Implementing complex e-Government solutions with open source and BPM

Export Control System phase 2 (ECS2)

Sirma Solutions & Sirma ITT

Vladimir Alexiev
PM & Analyst

Adrian Mitev
Web TL

Alexander Bukev
Web Dev

Java2Days Conference, 8.10.2009

Outline

- Who We Are
- What is ECS2
- Tools and Technologies
- Data Model
- Model-Driven Generation
- System Architecture
- Business Process Modeling and BPMS
- XPath-based DSL (Assignments, Rules and Conditions)
- GUI Generation and Features
- Deployment Model

Who are We

Who Are We (Company)

- Sirma Group Holding is one of the largest private Bulgarian software development and IT consulting holdings
  - 17 years of history, 400 staff, 9 daughter companies, 5 JVs. Offices in 5 Bulgarian cities and 3 overseas. See more at www.sirma.bg (BG), www.sirma.com (EN)
- Sirma Solutions Corp is the backbone of Sirma
  - 150 staff, government and private projects, outsourced development, *incubation* of new ideas and units. Inherited Sirma's history and goodwill at corporate reorganization (when holding was setup)
- Sirma ITT specializes in complex e-Government projects
  - 5 years of history, 40 staff (Ruse 30, Sofia 8, Varna 2), young (average age 23), one of the few BG companies who *raises its own staff*.
  - Focus on JEE, SOA, BPM, RUP, UML.
- Sirma bought ITT in late 2007 to add its experience and management practices
  - Several joint projects in the customs/excise area
  - Teams shared between Solutions and ITT and fully integrated

Who Are We (Authors)

- Vladimir Alexiev is a Sirma Group founder and CTO of Sirma ITT / Sirma Solutions. He has MS and PhD degrees in computer science, PMP certification and 18 years of IT experience. Having returned home after 12 years in Canada, his dream is to see effective IT use, efficient eGovernment and less corruption in Bulgaria. He is currently trying to start up a Sirma IT Consulting unit.
- Adrian Mitev is Web team lead at Sirma ITT. He has 8 years of experience with Object-Oriented languages, the last 4 specializing in Java front-end development. He contributes to open source projects JBoss Richfaces, Apache MyFaces, Apache Shale and JBoss Tools. Current interests include new technologies in the JEE6 stack JSF2 (JSR-314), CDI (JSR-299), Bean Validation (JSR-303) and JPA2 (JSR-317).
- Alexander Bukev is a Web developer at Sirma ITT. From early childhood his computer experience went through BASIC, C, ASM and embedded systems, then more recently, object-oriented programming and innovative web-based GUI using JSF and Seam. Special interests include innovations in user experience, usability, intuitive interactive GUIs, code optimization and redundant/failsafe systems.
- *Vladimir and Adrian are lecturing in the MS Software Engineering programs sponsored by Sirma at University of Ruse and Varna Free University.*
- *Alexander and 6 other Sirma ITT staff are graduating in the MS SwEng program*

What is ECS2

What is ECS2?

- Trans-European system involving 31 countries
  - Coordinated by DG Taxation and Customs Union (TAXUD)
  - Client: Bulgarian Customs Administration (BCA)
- Strict timeline. "Only" 6 countries were late. BG ECS2 was deployed on time on 1 July 2009
  - 11 months from start to production (some national functionalities finished 3m later)
  - Including Conformance Testing with EC (and fixing bugs in the CT!), deployment and training with BCA
  - Used intensively: currently handles 80-100,000 export movements (500,000 messages) per month.
  - Users: 3500 customs officers, 200,000 potential traders
  - Bulgarian Industrial Association is teaching traders how to use Electronic Declaration (paid trainings)
- Large project: 45 message types averaging 100 fields each, 50 Elementary Business Processes (EBP), 30 man/years effort. Subsystems:
  - ECS2 Core : message processing (in/outflow), business process, persistence
  - ECS2 Web: customs officers data view/entry
  - DTI B2B: Direct Trader Interface for system-to-system electronic declaration processing
  - DTI Web: trader portal for interactive entry of electronic declarations (manual or XML from file)
  - Messaging: EU (CCN/CSI communication, Redix), DTI (db/webservices), BICIS (webservices)
  - Routing between ECS1 and ECS2 (complex transition period!)
  - Processing Reference Data, Customs Office List, Unavailability (RD/COL/UNA)
  - External system integration (BICIS, CAS, RA, SDM, EORI/AEO, TARIC, other gov agencies)
  - Printing (Single Administrative Document), Reporting
  - Administration (Core and DTI)

The legacy: ECS1

- Executed during 2006-2007, deployed in Jul 2007
- Big problems with project organization and software architecture
  - System development continued for 14 months after deployment, some parts never deployed
  - Big problems with stability, performance and bugs.
- Problems with the tools used
  - IBM Process Server: heavy XML munching leading to bad performance
  - Therefore no long-running processes and human tasks used **à** used as a data flow engine, not a real BPMS
  - IBM Message Broker: used only for XML-XML conversion **à** not worth the overhead
- Architecture is not modular
  - Disparate technologies used for Core and DTI, different GUI technologies
  - All message munching is left to Process Server (Websphere Integration Developer)
  - Common steps such as validation, object construction, etc are not modularized
  - 50 disparate processes ("phases") of ~40 steps each: huge complexity, no reuse

  à Impossible to modify or extend
- ITT was subcontractor, so we knew the business and all problems first-hand

How do you approach this?

- Immediately (first 3 months) ran an architectural pilot
  - Took the hard decision to rewrite ECS1 (supported by client), instead of trying to build on a shaky foundation
  - Evaluated IBM vs open source
  - Picking tools was the "easy part": trust nothing, try everything, be modern but not bleeding edge
- Finding the architectural approach was the hard part:
  - Too complex data **à** generation of as many artifacts as possible is key to consistent implementation.
  - Too much data **à** share sub-objects, copy/pass references only
    - Share data beans (stateful Hibernate session) between all core components
  - Too much processing **à** strict code modularization and factoring
    - All process-independent steps are done outside of the BPMS
    - Screens are reused between core and DTI systems as much as possible
  - Too many Rules & Conditions (250, revised daily) **à** invented XPath based DSL, the same R&C are used in offline and GUI interactive validation
  - Complex processes (and some unclear/buggy TAXUD specs) **à** design processes visually, clarify with client numerous times, implement in BPMS

Tools and Technologies

- Servers
  - JBoss AS 5.0.1: application server
  - JBoss Messaging: message queuing
  - jBPM 3.3: business process execution
  - Informix Dynamic Server 10 FC8: database server (client requirement)
- Tools
  - JBoss Tools 3.0: development IDE
  - Hudson: continuous integration
  - CheckStyle: static analysis, coding conventions
  - Subversion source control, EclipseSVN
  - TestNG 5.8: unit testing
  - DBUnit: database testing
  - MS Project: planning
  - Jira: task tracking
  - Visio: process modeling
  - Word: specification writing
  - Wiki: engineering notes / collaboration

- Frameworks/libraries
  - JBoss Seam 2.1: application framework, IoC container
  - EJB 3.0: object model
  - JPA/Hibernate 3.3.2: persistence and O/R mapping
  - JSF 1.2: web framework
  - Facelets: JSF templating engine
  - RichFaces 3.3: JSF AJAX library
  - JiBX 1.1: XML/Java binding (serialization/deserialization)
  - Dozer 4.4.1: object graph cloning (deep copy), used sparingly
  - *JXPath 1.2: DSL for message assignments, R&C validation*
  - Saxon 9: XSLT transformations
  - Jasper reports 3.1.3: reporting, printing Single Admin Doc
  - Log4J: structured logging

Data Model

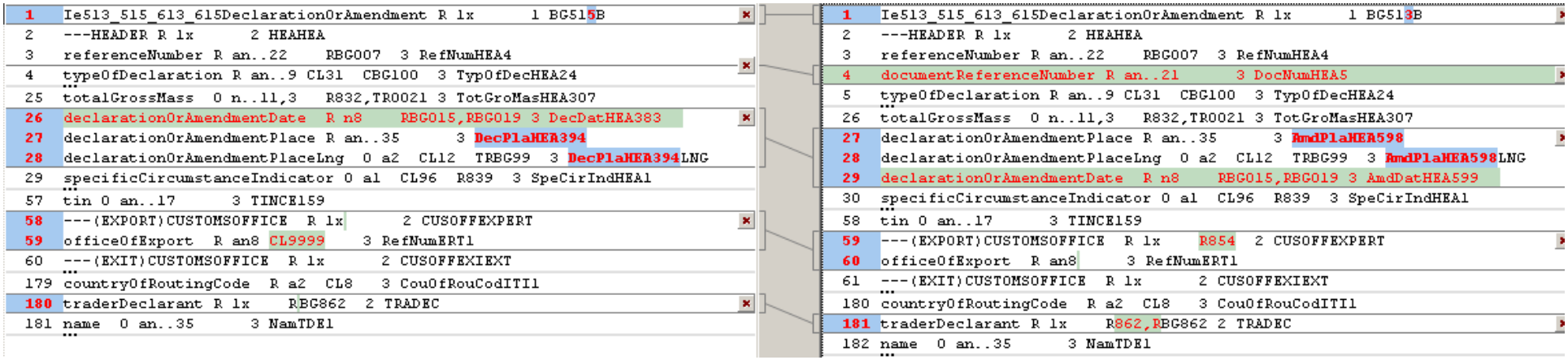
What's in the Data? (complexity)

- BG515B Export Declaration has 220 fields/groups like those below, nested 4 levels (header, shipment, goods item, package/container/document, 10 kinds of traders)
  - Columns: EN name, requiredness, type/repeatability, R&C, codelist, XML tag, Box number in paper document, BG name (label), BG description (tooltip)
- 45 messages averaging about 100 fields/groups each.
  - Imagine 220 chunks of fields like the one below! 150 printed pages

Name	R	Fmt	Rules	CL	Tag	Клетка	Име	Описание на значението на елемента
---(EXPORTER) TRADER	**D**	**1x**	**RBG028**		**TRAEXPEX1**	**2**	**Изпращач / Износител**	**Човекът или компанията, които предоставят пратките (стоките) на превозвач, за транспортирането им до Получателя.**
Name	O	an..35			NamEX17	2.1	Име	Наименование
Street and number	O	an..35			StrAndNumEX122	2.2	Улица	Улица, номер
Postal code	O	an..9			PosCodEX123	2.3	ПК	Пощенски код
City	O	an..35			CitEX124	2.4	Населено място	Населено място
Country code	O	a2		CL8	CouEX125	2.5	Държава	Код на държавата
NAD LNG	O	a2	TRBG99	CL12	NADLNGEX	2.6	Език	Език на тази група данни
TIN	R	an..17	RBG028		TINEX159	2.7	TIN	Уникален идентификационен номер на търговеца
Company Type	D	a..4	CBG004,RBG009	CL9556	ComTypEXP165	2.8	Вид компания	Вид компания (ЕТ, ООД, АД и пр). Не се попълва за физически лица.
Tax Directorate	D	n..4	RBG010	CL9513	TaxDirEXP167	2.11	Териториална Дирекция на НАП	Попълва се ако търговецът има регистрация по българското законодателство (БУЛСТАТ/ЕИК, ЛНЧ, ЕГН).
Tax Office	D	n..4	RBG010,RBG011	CL9514	TaxOffEXP168	2.12	Данъчна служба	Попълва се ако търговецът има регистрация по българското законодателство (БУЛСТАТ/ЕИК, ЛНЧ, ЕГН). Трябва да е подразделение на <Териториална Дирекция на НАП
---(CONSIGNEE) TRADER	**O**	**1x**	**R011**		**TRACONCE1**	**8**	**Получател**	**Човекът или компанията, посочени от Износителя, които получават пратките (стоките) от превозвача.**
Name	R	an..35			NamCE17	8.1	Име	Наименование
Street and number	R	an..35			StrAndNumCE122	8.2	Улица	Улица, номер
Postal code	R	an..9			PosCodCE123	8.3	ПК	Пощенски код
City	R	an..35			CitCE124	8.4	Населено място	Населено място
Country code	R	a2		CL8	CouCE125	8.5	Държава	Код на държавата
NAD LNG	O	a2	TRBG99	CL12	NADLNGCE	8.6	Език	Език на тази група данни
TIN	O	an..17			TINCE159	8.7	TIN	Уникален идентификационен номер на търговеца

Commonality in Data (Factoring)

- BG515B (declaration) and BG513B (amendment) differ by only 1%: 3 tag names (meaning is the same), order of 2 fields, 1 added field, and a few rules:



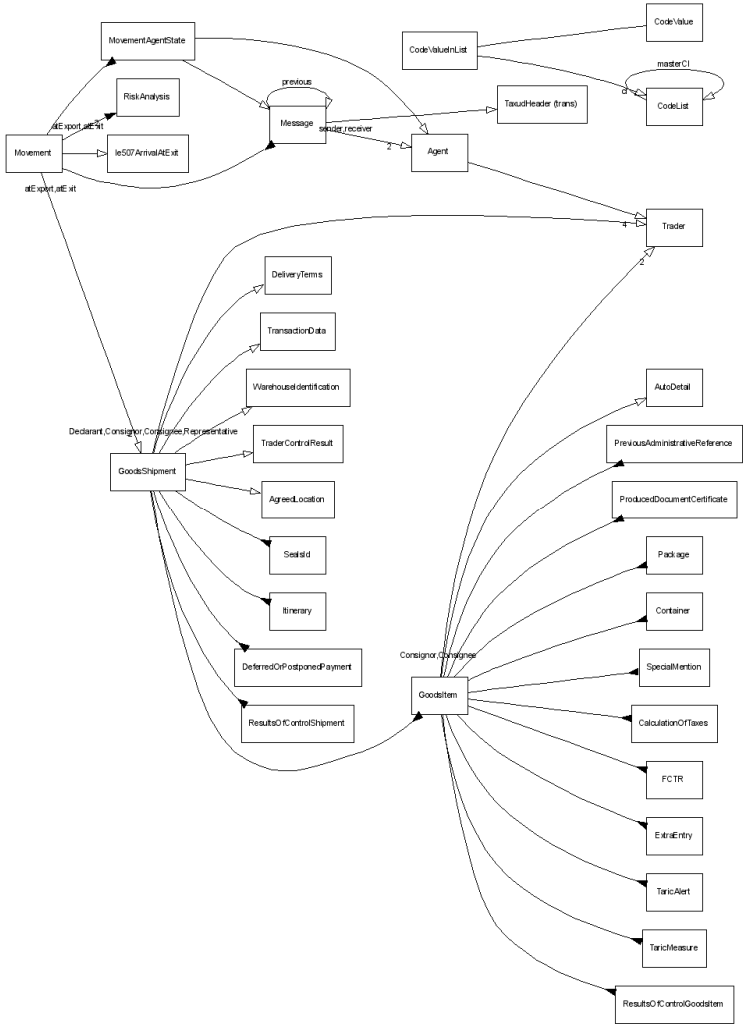
- So we map them to the same java bean class (Ie513_515_613_615DeclarationOrAmendment) and generalize data names/requiredness appropriately
- How to discover all commonality and create a properly factored schema?
  - Generate TXT message descriptions, Diff the appropriate pairs, move repeated groups and common parts into sub-objects, replace with references (FKs), generate mapping from several messages to the same generalized class
  - I.e.: pour, compare, slice, stir but don't shake. And think a lot (repeat, think not drink).

Data Model (Process)

- Developed starting from DDNXA Message Structure (MDB file), including fields, groups, types, repeatability, requiredness, R&C, XML tag, description
- Automatically generated readable tags (e.g. TRACONCE **à** traderConsignor).We don't want no Greek tags inside the system! (And this is the modern way of EMCS)
- Factored to expose and exploit all commonalities (normalization)
- Added own business objects (Agent, Message, Movement etc)
- Added presentation info: BG name, BG description, SAD box number
- Single schema used for: RDBMS, Java objects (data beans), Hibernate mapping, XSD, XML serialization/deserialization, screen generation, and numerous other things
- Formally described BG changes/additions to DTI messages, added ds:Signature
- Formally described CD changes: bugfixes, workarounds for CTP peculiarities
- Generated DTI Publication for the traders (Excel description and XSD)

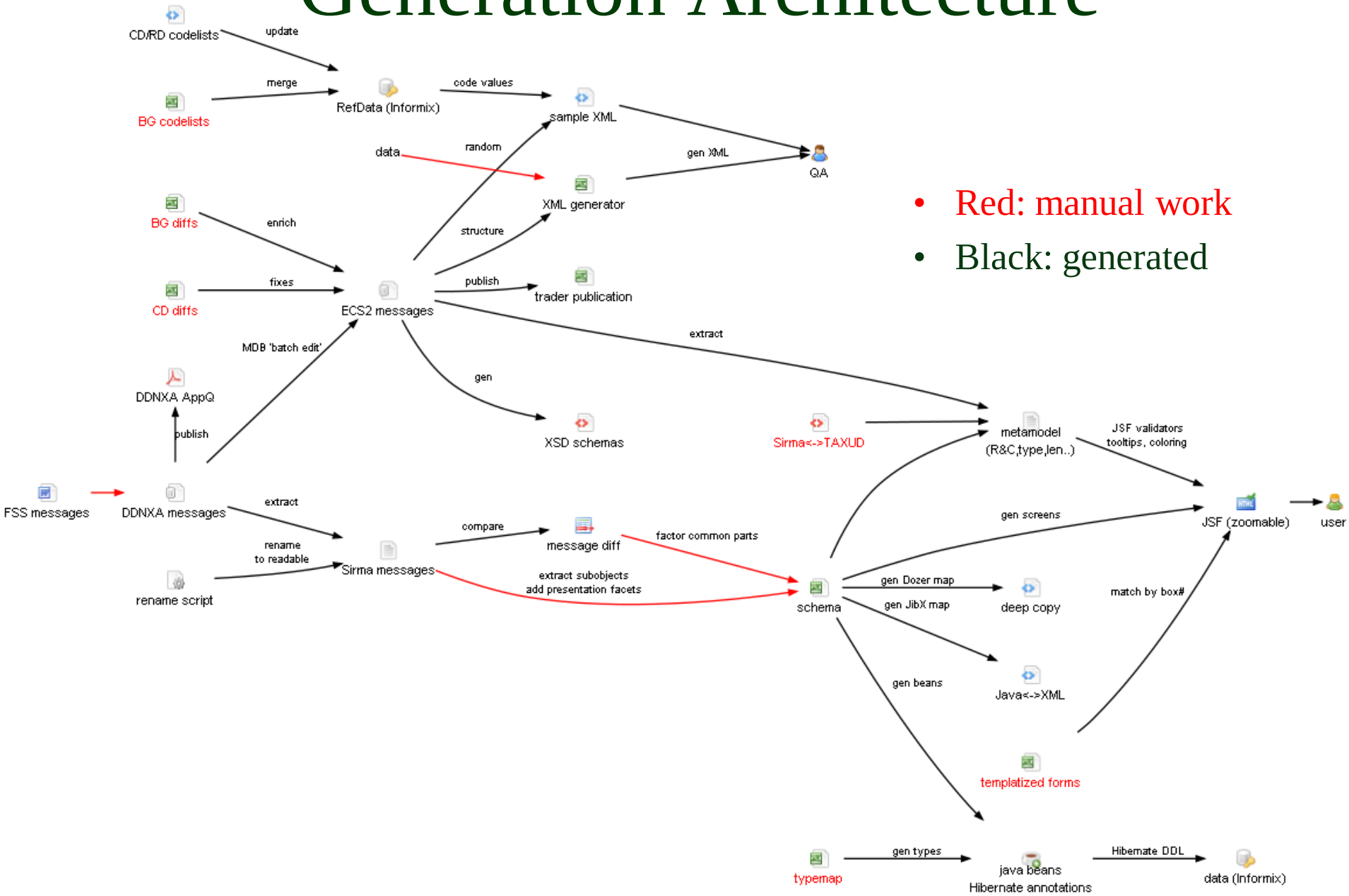
Data Model (Result)

- 93 tables, 750 fields. 45 messages mapped to 28 classes (e.g. Ie501_503_601_603DeclarationData carries 4 messages). Also covers reference data (RD/COL/UNA), risk analysis data, etc



Model-Driven Generation

Generation Architecture

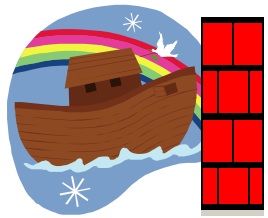


- Red: manual work
- Black: generated

Model-Driven Architecture

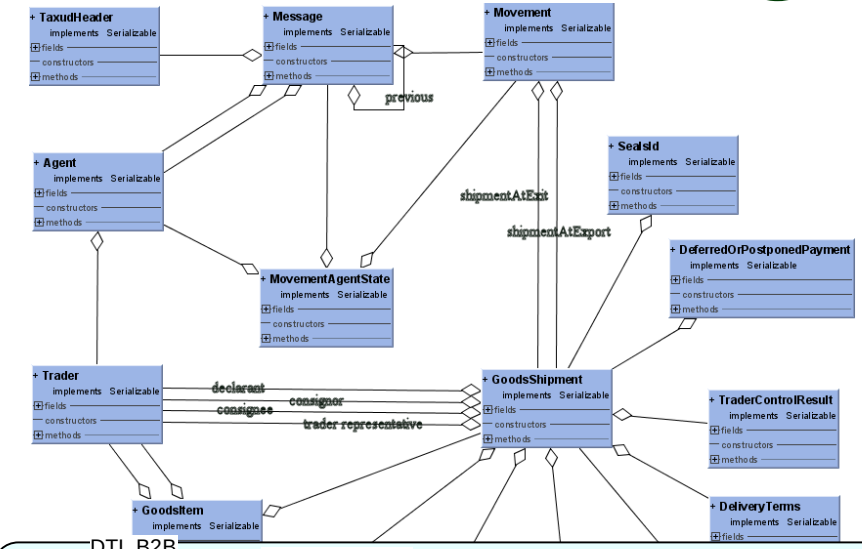
- You can see the ECS2 architecture is heavily model-driven
  - The models are highly pragmatic and specific
  - You don't need expensive or heavy tools
  - All you need is Excel and a couple goods hackers

  à Excel-Driven Software Engineering ™ ;-)
- UML models
  - We used sequence and activity diagrams complex module design
  - Class diagrams for illustration only
  - Data is given in MDB: we want to use it directly, not transcribe it in a UML tool (would be too much work for too little gain)
  - Transcription leads to bugs like
    "n..5 is maxInclusive=5" *that stop ships*
    (ECS1 example)

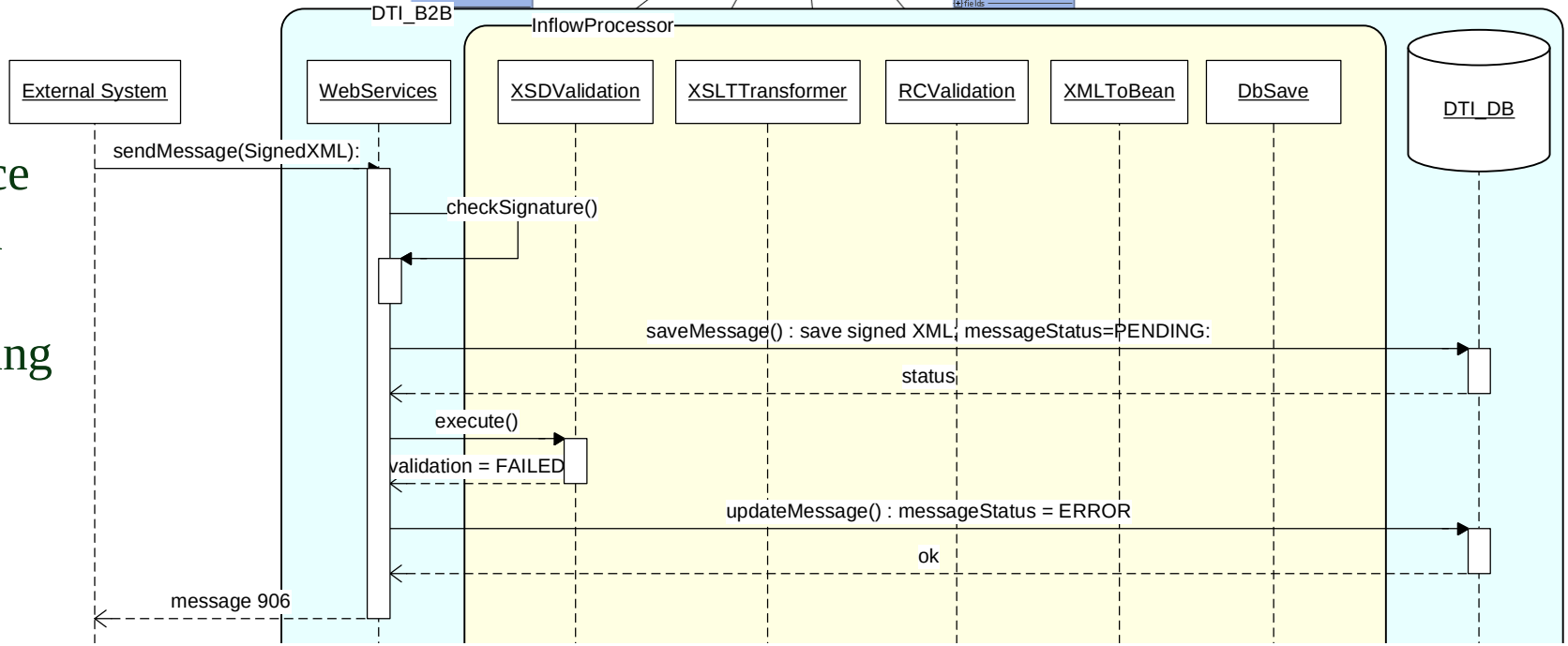


UML Modeling

- Class diagram showing part of the schema



- Sequence diagram for DTI processing



Reference Data

Codelists: as important as data model

Drive all presentation aspects of the application

Reference Data (codelists)

- All fixed values are managed in the RD part of the database (*two* tables)
  - TAXUD values are received from CS/RD and loaded with an incremental update algorithm
  - National values are managed in Excel and loaded with a generator: *highly customizable by the client*
  - Managing all RD in one place from day 1 was an excellent decision
- RD feeds all listboxes and CL validation and drives many other aspects of ECS2
- E.g. consider National codelist CL9058 "Movement State":

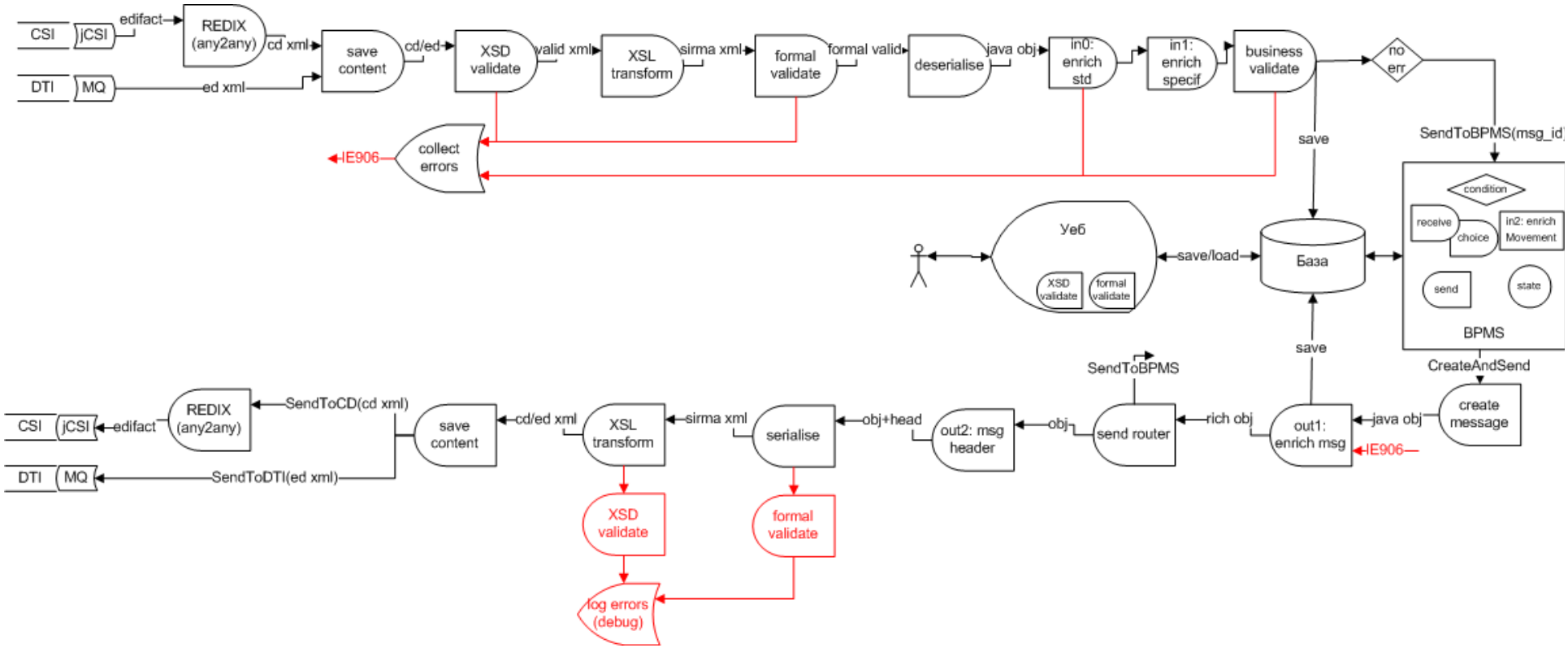
cl/value	descrEn	descrBg	extra	extra1	extra2	extra3	order	masterCl/	java
9058	Movement state	Състояние на движение	--show?	CL58				9057	CL_MOVEMENT_STATE
EXPREG	EXS Registered	ОДН регистрирана	1	L01			1101	XP	EXS_REGISTERED
EXPNRE	Not Released for export	Неразрешен износ	1	F07			1201	XP	EXP_NOT_RELEASED_FOR_EXPORT
EXPREL	Goods Released for Export	Разрешен режим	1	F02			1202	XP	EXP_RELEASED_FOR_EXPORT
EXPDAC	Diversion Accepted	Прието отклонение	1	F06			1204	XP	EXP_DIVERSION_ACCEPTED
EXPCAN	Cancelled	Анулирана	0	F09			1206	XP	EXP_CANCELLED
EXPSTP	Export Stopped, discrepancies at Exit	Износът спрян	1	F07			1207	XP	EXP_STOPPED
EXPEXP	Exported	Износът осъществен	1	F08			1208	XP	EXP_EXPORTED
FXPFUP	Follow-up	Проследяване	2	F11			1301	XP	FXP_FOLLOW_UP
FXPUFR	Under Exit Confirmation request	Под запитване за проследяване	2	F11			1302	XP	FXP_UNDER_EXIT_CONFIRMATION_REQUE
CXPRCO	Cancellation Requested by Customs Officer	Анулиране заявено от митнически служител	1				1401	XP	CXP_CANCELLATION_BY_OFFICER
CXPRTE	Cancellation Requested by Trader at Export	Анулиране заявено от търговец	1				1402	XP	CXP_CANCELLATION_BY_TRADER
CXPUCD	Under cancellation decision	Под решение за анулиране	2				1403	XP	CXP_UNDER_CANCELLATION_DECISION
EXTCRE	Created	Очаквана пратка	1	G02			2101	XT	EXT_CREATED

  - **cl/value**: state code stored in the database and handled by BPMS
  - **descrEn/Bg**: screen labels
  - **extra=show?**: determines whether to show Quick Search by this state, and in what color
  - **extra1=CL58**: determines mapping to TAXUD codelist for messages CD904/CD905
  - **order**: the order of Quick Search links on the screen (see shot at the bottom)
  - **masterCl**: for which office is this value applicable (eXPort or eXiT)
  - **java**: enum constant to be used in source (e.g. EXP_DIVERSION_ACCEPTED speaks more than "EXPDAC")

Търсене: ОДН регистрирана (72) , Разрешен режим (689) , Прието отклонение (22) , Износът спрян (31) , Износът осъществен (24) , Под запитване за проследяване (694)

ECS2 System Architecture

Backend Architecture: Pipelines



- All process-independent data munching, scrubbing and washing is done in one place, outside BPMS

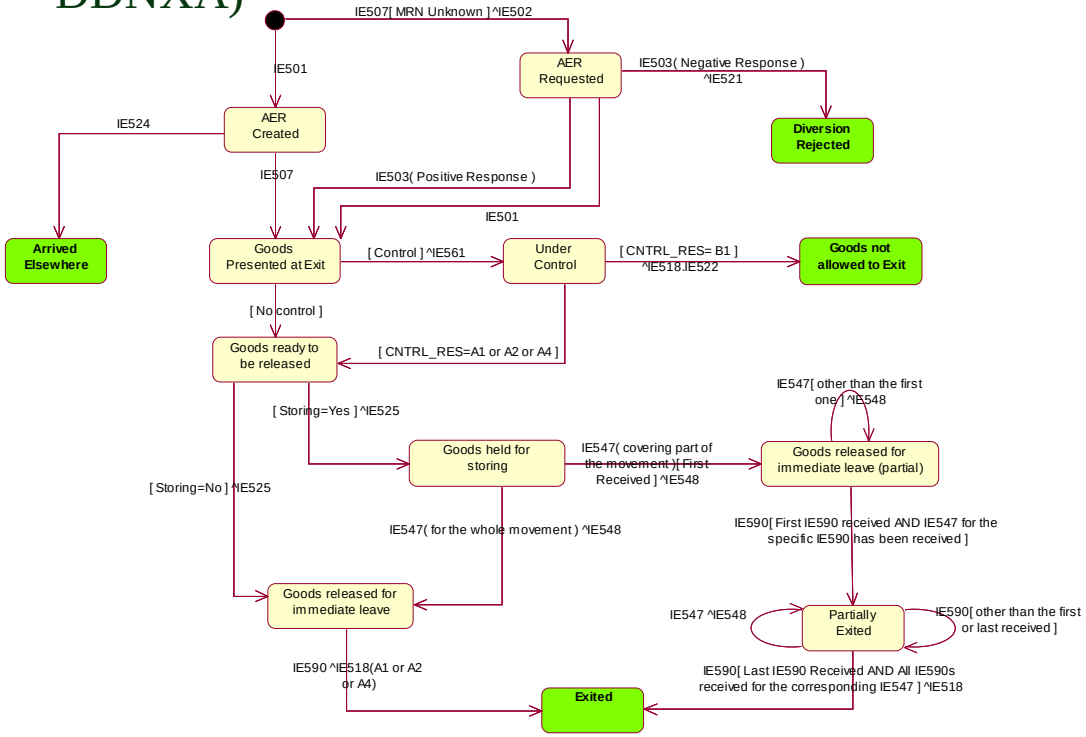
Backend Architecture: Processing

- All process-independent data munching, scrubbing and washing is done in one place, outside BPMS
- Inbound (inflow): REDIX transformation, XSD validation, XSL transformation, formal validation (R&C, CL, date/time), deserialization XML **à** object, enrichment (message assignments DSL), business validation (missing/duplicate MRN/LRN, EORI for TINs, SDM, state=sequence check), save to DBMS, error handling, creation of 906 error message
  - Inbound pre-Router and Convertor: decides whether to pass to ECS2 or ECS1 (complex coexistence requirements)
- If BPMS accepts the message, it contributes to the main object (Movement), and parts are copied in outgoing messages
  - Only references are copied, objects are shared between pipelines (stateful Hibernate session), so DBMS queries are minimized
  - XML is used only for input/output. Inside the backend, java objects are shared
  - Example: an incoming message is deserialized to an object and saved to the database. When accepted by BPMS, some of its data is coped as references (e.g. movement/shipmentAtExit=501/shipment). The BPMS then decides to send 2 outgoing messages, which are created by copying from the incoming and from Movement. The outgoing messages are saved and queued. Database transaction is closed, committing all saves.
  - During all of this *no select statements are executed*. Only references are copied and only the minimum possible number of inserts are executed.
  - Compare this to the heavy XML copying/parsing/xpathing done by WID/PS involving its Business Objects
- Outbound (outflow): message creation, header creation, serialization object **à** XML, XSL transformation, REDIX transformation, optional validation (for debugging). Outbound routing:
  - CD message to BG ("loopback"): never serialized. It's already written to the DB, so just signal BPMS to process it.
  - CD message to another country: Redix for conversion to EDIFACT, then jCSI to queue it in CCN/CSI
  - BG message to trader (DTI): queue to DTI B2B agent that stores it to DTI DB for final delivery
  - BG message to BIMIS: locate the distributed BIMIS instance corresponding to the receiving customsOffice, deliver by webservice

ECS2 Business Processes

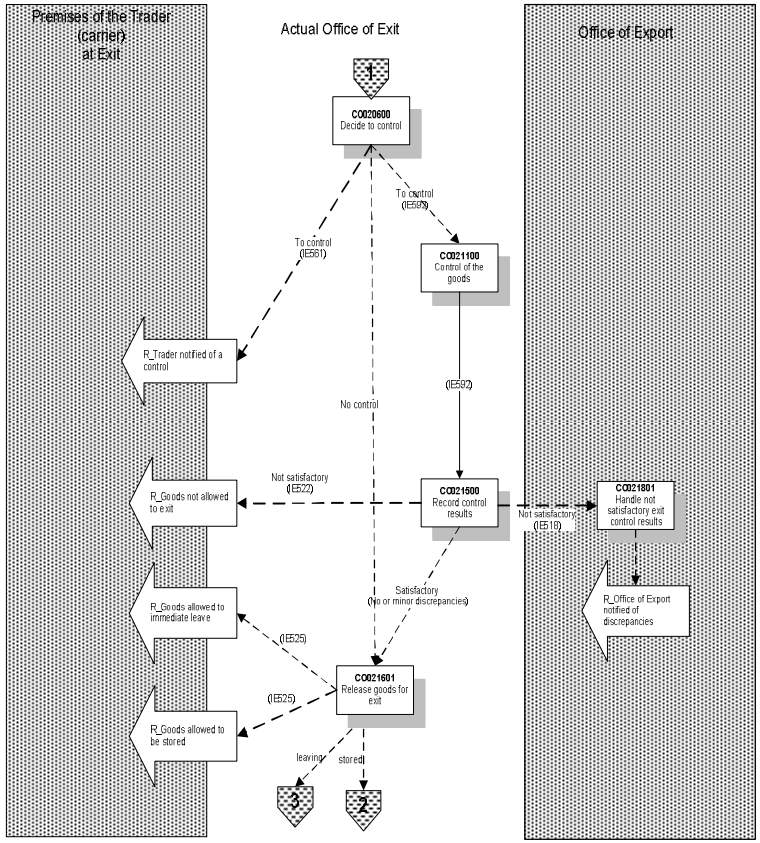
Business Process Modeling

- Coupled with state transition diagrams like that (from DDNXA)



- TAXUD provides global process diagrams like this (from FSS)
- The boxes are "EBP": linear atomic processing

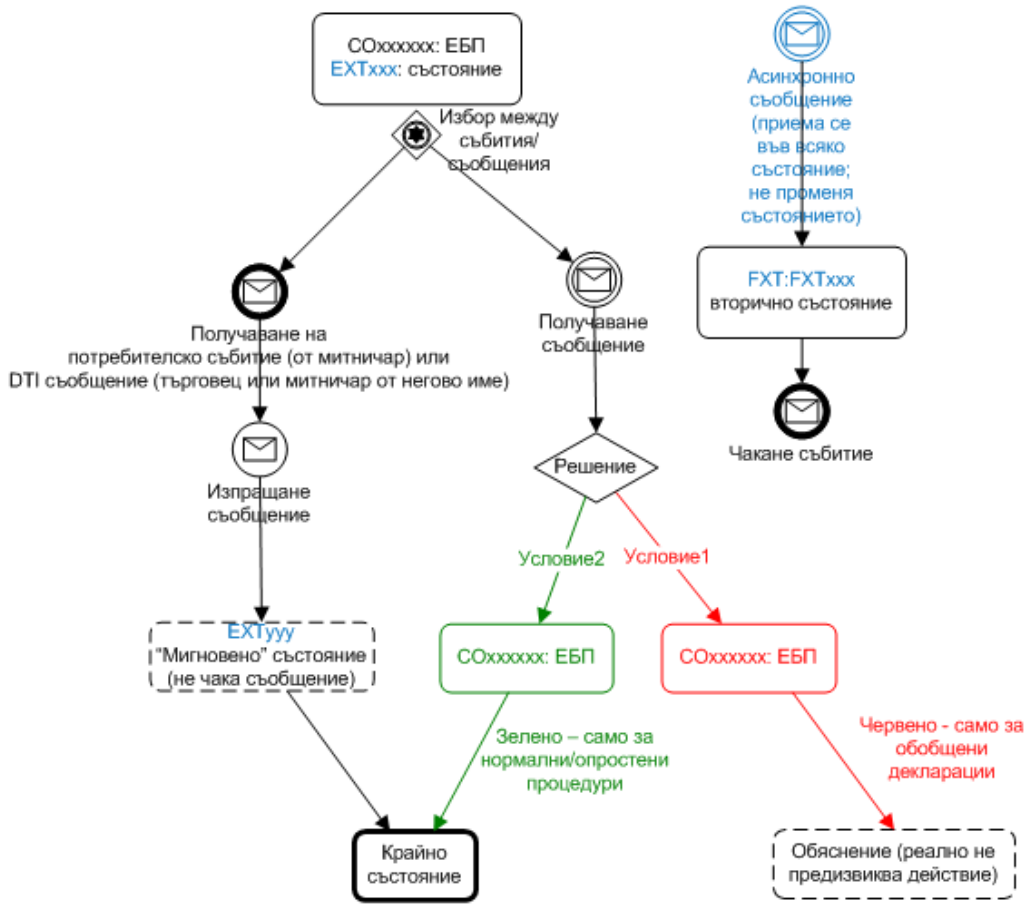
CO02 - Process Arrival at Office of Exit part II



- Ambiguities, discrepancies and bugs exist

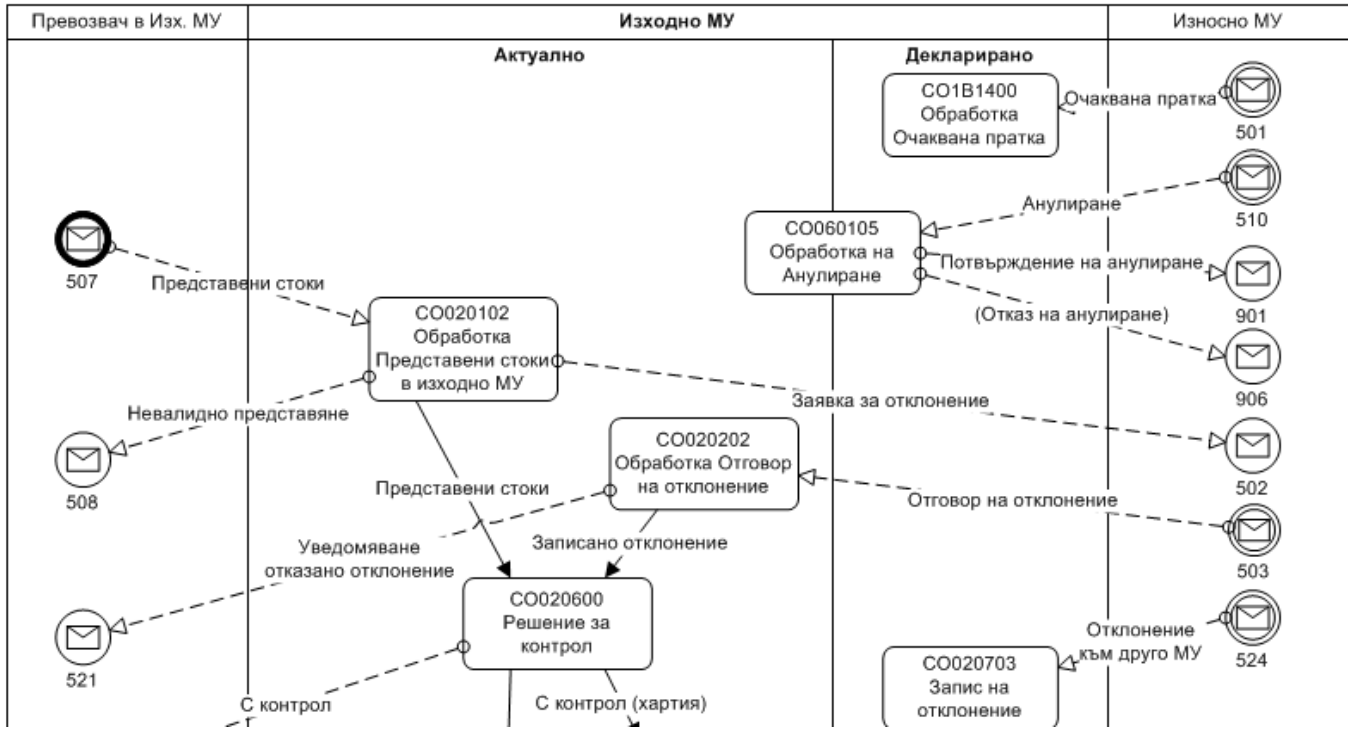
BPM: Primitives (Notation)

- Selected a minimal set of primitives (based on BPMN) that are easy to implement in the chosen BPMS yet sufficient:
  - EBP
  - send message
  - wait to receive message
  - wait for user event: *ALL user actions are recorded just like messages*
  - wait for timer
  - choice (wait for several message types)
  - decision (condition)
  - set state
  - async subprocess
- Turns out there are multiple states per Movement: per role, and primary/secondary
- Modeled processes with Visio
  - Not using BPEL. And proud of it **J**
  - *Very well received by client because they understand it*
  - Two levels: Context and Detail

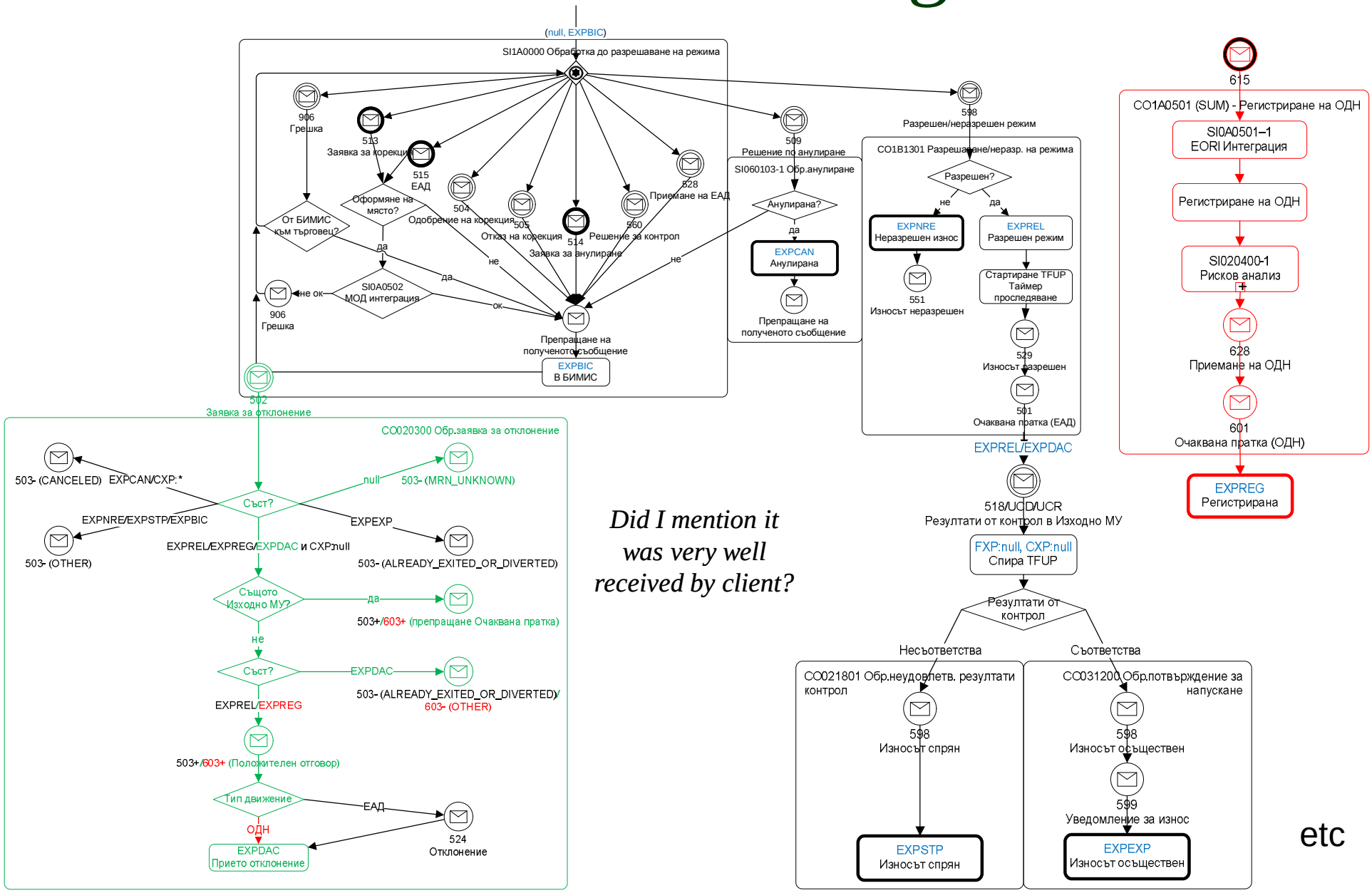


BPM: Context Diagram

- BPM with the client takes TAXUD as a basis, then should reflect all clarifications, decisions and national requirements
- Context diagram shows all messages and EBP for one agent (customs office or trader type)
  - Focuses on one role at the time, i.e. what to implement for that agent
- The global process is implemented through message exchange and choreographed state change in independent systems
- Shows only the messages exchanged by the other agents
  - Uses swim-lanes for clarity
  - *Very well received by client*



BPM: Detail Diagram



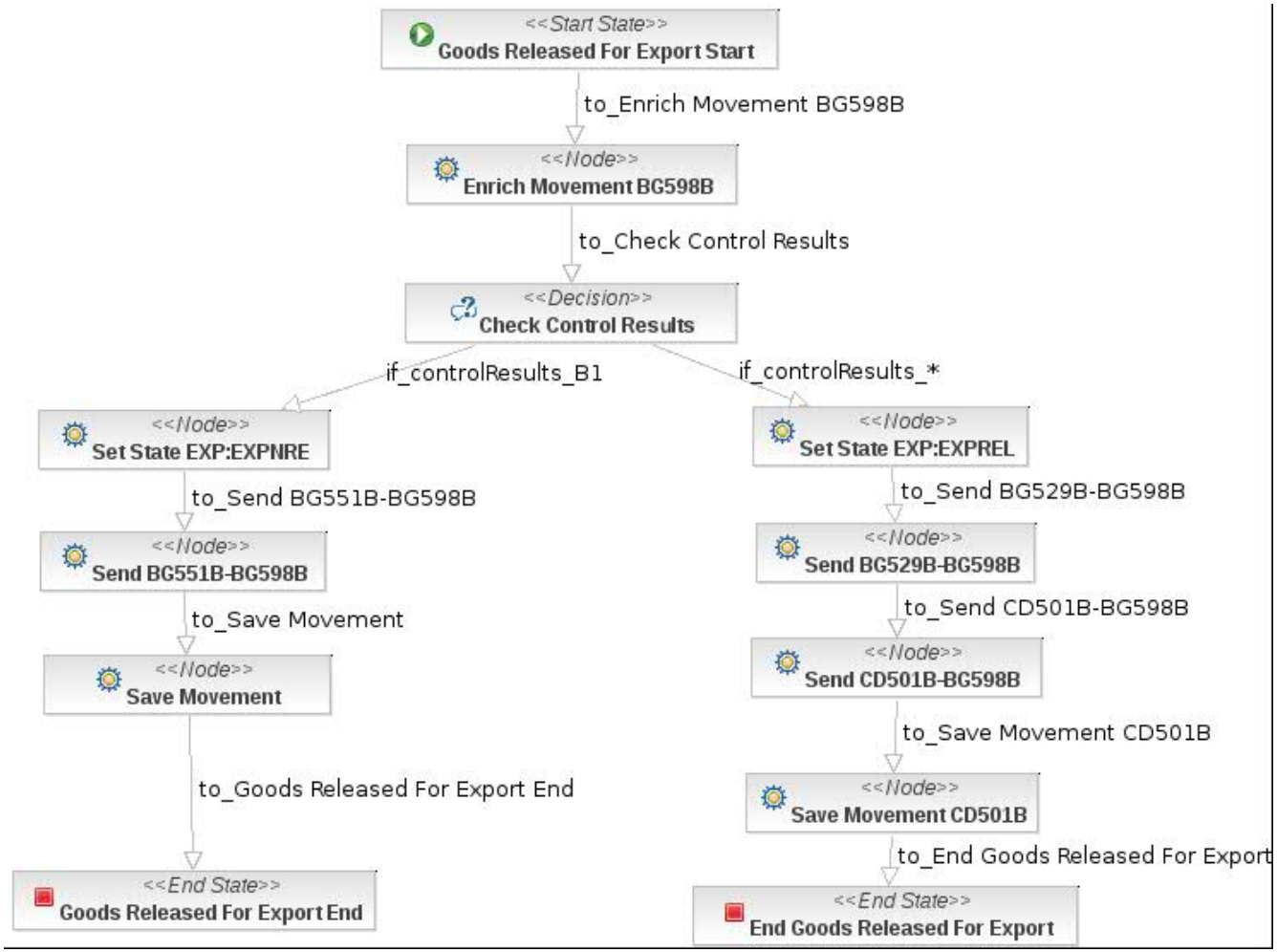
Did I mention it was very well received by client?

etc

BPM: Implementation

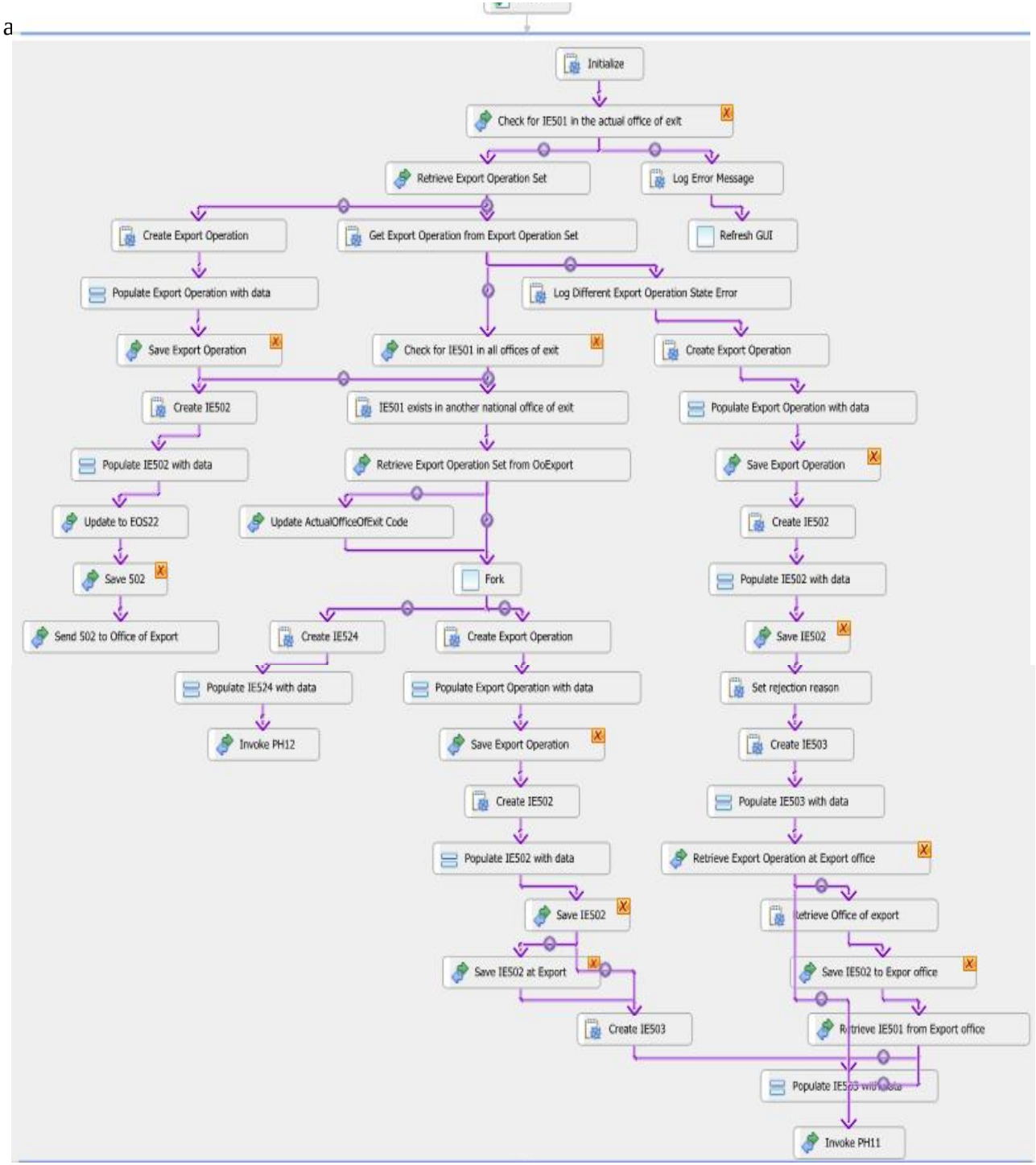
- Implementation in jBPM follows closely the process specification.
- Most importantly, it is at almost the same level of abstraction (2-3x the nodes) because:
  - It doesn't indulge in low-level data munging
  - All common steps are outside of BPMS

Ø Contrast with next page



ECS1 BPM Impl.

- In ECS1, *one* EBP (CO020300 Handle AER Request) is implemented in about 50 nodes
- My oh my! I'd rather just write Java!
  - (Actually you don't want to see the Java code inside many of these nodes)
- And no long-running processes are used



XPath-based Domain Specific Languages (DSL)

Message Assignments

- How do you implement the *in*, *in2* and *out* steps in the pipelines? They are simple assignments
  - But there are 134 assignment groups of 5-10 assignments each (over 900 total): how do you manage them?
  - You don't want to write this in Java
- Use a DSL: it's all the rage nowadays
  - XPath is ideal for this purpose. JXPath library groks java objects.
  - Now the BA and App Dev (BPMS) teams can share the responsibility for these assignments
    - "a/b=c/d" looks nicer than "getA().setB(getC().getD())"
    - More importantly, it's much easier to understand and manage
  - Embellish as needed with conditionals, "set if non-null", comments…

Message Assignment examples

XPath-ish	English
-CD501B:in sender/role="EXP" sender/customsOffice=shipment/officeOfExport sender/country=sirma:countryFromOffice(sender/customsOffice)	When 501 is received, enrich it (even if it is invalid or unacceptable, we want to save it) The sender is Office of Export The sender is a customs office, and its code is stored in the shipment data The country of that sender is extracted from the office code (this is an Extension Function)
-CD501B:in2 movement/isSummary="false" movement/shipmentAtExit=shipment	When 501 is accepted by BPMS Set the Movement as "not summary" (because summary movements are indicated by 601) Save the shipment data from 501 to the movement (the slot concerning Office of Exit)
-CD501B:out *filter shipment and remove national elements shipment=movement/shipmentAtExport sirma:mapExpTo501()	To create an outgoing 501 (quite a program here, so we have a comment) Set message shipment data to the movement shipment data (from slot of Function call that deep-copies shipment and modifies it to filter out national fields and array elements. (The only deep-copy in the whole system)

- Then it grows on you. Why can't we use the same to capture interactive Web context data?
  - Indeed we can:

XPath-ish	English
-*:web dtOfPreparation=$now !sender/customsOfficer=$webContext/userId	When ANY message is created by the user of ECS2 Web (customs officer) Capture NOW in the universal message field "datetime of preparation" And capture the current user as the message Sender. Instant audit trail!
-BG507B:web arrivalNotificationDate=$now !actualOfficeOfExit=$webContext/customsOffice	When 507 is created by the user of ECS2 Web (customs officer) or DTI Web (trader) Capture NOW in a message-specific field (507 is Arrival Notification) Capture the current Customs Office if present (that's in ECS2 Web) •A customs office cannot register 507 for another office

Rules & Conditions

- ECS2 has about 250 rules, and BG ECS2 adds 50 more. Applied to over 2000 fields
- Add cross-field validations that cannot be expressed in XSD. Example:

```
C060: IF 'Kind of packages' (Box 31) indicates 'BULK' ('VQ', 'VG', 'VL')
 THEN
  'Marks & numbers of packages'(Box 31) = 'O'
  'Number of packages' can not be used
  'Number of Pieces' (box 31) =  can not be used
 ELSE IF 'Kind of packages' (Box 31) indicates 'UNPACKED'('NE','NF',NG')
 THEN 'Marks & numbers of packages' (Box 31) = 'O'
  'Number of packages'(box 31') can not be used
  'Number of Pieces' (box 31) = 'R'
 ELSE 'Marks & numbers of packages' (Box 31) = 'R'
  'Number of packages'(box 31') = 'R'
  'Number of Pieces'(box 31) can not be used
```

- We introduced a classification inspired by XForms, with some extensions:

T	Type	XForms	GUI effect
R	Required	Required	Yellow if empty
N	Not applicable	Not Relevant	Grayed out
C	Calculate	Calculate	Calculated static text
X	Constraint	Constraint	Red if false
U	Unchangeable	Readonly	Static text
I	Init (calculate once)		None

- Our implementation uses a JXPath DSL (*again!*):

```
C060.shipment/goodsItem/package/numberOfPackages.N
  sirma:isBulk(../kindOfPackages) or sirma:isUnpacked(../kindOfPackages)
  EN Not applicable when "Kind of Packages" is "%s" % ../kindOfPackages
  BG Неприложимо когато "Вид пакети" е "%s"
C060.shipment/goodsItem/package/numberOfPieces.N
  ../kindOfPackages and (sirma:isBulk(../kindOfPackages) or
    not(sirma:isUnpacked(../kindOfPackages)))
  EN Not applicable when "Kind of Packages" is "%s"  % ../kindOfPackages
  BG Неприложимо когато "Вид пакети" е "%s"
C060.shipment/goodsItem/package/numberOfPackages.R
  ../kindOfPackages and not(sirma:isBulk(../kindOfPackages) or
    sirma:isUnpacked(../kindOfPackages))
  EN Required when "Kind of Packages" is "%s" % ../kindOfPackages
  BG Задължително когато "Вид пакети" е "%s"
C060.shipment/goodsItem/package/marksNumbersOfPackages.R
  ../kindOfPackages and not(sirma:isBulk(../kindOfPackages) or
    sirma:isUnpacked(../kindOfPackages))
  EN Required when "Kind of Packages" is "%s" % ../kindOfPackages
  BG Задължително когато "Вид пакети" е "%s"
C060.shipment/goodsItem/package/numberOfPieces.R
  sirma:isUnpacked(../kindOfPackages)
  EN Required when "Kind of Packages" is "%s" % ../kindOfPackages
  BG Задължително когато "Вид пакети" е "%s"
```

- The error messages to be returned (in 2 languages) are also given above
- sirma:isBulk() and sirma:isUnpacked() are obvious Extension Functions for this rule
- If you know the data model and a bit of XPath, the implementation as easy as English. Even I can do it!

Rules & Conditions

- Do you want to write rules in Java? Don't!
- The same rule is implemented in 60 lines of dense Java in the Standard Transit Testing Application (STTA) by Intrasoft
  - Furthermore, our rules can be used both for offline validation, and GUI interaction
  - The STTA rules can be used only for validation and returning error (906)

```
public class C060 extends Rules {

  private String kindOfPackValue;
  private String errorDescriptionStr;

  public boolean isRuleValid(Document doc) {

    ruleName = "C060";

    NodeList listOfGDS = doc.getElementsByTagName("GOOITEGDS");
    int numberOfGDS = listOfGDS.getLength();

    String tags[] = {"NumOfPacGS24", "NumOfPieGS25", "MarNumOfPacGS21"};
    String description[] ={"Number Of Packages", "Number Of Pieces", "Marks and Numbers of Packages"};
    int id[] = new int[3];
    for (int k= 0; k < tags.length; k++) {
      FieldsInfo info = new FieldsInfo();
      info.setTagName(tags[k]);
      info.setTagDescription(description[k]);
      fieldInfo.addElement(info); // add a FieldsInfo object to the fieldInfo Vector
    }

    for (int i = 0; i < numberOfGDS; i++) {
      Node GDSNode = listOfGDS.item(i);
      NodeList listPackages = ((Element) GDSNode).getElementsByTagName("PACGS2");
      int numberOfPackages = listPackages.getLength();

      for (int j = 0; j < numberOfPackages; j++) {
        Node packagesNode = listPackages.item(j);

        kindOfPackValue = getTagValue((Element) packagesNode, "KinOfPacGS23");

        if (kindOfPackValue != null) {
          if (kindOfPackValue.equals("VQ") || kindOfPackValue.equals("VG") ||
              kindOfPackValue.equals("VL") || kindOfPackValue.equals("VY") ||
              kindOfPackValue.equals("VR") || kindOfPackValue.equals("VO")) {
            id[0] = FieldsInfo.NOT_USED;
            id[1] = FieldsInfo.NOT_USED;
            id[2] = FieldsInfo.OPTIONAL;
            errorDescriptionStr = "Since the GOOITEGDS(" + (i + 1) + ").PACGS2(" + (j + 1) + ").KinOfPacGS23 " +
                                  "has a value of '" + kindOfPackValue +
                                  "' which indicates BULK ('VQ', 'VG', 'VL', 'VY', 'VR', 'VO'), the GOOITEGDS(" +
                                  (i + 1) + ").PACGS2(" + (j + 1) + ").";
          }
          else if (kindOfPackValue.equals("NE") ||
              kindOfPackValue.equals("NF") ||
              kindOfPackValue.equals("NG")) {
            id[0] = FieldsInfo.NOT_USED;
            id[1] = FieldsInfo.REQUIRED_NON_ZERO;
            id[2] = FieldsInfo.OPTIONAL;
            errorDescriptionStr = "Since the GOOITEGDS(" + (i + 1) + ").PACGS2(" + (j + 1) + ").KinOfPacGS23 " +
                                  "has a value of '" + kindOfPackValue +
                                  "' which indicates UNPACKED ('NE', 'NF', 'NG'), the GOOITEGDS(" + (i + 1) +
                                  ").PACGS2(" + (j + 1) + ").";
          }
          else {
            id[0] = FieldsInfo.REQUIRED;
            id[1] = FieldsInfo.NOT_USED;
            id[2] = FieldsInfo.REQUIRED;
            errorDescriptionStr = "Since the GOOITEGDS(" + (i + 1) + ").PACGS2(" + (j + 1) + ").KinOfPacGS23 " +
                                  "has a value of '" + kindOfPackValue + "', the GOOITEGDS(" + (i + 1) +
                                  ").PACGS2(" + (j + 1) + ").";
          }

          ((FieldsInfo) fieldInfo.elementAt(0)).setIdentification(id[0]);
          ((FieldsInfo) fieldInfo.elementAt(1)).setIdentification(id[1]);
          ((FieldsInfo) fieldInfo.elementAt(2)).setIdentification(id[2]);

          checkField((Element) packagesNode, fieldInfo); // check if fields exist in doc

          if (errorPointer != null) {
            for (int x = 0; x < errorPointer.size(); x++) {
              error = new RuleError();
              error.setErrorType(15);
              error.setErrorReason(this.toString());
              error.setOriginalAttribOfNode("N/A");
              error.setRuleName(this.toString());
              error.setErrorPointer("GDS(" + (i + 1) + ").GS2(" + (j + 1) + ")." +
                  ((FieldsInfo) errorPointer.elementAt(x)).getTagDescription());
              error.setErrorDescription(errorDescriptionStr + (String) errorDescriptionList.get(x));
              errors.addElement(error);
            }
            errorPointer.removeAllElements();
          }
        }
      } // for loop for packages
    } // for loop for good items

    return (!(errors.size() > 0));
  } // is rule valid method

}
```

Rules Used for Validation and GUI

- Consider this rule:

```
C010:IF first digit of "Transport mode at border" (box 25) = 2, 5 or 7
 THEN "Nationality crossing border" (box 21) is Optional ELSE Required

C010.shipment/nationalityOfMeansOfTransportCrossingBorder.R
not(sirma:RexexpMatch(../transportModeAtBorder,"^[257].*
  EN Required when "Transport mode at border" does not start with 2, 5 or 7
  BG Задължително когато "Вид транспорт на границата" не започва с 2, 5 или 7
```

- The implementation is simple (nothing for "Opt")
- While editing: box 25 doesn't start with [257] so the field is Required (yellow). In this way the user can see all erroneous fields at a glance. *AJAX implementation: submits&receives only the changes.*
- If the user hovers over the field, its tooltip gives error description (red text) and all details about the field (tag name, box number, description, format, even codelist)

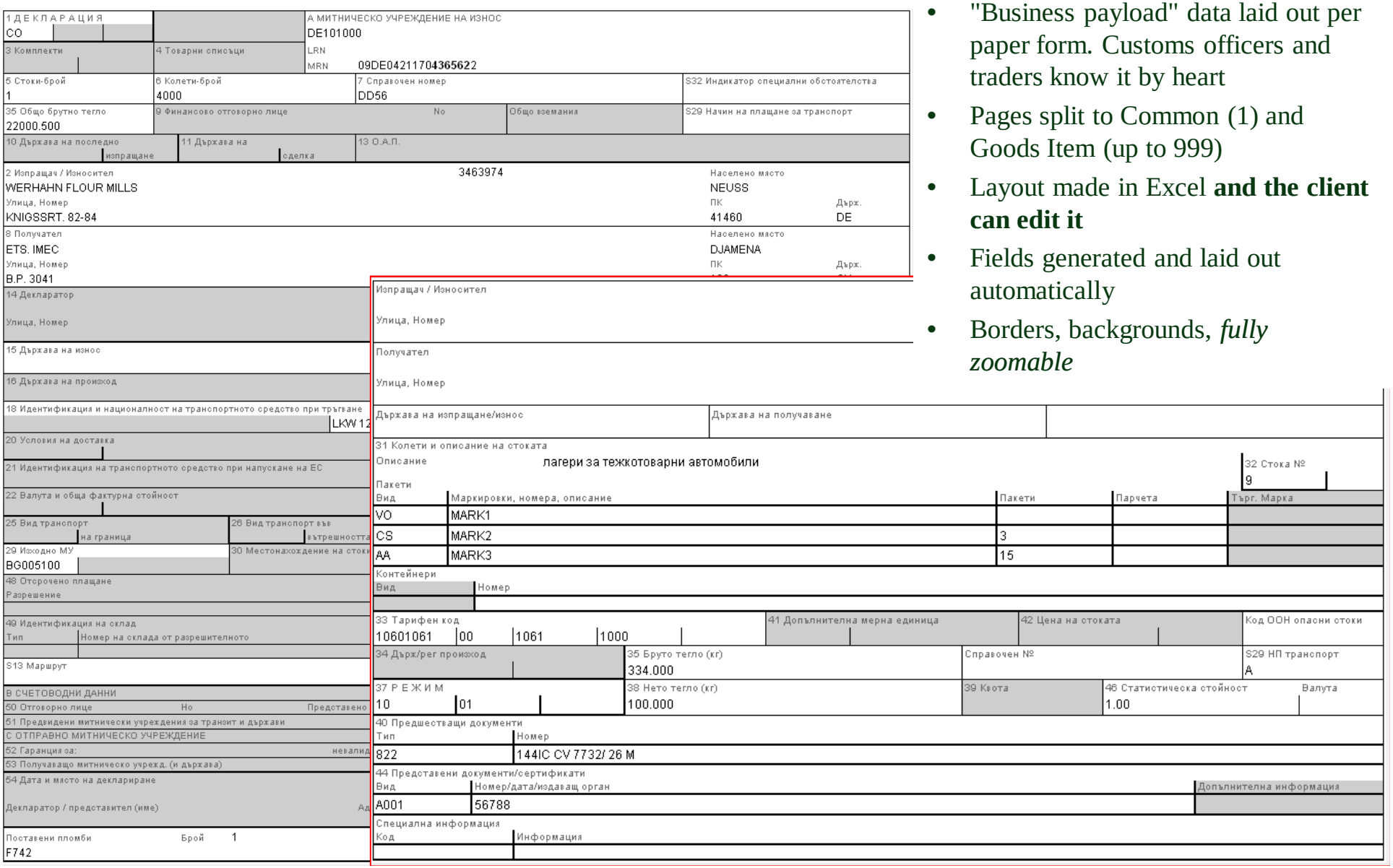
- If an incoming message has this defect, the *same rule implementation* returns appropriate error message
  - Bilingual ErrDes handled
  - CD906A doesn't *have* ErrDes, so we map to OriAttVal

```
<BG906B>
   <MesSenMES3>NECA.BG</MesSenMES3>
   <SenIdeCodQuaMES4>BG005100</SenIdeCodQuaMES4>
   <MesRecMES6>TRA.*</MesRecMES6>
   <RecIdeCodQuaMES7>BGB83122253072ZZZ</RecIdeCodQuaMES7>
   <DatOfPreMES9>20091007</DatOfPreMES9>
   <TimOfPreMES10>1717</TimOfPreMES10>
   <IntConRefMES11>31d9fe70b34c11</IntConRefMES11>
   <TesIndMES18>0</TesIndMES18>
   <MesIdeMES19>31d9fe70b34c11</MesIdeMES19>
   <MesTypMES20>BG906B</MesTypMES20>
   <OriMesIdeMES22>69307e20b34315</OriMesIdeMES22>
   <FUNERRER1>
      <ErrTypER11>13</ErrTypER11>
      <ErrXPaER12>HEAHEA/NatOfMeaOfTraCroHEA87</ErrXPaER12>
      <ErrReaER13>C010</ErrReaER13>
      <ErrDesER15>Задължително когато "Вид транспорт на границата" не започва с '2', '5' или '7'</ErrDesER15>
   </FUNERRER1>
</BG906B>
```

GUI Frontend

JSF+AJAX, lots of generation

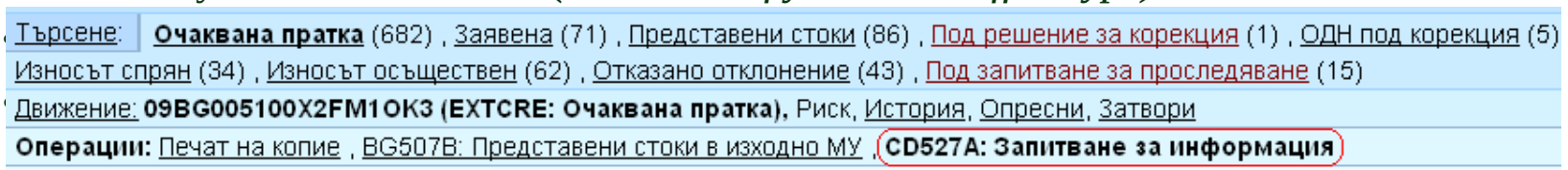
GUI Follows the Single Administrative Document



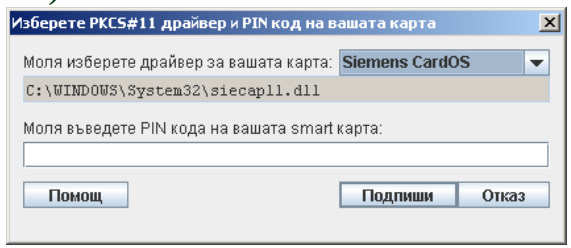
- "Business payload" data laid out per paper form. Customs officers and traders know it by heart
- Pages split to Common (1) and Goods Item (up to 999)
- Layout made in Excel **and the client can edit it**
- Fields generated and laid out automatically
- Borders, backgrounds, *fully zoomable*

GUI: Features

- Hired a Usability consulting company (still most of the ideas are our own)
- Context-dependent "navbars" show important info
  - User, office, role
  - Number of movements per state in the current office, which are Quick Search links *(from CL9058)*
  - Currently selected movement *(sticks even if you switch office type)*, with links to its data



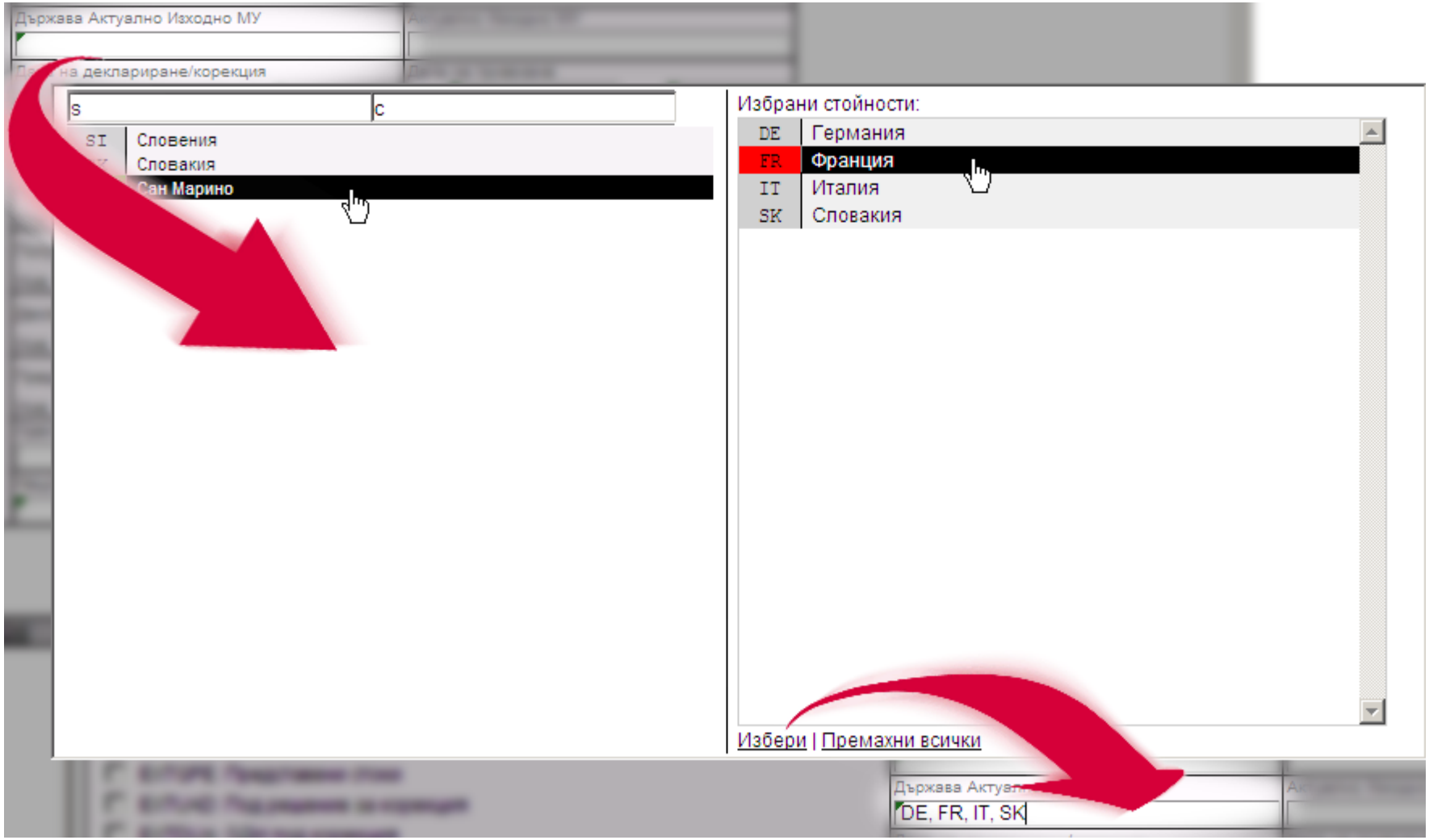
- Full type, length, regexp, requiredness, CL, R&C validation (yellow: required unfilled, red: erroneous). JSF validators are generated from message definitions and R&C metamodel
  - Tooltip gives all details about each field (tag, name, description, box number, error)
- Electronic signatures (PKI certificates) used for trader access and document signing



- The *same* business data forms are used in ECS2 Web and DTI Web.
  - The navbars ("chrome") vary per user kind (customs officer, trader) and roles/permissions
- Goods Pager: efficiently handles up to 999 goods, add item [+], direct jump, erroneous list

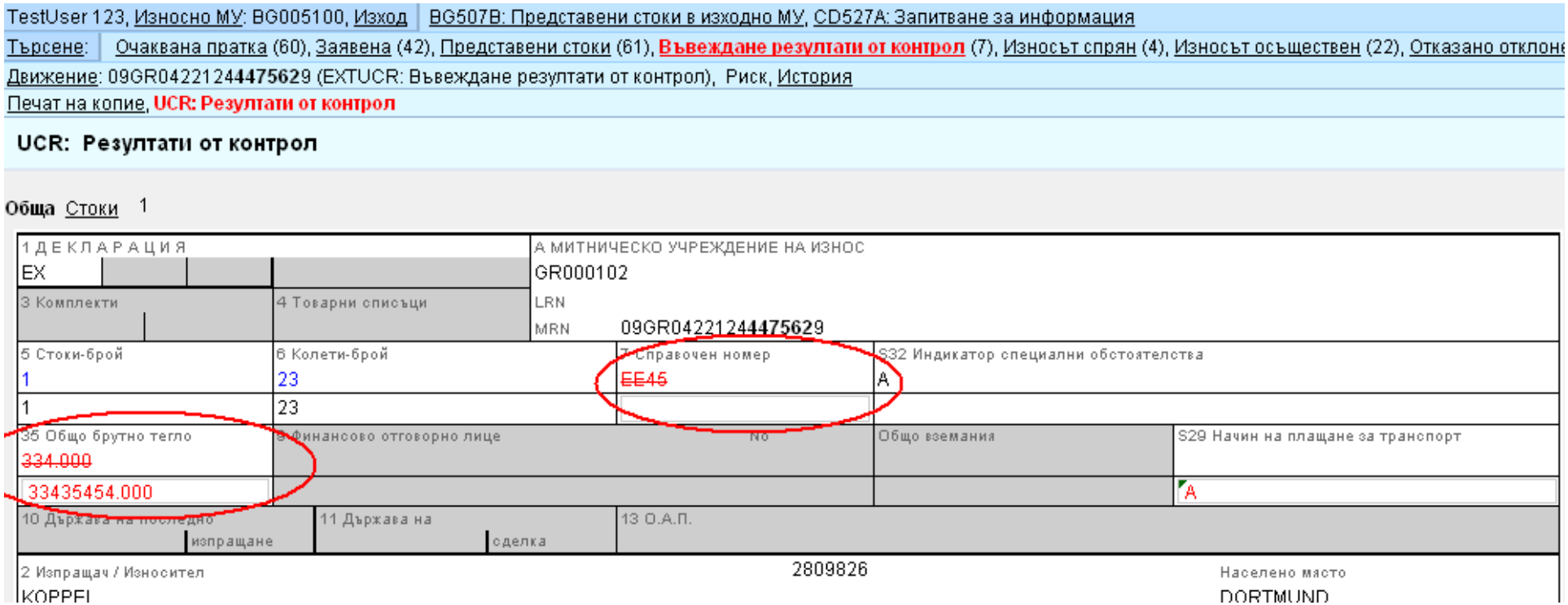
GUI: Features

- Codelist popup (RD selector): allows direct entry (speed) or selection (convenience), always validates, allows multi-selection

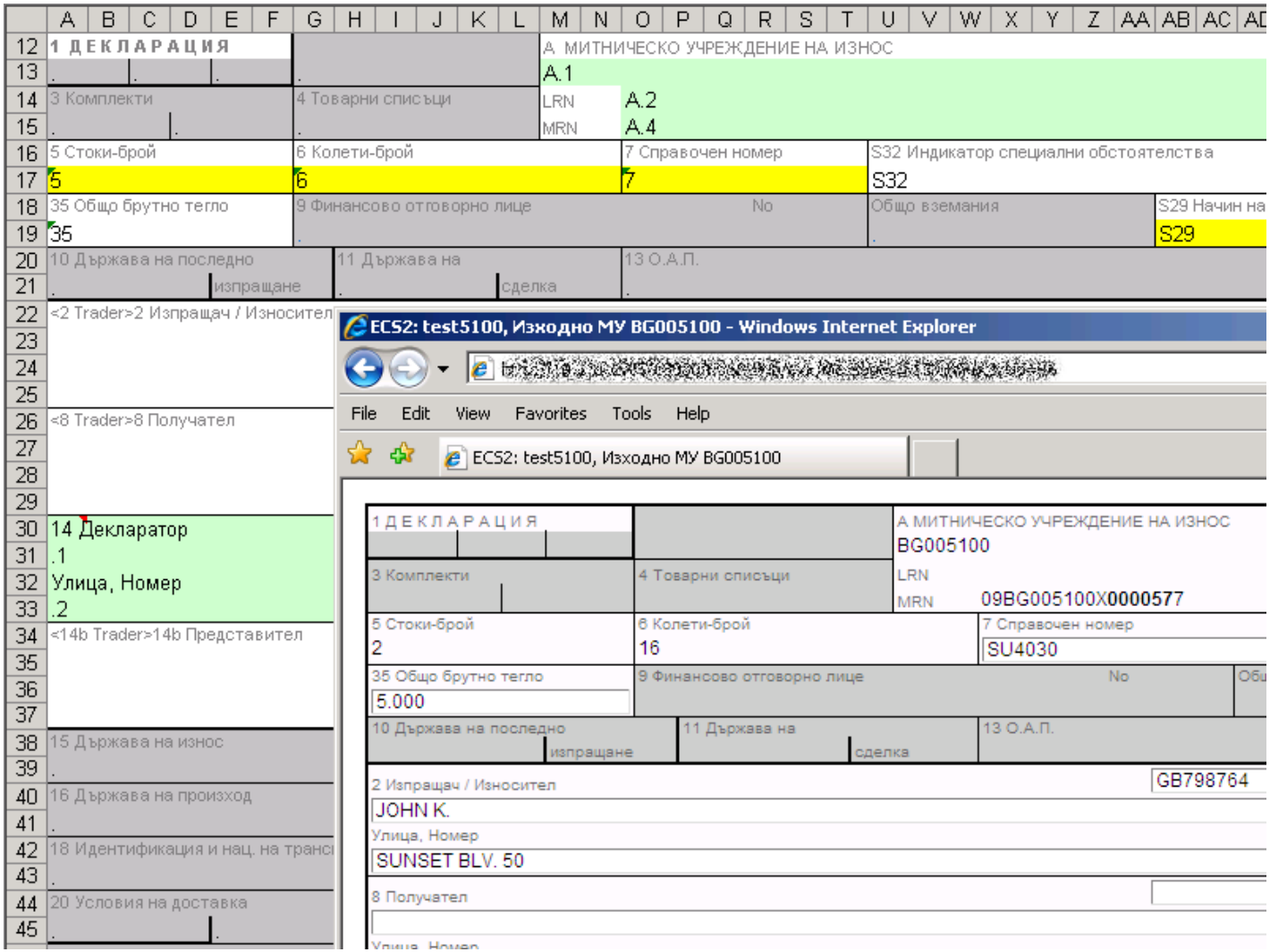


GUI: Results of Control

- The most complicated screen in ECS2
- New values (discovered by the customs officer during control) can be entered below the old values (given in the declaration)
- Color coding is used to highlight deleted values (~~red strikethrough~~), new values (red) and unchanged values (blue/black)



GUI: Generated from Excel



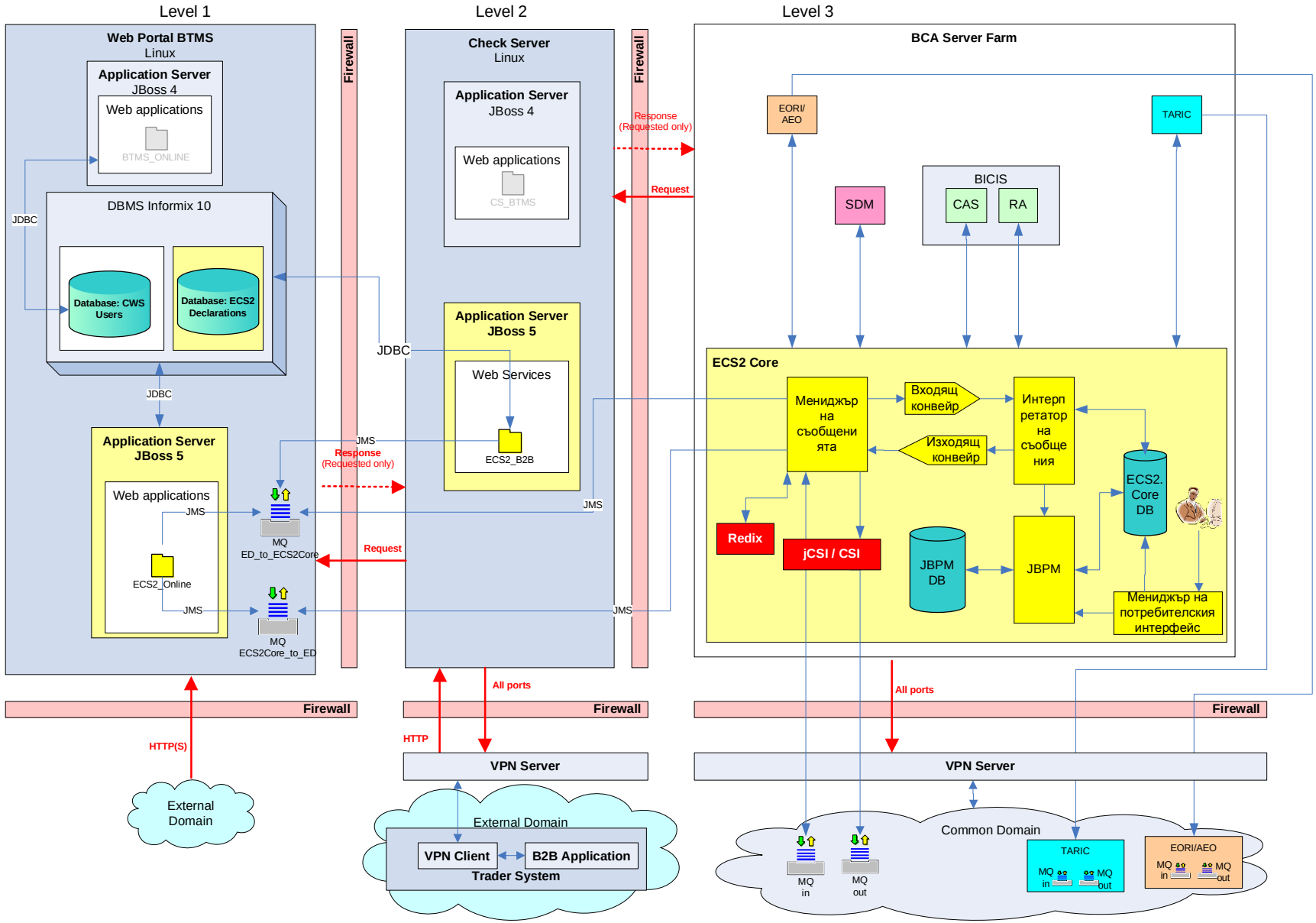
- Best news: the GUI is generated
- Presentation attributes from schema.xls, validations from R&C metamodel, layout from forms.xls, lists of values/links from codelist.xls.
- Very flexible and powerful. The client can edit any visual aspect. *The client loves it.*
- *(Can you guess what's the yellow?)*

GUI: Technical Details

- Conversation scope: while the user works with one Movement, all fetched data is cached in a Unit of work, even if the user switches Office Role. When the user goes away, the object is freed and the cache is released.
- Special JSF components integrating with the R&C framework. Every JSF component uses the R&C metadata about a field or group, which affects type, validators, error status, color, tooltip.
  - R&C marks key fields (those participating in rule xpaths) as *AJAX triggers*: changing such field causes partial submission, R&C recalculation and *partial* refresh
- The generator takes care to put specific IDs on client-side controls, facilitating AJAX and automated testability (JSF puts dynamically generated default IDs that can't be used for such purposes)
- Web optimizations:
  - The GUI is fully zoomable, ensuring optimal use of screen resolution
  - CSS compliance, using a cross-browser CSS framework
  - XHTML doctype tagging, ensuring best browser compatibility mode
  - All resources (CSS, JS, images) are properly marked for caching, ensuring the browser caches them for 1 month and reduces traffic
  - Resource merge: all CSS and all JS files are merged, ensuring faster page processing time.
  - Resource versioning: the build date is added to each resource filename, ensuring the browser will refetch it.
  - HTTP compression: main pages (business content) are gzipped to reduce traffic. The largest page *zips to 10kB*.
- Web traffic optimization was critical to enable centralization of ECS2 Web
  - BICIS is distributed (120 instances across the country) .
  - ECS1 Web is also distributed, while ECS1 Core is centralized, creating weird timeout problems
  - ECS2 is centralized and ensures fast response time, even though some remote Customs Offices have bad connectivity (64-128 kbps)
  - *Client's sysadmin loves it*. We love it too, it simplifies new version deployment immensely.

Deployment Model

- **Platform**: RedHat Enterprise Linux, Informix 10, JBoss, 5.0.1, JEE 6, EJB 3, JSF
- **Centralized** (ECS1 was semi-distributed)
- **Dedicated servers** (ECS1 shares server with 4-5 other applications, sometimes crashed)



Thanks for your time!

- Questions/discussion?
- We think our ECS2 architecture is pretty smart, and would like to hear your comments
  - What would you have done differently?
- We'd love to apply this architecture to similar problems, e.g. import/export/transit/excise systems.
  - Especially in nearby/EU candidate countries (Macedonia, Serbia, Bosnia-Herzegovina, Turkey, Russia)
- Contacts: vladimir@sirma.bg, +359 (888) 568 132, callto://valexiev1