

OpenTravel 2.0

Speak – Code - Think



2010 ADVISORY FORUM

Agenda

- Speak OpenTravel
 - Common Vocabulary
- Code OpenTravel
 - Changing Role of Schema in the Organization
- Think OpenTravel
 - Embedding OpenTravel in your business functions.



The Language of Travel

- OpenTravel terminology is increasingly used as a common language for the travel industry.
 - Shared across companies business to business.
- The goal of 2.0 is to increase sharing and the value of sharing.
 - Shared within companies business to technical.



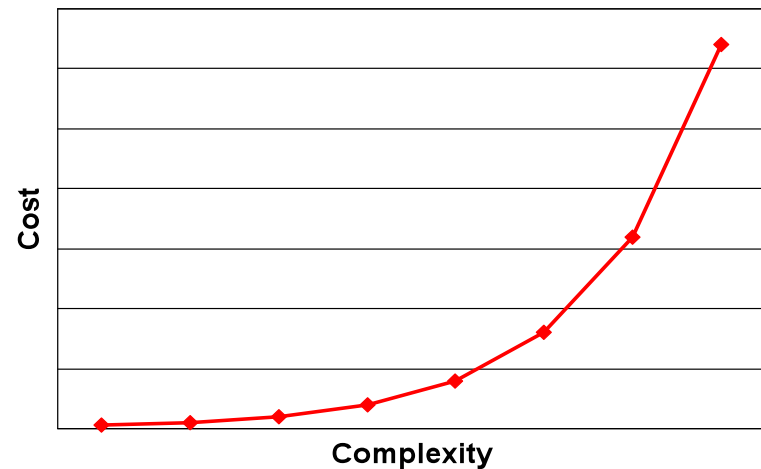
Standard Is Better Than Better

- Companies are aligning themselves with the OpenTravel specifications over proprietary specifications.
 - Increase reusability.
 - Increase marketability
- Even if it is sometimes at the expense of:
 - Features
 - Ongoing support costs*



Cost of Complexity

- After 10 years of pioneering we have an overly complex product.
 - Cleanup is overdue
- Adoption and maintenance costs increase exponentially with complexity.
 - often hidden
- Smaller entities simply lack the resources to deal with the complexity
 - Small and Midsize Enterprises (SMEs)
 - Departments within large organizations.



Code and Think Needs an Easy Button



Better is Better Than Standard

- Updating the standard moves everyone forward.
- The way we adopt XML has been transformed since OpenTravel started in 1999. OpenTravel has not always kept up. Good decisions taken in the early years that fostered adoption and acceptance now inhibit expanded adoption.
 - hardware acceleration
 - code generation (binding) tools
 - WORM (Write Once Read Many) Principle
 - SOA and web services
 - web crawlers
 - UML
 - AJAX

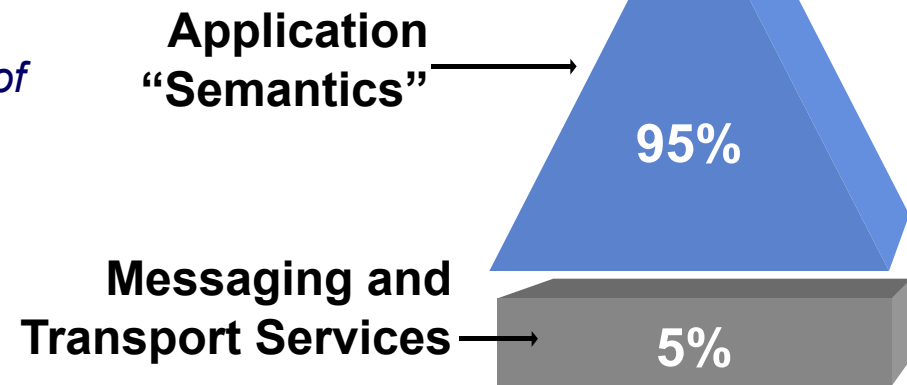


What Problem Do We Want To Solve?

“Interoperability requires the entire interface between applications to be standardized.

- *Only 5% of the interface is a function of the middleware choice.*
- *The remaining 95% is a function of application semantics.”*

Gartner Group



- Interoperability's biggest challenge is directly affected by schemas
 - Used to define service interfaces that accurately describe the service provider's *application semantics*



Schema Styles

- There is no “right answer” to building a “good” schema
 - Schema must strive to reach balance with its intended use
- OpenTravel and IATA had chosen to balance their schemas as
 - Flexible, extensible and permissive
- Implementers need to rebalance the schemas to be
 - precise and consistent
- Descriptive vs. Procedural
 - What it is vs. how it is used
- Prescriptive vs. permissive
 - Who pays to make data right?
- Loose vs. tight
 - How many semantics?
 - Easy to author vs. reuse
- Interchange model
 - Blind or pre-defined partners?
- Extensibility
 - Kept up to date vs. predictability
- Capacity vs. load
 - When to validate and at what cost?



Data Dictionary



- At the heart of "*application semantics*" is an XML schema of common data types that represent a consistent model of the business
 - Reservation, RatePlan, Vehicle, SeatMap etc.
- Without a single model implementing and managing multiple messages across multiple interfaces becomes expensive and slow.
 - multiple incompatible expressions of the same data
 - exception logic in the application layer
 - developers have to 'debug' the schema
- These definitions (like Lego) become the elemental building blocks from which messages are constructed.



Moving Beyond Messages

- We Speak in Messages.
- We Code and Think in Building Blocks.
- Building blocks for future evolution
 - OpenTravel Standards
 - Semantic Web
 - Runtime Discovery
 - Non-OpenTravel Messages and Applications



Moving Beyond Messages

- Create a schema type library (data dictionary)
 - Common business components
 - Airport, Vehicle, SleepingRoom
 - Common 'business agnostic' complex types
 - Address, Telephone
 - Limited number of simple types
 - Strings, Dates
- Reorient OpenTravel specifications around this library.
- Reused by other standards organizations
 - HTNG



Moving Beyond Messages

- *Code* based on the OpenTravel Type Library
 - Build your own messages based on your needs.
 - Local Interfaces e.g. GUI Front-End.
 - Build your own code libraries.
 - Build your own applications.
- *Think* using the OpenTravel Intellectual Property
 - Blend with 'other' standards e.g. Credit card companies.



Schema Type Library

- Limit the following common data-in-motion practices:
 - Smuggling
 - Allowing unintended data to be included
 - Tag Abuse
 - Putting data in mislabeled locations
 - Micro-parsing
 - Tokens that applications are expected to break apart
 - Deferring Responsibility
 - Leaving to the instance data what should be designed into the schema
 - Substitution groups
 - xsi:type
- Define constraints to test against
 - Physical structure
 - Field data types
 - Cardinality



OpenTravel Type Library



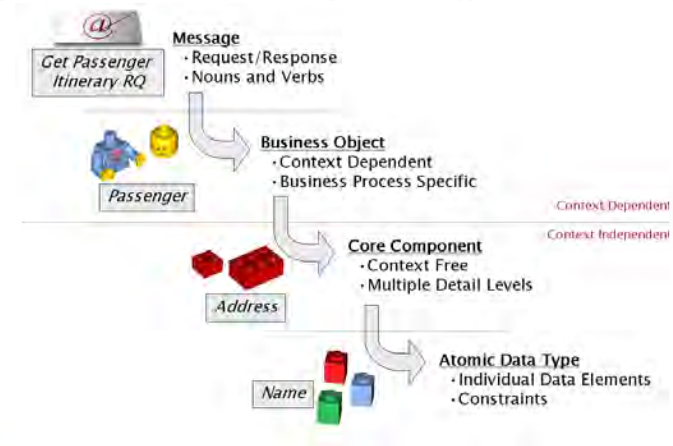
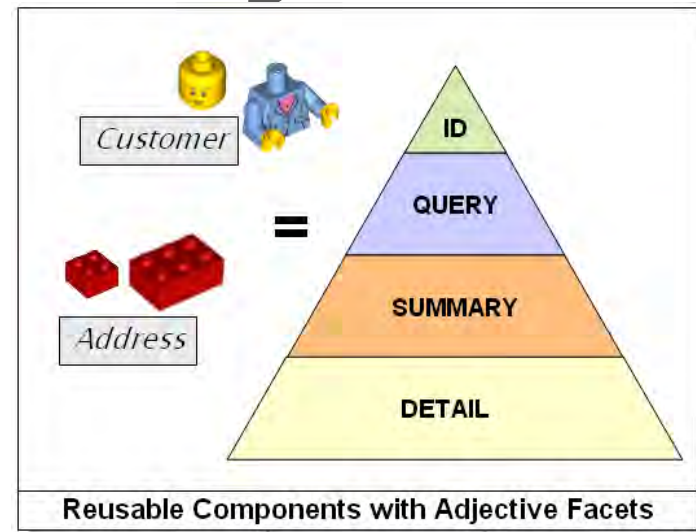
- OpenTravel Type Library
 - Technical Expression of the Language's Dictionary
 - Types - Schema building blocks ("Lego")
 - Library - Development is governed by a central team
 - Use in messages is governed by a project team
- Types reduce cost and complexity of deploying services
 - Allows business specific messages to share core definitions
 - Enables the reuse of code built from the schemas
- Library provides building blocks designed to allow consistent and precise schemas.
- Assumes schemas should change
 - Everything else changes, so why not schemas?



Schema Type Design

- 4 Transactional variants
 - Reference (ID) Type
 - Query Type
 - Summary Type
 - Detailed Type

- 4 Layers of Types
 - Message
 - Business
 - Core Common
 - Atomic (Simple)

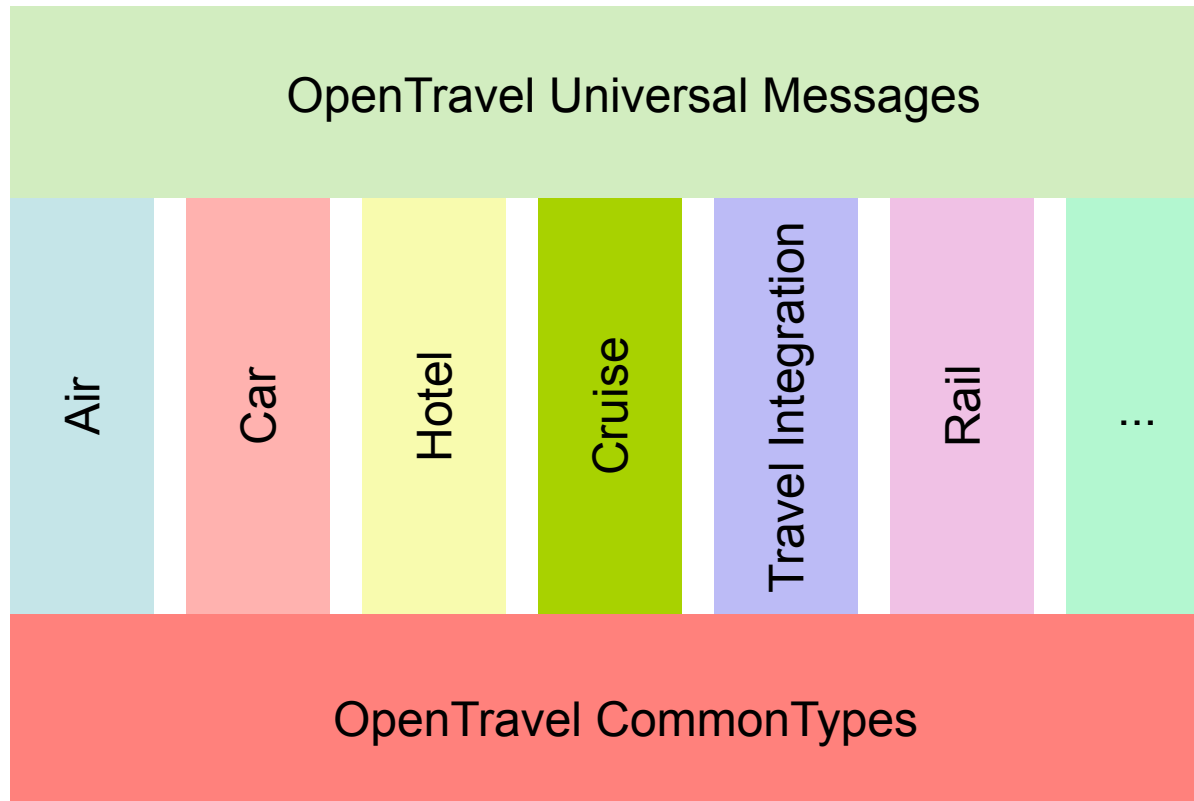


Namespaces

- Conflicting definitions across multiple messages is not W3C compliant.
 - Tools complain when merge multiple messages.
- Reorganize the namespaces vertically along industry groups and horizontally across common schema types and superset messages.
 - Vehicle, Cruise, Hotel, Air etc.
 - Common (replaces current chameleon types)
 - OpenTravel (SuperPNR)
- Isolate groups from each other so changes do not impact one another.
 - Evolve at different speeds



Namespaces



Internationalization

- Use elements, not attributes for text descriptions
 - any text elements should have an `xs:lang` attribute (and be allowed to repeat)
- Locale on Request: language, time-zone, currency
- Common data: multiple address formats.
- Any price/amount should support multiple currencies, currency conversion rates with a flexible number of decimal places.
- Dates and Times: mandate time-zone qualifiers.
- Location elements: (i.e. airports) include time-zone
- Support wider range of tax schemes



Getting Started

- Learn more
 - See the Architecture Team
- Join Architecture
 - Architecture needs you
- Look at your company
- Ask your developers



Conclusion

- Speak OpenTravel
 - Common terminology is the hallmark of successful organizations
 - FASB, PMI, OASIS
- Code OpenTravel
 - schema IS code
 - needs debugging
 - standardized semantic expression of the business
 - reusable, scalable, inexpensive
- Think OpenTravel
 - Internalize OpenTravel for all your interoperability needs not just distribution.
 - If OpenTravel only happens at the boundary of your business everyone loses.



Questions?

- This page intentionally left blank.

