Network of Information and Intelligent Services

Realized from higher level of abstraction with automation

Kalle Launiala
kalle.launiala@citrus.fi
18.10.2011
Motivation

• Person-to-person services constantly being automated to increase productivity
  ▪ Travelling ticket systems as prime example
  ▪ Self-service systems are automated to reduce the cost from person-to-person interaction

• User experience can be better than in person-to-person interaction
  ▪ When the user is familiar with context
Integrating the systems is the next step
- The less manual work is required to perform the tasks between multiple systems, the less errors and more productivity is gained

Seamlessly integrated systems behave as single intelligent network
- Views and interfaces can focus network-wide for all the needed services

Absolute simplicity is the key to absolute productivity
Problem

• Mainstream software development is complex
  ▪ Unpredictable and unreliable even within single closed system
  ▪ Integrating different systems adds even more complexity

• Problem is the human labor aspect of the industry
  ▪ Software development has automated all the other industries, but failed to automate itself
  ▪ Traditional automation solves part of the problem, but makes it more complex to solve the rest
Approach

• Human labor aspect **must not** be ignored in automation
  ▪ Raise of abstraction must not sacrifice any required freedom of custom code

• The human aspect is so strong because it is absolutely needed
  ▪ Automation needs to bend over and around the human aspect, not the other way around
... Approach

• To maximize the adoption it must be made using as familiar elements as possible
  ▪ Mainstream elements allow wide penetration
  ▪ Familiar elements make adoption fluent
  ▪ => XML Schema + mainstream code-generation

• Version controlled and modularized
  ▪ Distribution simplified down to one folder
    ▪ Source controllable within main solution changeset
    ▪ All source control system features apply
  ▪ Modularity enables building blocks on higher level of abstraction
Focusing on the Network

THE CHALLENGE
What Really Matters

• Three main components
  ▪ The user
    ▪ Individual in certain role
    ▪ Legal entity such as organization
  ▪ Information - authorized and audited
    ▪ Controlled validity timed/expiration scope
    ▪ Controlled authorization timed/expiration scope
    ▪ Dependency chain of information achievable
      ▪ Produce only information that is required elsewhere
  ▪ Behavior/Action taken on the information
    ▪ Proactive = Initiated by the active user
    ▪ Automatic = Events launched by the system

• These components are going to be present regardless of the design decisions
Information Security Enforced

• Nobody sees the data without end-user consent
  ▪ Not even the trust providers that validate the trusts
  ▪ Encryption key management is divided with no single master to break the user key

• **End-user chooses who to trust with which data**

• Critically trusted parties refine data for the less trusted
  ▪ Trusted data processing produces less confidential information from absolutely confidential facts
  ▪ Train ticket system needs only ages, head-count and possibly names of students
  ▪ The family members (including their names and ages) is hardly confidential information
Realization

• Modular automation from software design level
  ▪ No requirement for certain platform
  ▪ No requirement for certain framework
  ▪ Support all the platforms and frameworks in parallel

• Modular automation with per-module certification
  ▪ Automation modules can be trusted
  ▪ Software built based on them can be trusted
  ▪ End-user chooses which software to trust

• Anyone can participate – including ones with no trust
  ▪ Anyone can make new modules
  ▪ Not every software needs to be highly trusted
    ▪ As long as the software is only authorized to access common data
    ▪ Sandboxing the automation makes certification process simple and cost-effective
Example: Order Food for Party

- User has profile set to
  - Allow "Neutral Contact Info" for "Delivering Business"
  - Allow Online Payment with confirmation

- "Combine Food Orders"
  - Is independent business combinating service
  - Pay 2% of order total for the service
  - Combines the searches and the order from any joined restaurants
  - Synchronized the deliveries with any joined delivery providers

- User actions
  - Allows service to accept the restaurants with 95%+ delivery rate and with 45 minute guarantee
  - Searches and picks orders based on party members’ requests
  - Confirms order by confirming the bank payment (within the same application, but Bank’s certified area of screen)
Abstractions and Automations (providers separated with color)

GOVERNMENT INFRA:
- Critical trust requirements
- Critical infrastructure
- Free to use

OPEN SOFTWARE ALLIANCE:
- Common agreed structures
- Free to use

RESTAURANT ERP SOFTWARE:
- Restaurant specialized provider
- 1% of sales + hosting 9€/month

TRANSPORT ERP SOFTWARE:
- Transport specialized provider
  - 9€/month / transport vehicle

ORDER COMBINATOR SERVICE + MOBILE APPLICATION:
- Using existing mobile application automation module with infrastructure automation(s)
- Providing integration automation
- Free to use for restaurants and transports
- 2% from customer order
Actual Implementation (each responsible party colored)

ORIGINAL ABSTRACTION/AUTOMATION SOURCE:
- Established mainstream data structure for community to use
- Supported in parallel, no reason at all to be the only exclusive interface

"ORGANIZATION WIDE" STANDARDS AND REQUIREMENTS:
- Decision to support established data structure
- Verification and necessary modification in automation template

PROJECT LEVEL ISOLATION:
- Adjusted to real-world existing system
- Synchronized with organization (both ways) in a controlled fashion

PROJECT LEVEL ISOLATION:
- Adjusted to real-world existing system
- Transparent use of existing mainstream structure
- Modified as little as possible
• Distributed source control (git) allows strict chain of responsibility
  ▪ Any source can be used, as the controlling chain of responsibility is implicit

• The development flow is two-way
  ▪ Project level improvements can be pulled back to the main source or organization’s other projects
  ▪ This is already reality in open-source projects

• Public repositories allow safe reusability
  ▪ Updates to abstractions/automations are easy to control and track
  ▪ Creating trusted and neutral party to certify abstractions (and digitally sign them) is possible
Realization vFinal (in the end)

• Software is built with design level building blocks
  ▪ Horizontals are as wide as possible on common features
  ▪ Verticals are established based on requirements not based on categories
  ▪ End-user or organization can choose and hand-pick its own verticals

• Every vertical can be unique of its kind
  ▪ Sharing the software in common areas
  ▪ Differentiating in software where the vertical differs

• For example single delivery truck driver versus multiple delivery trucks working together
  ▪ Multiple delivery trucks can also swap deliveries and schedule the rendezvous points for the trucks
Vertical Specific Operation: "Accept Delivery Request"

Accept Delivery Request
- RequestID
- FromAddress
- ToAddress
- CratesAmount
- PickupTimeWindow
- DeliveryTimeWindow

Single Delivery Truck Company
- VerifySchedulePossible
  - FromAddress
  - ToAddress
  - PickupTimeWindow
  - DeliveryTimeWindow
- VerifyIfRoom
  - PickupTimeWindow
  - DeliveryTimeWindow
  - CratesAmount
- AcceptDeliveryRequest
  - RequestID

Multiple Delivery Truck Company Or Network of Companies
- VerifyScheduleAndRoomPossibleForAllTransports
  - FromAddress
  - ToAddress
  - CratesAmount
  - PickupTimeWindow
  - DeliveryTimeWindow
- AcceptDeliveryRequest
  - RequestID
- ScheduleDeliveryRequestForAllTransports
  - RequestID
  - FromAddress
  - ToAddress
  - CratesAmount
  - PickupTimeWindow
  - DeliveryTimeWindow

Return Value
- Accepted
• Interface to the network is also modular
  ▪ "Vertical" customizing in end-user applications
  ▪ User Interface in interacting with network is end-user tailorable

• Not everyone wants to manually set all the trusts and interface choices for verticals
  ▪ Public profile libraries for different users
    ▪ Apply for trust authorization and verticals
  ▪ Any third party can share its own profile libraries
    ▪ Communities can step in and contribute
  ▪ Seniors get their own simpler interface
What About Now – How To Get There?

- Start with integration level automation
  - Recognize what we have now for trust, information and data
  - Public sector is already doing the integration actively
  - Automation modules simplify the integration for anyone to use

- Provide component abstraction/automation blocks
  - Applicable from single web-application to governmental level information platforms
  - Once done, reusable by anyone – productivity boosted with automation

- Simply start automating the existing ways of doing from design level
  - This allows backdated update for automation to realize the new features network-wide

- Instant productivity gains in active development
  - The cost is smaller than producing traditional documentation
  - Reusability of automation blocks allow network-adoption once someone/anyone makes the automation block
Critical Security Components Required

REFERENCE OF EXISTING
Strict Security Requirements (if sensitive data is involved)

- Data Encryption Separated
  - Trust Providers issue tokens and establish communication
  - Separate infrastructure for end-user controllable security

- Databases Secured (in cloud or on-site)
  - End-user/organization key controlled data
  - Organizations’ internal integration security satisfied as well

- Physical Security Required
  - Encryption keys are secured
  - Plaintext data (used in applications or during database retrieval) is not if hardware running the application can be tampered
SQL Server Row/Cell Level Encryption

TIP: To ensure good performance, avoid encrypting data using certificates or asymmetric keys.
Custom/NoSQL

- Open source databases provide alternative to SQL Server
  - MariaDB specializing in MySQL
  - Required features can be implemented on open source codebase

- NoSQL
  - Critical data might be better served without SQL at all on core facts
  - Refined queries of nonconfidential (yet authorized) information can be on SQL
Distributed Key Management

- Hardware based solutions exist
  - [http://townsendsecurity.com/](http://townsendsecurity.com/)

- Core requirement is to have sealed and trusted hardware boxes to manage and transform the keys
Dramatically simple and completely open (by design)
Raising the level of abstraction in logically pure manner
Automating any code-block faster than properly documenting it
http://abstractiondev.wordpress.com

FUTURE IS NOW!