = oppositeName : commonsEF = . Resolver = (=) =						
					modifiers : modelsEF = . FeatureModifiers = (=) = type : modelsEF = . SimpleName = (= : "URI") =						
				: modelsEF . Feature	name : modelsEF = . SimpleName = (= : "uri") =						
	declarations - modelsEE R - MadelDadarstings - R				oppositeName : commonsEF = . Resolver = (=) =						
return modelsEF 🛛 . Model 🔍	declarations : modelsEF \square . ModelDeclarations \square		l	l							
			modifiers : modelsEE = Entitul	Modifiers = (=) =							

import Foundation

public func createModelsModel(context: MoldingContext) -> Models_Model {

let modelsEF = ModelsFactory(context: context)