DAML Integrated ONtology Evolution (DIONE) Tools

DAML PI Meeting, 25 May 2004

Dr. Brian Kettler, ISX Corporation
(bkettler@isx.com)

Prof. Jeff Heflin, Lehigh University
(heflin@cse.lehigh.edu)

Agenda

- **Brief Overview**
- **Components – Approach and Plans**
  - ONTADEPT (Kettler)
  - VPI (Heflin)

- **Demonstrations of initial work at the Demo Session**

DIONE Project

- Sponsored by DARPA (DAML)
- Supporting ontology versioning in SATURN and other Semantic Web applications
  - ontologies in Web Ontology Language (OWL)
- Plan to deploy tools to SATURN and via http://www.SemWebCentral.org
  - SATURN users are knowledge engineers authoring/maintaining ontologies and end users of the Portal/Query capability
- Underway in April 2004 (9 month effort)
  - tools to be deployed in CY04
- Contractors: ISX (prime) and Lehigh University

Ontology Versioning/Evolution Problem

- Ontologies are proliferating
  - distributed authors and consumers
- Change is inevitable
  - change in conceptualization of a domain
    - concepts added, changed, etc.
  - *change in specification of a conceptualization*
  - change in representation
    - e.g., SHOE -> DAML -> OWL
- More than just configuration management
- Wish to avoid
  - breaking dependent ontologies, applications, and data
  - rendering existing data inaccessible or incomprehensible
  - extensive effort to translate existing data
- Reduces in worst case to the ontology mapping problem
  - i.e., if change between versions is extensive enough
- Some previous work to leverage
  - database schema versioning (e.g., see [Roddick 1995])
  - ontology versioning/evolution – e.g., Stanford (N.Noy/M.Musen – Protégé, PROMPT, etc.), U. Maryland (J.Heflin/J.Hendler), Vrije University (M.Klein/D.Fensel - OntoView), U. Karlsruhe (Stojanoic/Motik), etc.

Scope

- Not solving the entire versioning problem (still an active research area…)
  - going for "low hanging fruit"
  - need practical tools to handle routine cases soon for SATURN
- Addressing more "routine" changes between versions (vs. total refactoring)
- Not solving the (general) ontology consistency problem
  - other tools for this (e.g., Chimaera, ConsVISor, DL-based reasoners)
- Not doing change detection
  - "ontology diff" – other tools do this (e.g., PROMPTDiff, OntoView, etc)
- Compatibility is really at the level of conceptualization (versus specification)
  - i.e., inside the ontology author's head
  - assume (for the sake of simplification)
    - that a term in Version 2 has the same meaning (i.e., same set of instances intended by the author) as the same term in Version 1
    - that a term in Version 1 maps to some term in Version 2
- Focus initially on "ontology modification scenarios" for the "basic" OWL constructs

Ontology Modification Scenarios (Example)

- Scenarios for ontology versioning
  - mostly above the "atomic" change level
  - focus on "basic" OWL constructs initially
- Example: OMS-1: create a new concept C
  - Steps (not necessarily in order):
    - define new class C
    - locate C within the class hierarchy in O
      - Variant 1: add C as a new leaf class of existing ("native" or imported) classes E1..En
        - may need to add new subclasses between Ei and C
      - Variant 2: add C into the middle of ontology hierarchy
    - add new subclasses to C
    - define unique characteristics of C (optional)
      - add new properties applicable to C
      - modify existing properties
      - restrict properties for C
        - e.g., range, cardinality, etc.
    - add class expressions involving C (optional)
  - E.g, create a new concept Shoe Bomb
    - define class ShoeBomb
    - add into hierarchy with parent classes Shoe and Bomb (or combine Shoe and Bomb using an OWL class intersection)
    - add new property mininumShoeSize for ShoeBomb
    - restrict property maxContainedExplosivesLbs to 1.5
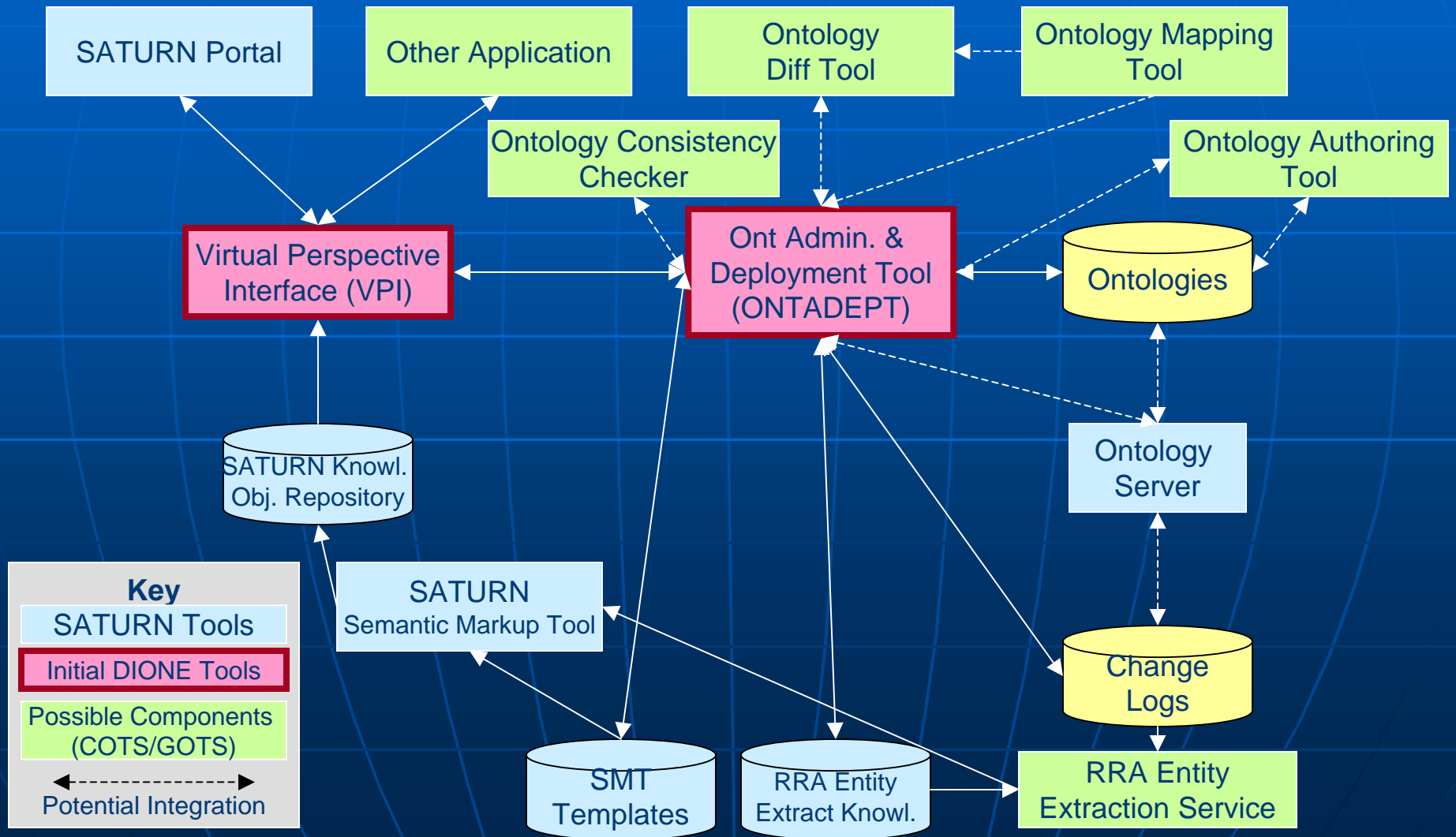- Prioritize scenarios based on frequency, SATURN needs, etc.

DIONE Tools

- **Pre-deployment/Post-deployment support (ISX)**
  - when ontologies change, manage subsequent changes smoothly (e.g., other ontologies, markup templates, etc.)
  - tools used by a knowledge engineer
    - Ontology Admin and Deployment Tools (ONTADEPT)
- **Post-deployment support (Lehigh)**
  - when ontologies change, make sense of instance data
    - focus initially on accessing *old* data with *new* ontologies
    - supports *concurrent* use of old and new ontologies
    - avoids need to translate old data
  - users are people (or client applications) querying a knowledge base of instance data
    - theory of ontology perspectives
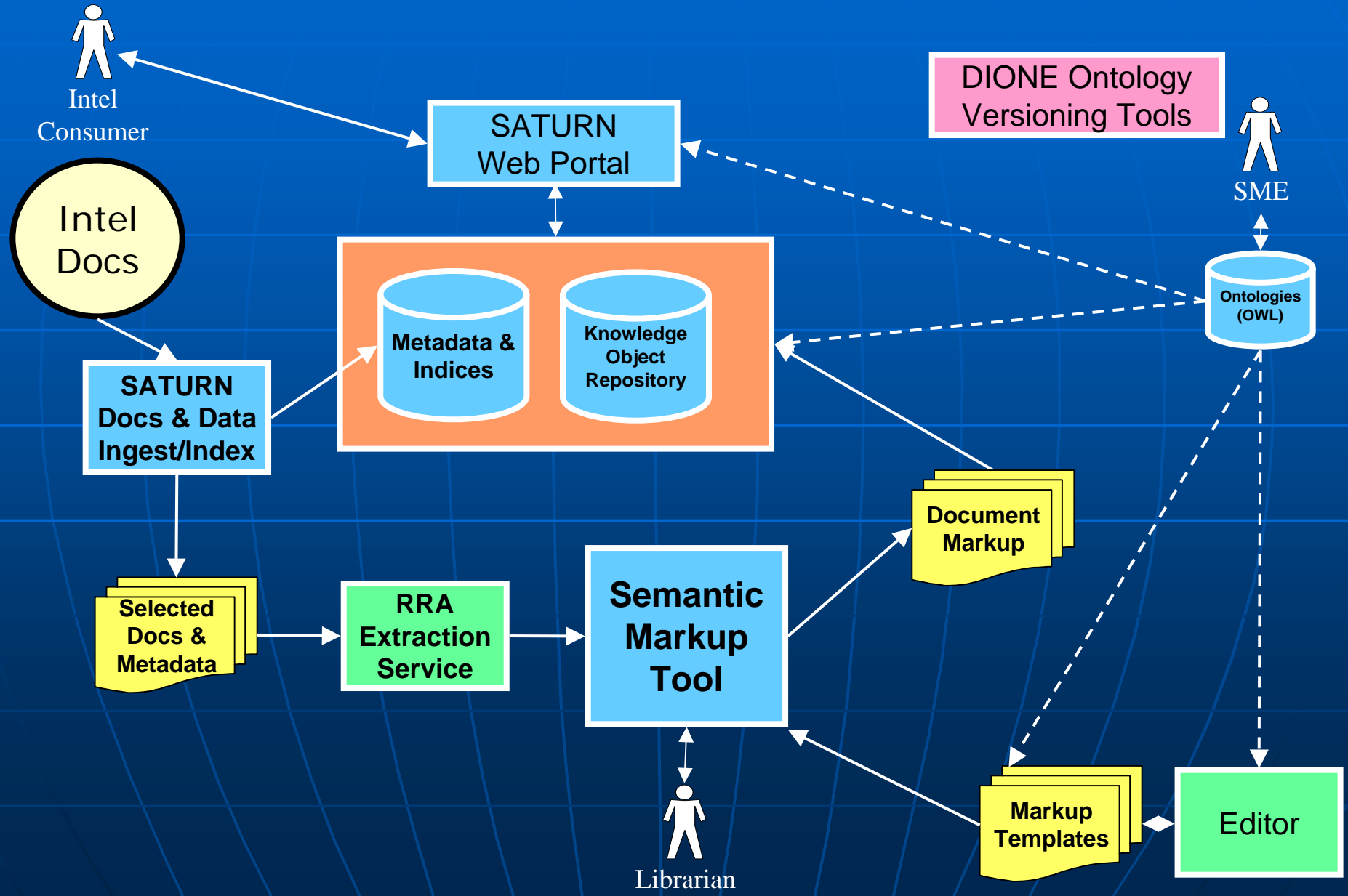   - applied in Virtual Perspective Interface (VPI) tool

System Concept

SATURN Portal

Other Application

Ontology Diff Tool

Ontology Mapping Tool

Ontology Consistency Checker

Ontology Authoring Tool

Virtual Perspective Interface (VPI)

Ont Admin. & Deployment Tool (ONTADEPT)

Ontologies

SATURN Knowl. Obj. Repository

Ontology Server

SATURN Semantic Markup Tool

Change Logs

SMT Templates

RRA Entity Extract Knowl.

RRA Entity Extraction Service

Key
SATURN Tools
Initial DIONE Tools
Possible Components (COTS/GOTS)
Potential Integration

Agenda

- **Brief Overview**
- **Components – Approach and Plans**
  - ONTADEPT (Kettler)
  - VPI (Heflin)

SATURN Components

Intel Consumer

DIONE Ontology Versioning Tools

SME

SATURN Web Portal

Intel Docs

Metadata & Indices

Knowledge Object Repository

Ontologies (OWL)

SATURN Docs & Data Ingest/Index

Document Markup

Selected Docs & Metadata

RRA Extraction Service

Semantic Markup Tool

Librarian

Markup Templates

Editor

Dependency Management

- **Determine Dependencies**
  - parse files (ontologies, templates, etc.)
  - get from user (for complex dependencies)
  - infer
  - using Drexel's OWLJessKB
    - http://edge.cs.drexel.edu/assemblies/software/owljesskb/
    - uses HP's Jena for ont. parsing, Sandia's Java Expert System Shell (Jess) for inference
    - uses CLPS rules based on OWL specification plus others
  - using XML (Java) parser for templates
- **Track Dependencies**
  - store (cache) persistently
- **Provide GUI for Knowledge Engineer to**
  - view/edit dependencies
    - e.g., what depends on X, what does X depend on

Change Management

- Change Detection/Determination
  - obtain from user (as actual or proposed changes), an "ont diff" tool, or an external tool (e.g., ont authoring tool's log)
  - DIONE focus on what to do with changes once they've been determine
- Change Propagation
  - determine subsequent changes
  - rules (in Jess) to propagate
  - some changes suggested by inconsistencies
    - as determined by (external) consistency checker
    - rules may help to pinpoint cause of inconsistency, remedy
- Change Tracking
  - changes required
    - stored as tasks on an agenda
  - changes made
    - stored as entries in a change log (machine-processable, using ontology of changes)
- GUI for Knowledge Engineer
  - view/edit changes – see rationale (dependencies/rules)

Agenda Management

- **Agenda (Task) Tracking**
  - maintain status of tasks (changes)
  - workflow management*
- **Agenda (Task) Execution***
  - provide automation to assist with task execution
    - e.g., launch appropriate editor to make a change, replace strings in a file, etc.
    - interface with other editors (e.g., ontology authoring tool)
- **GUI for a Knowledge Engineer**
  - view/edit task (task status, etc.)
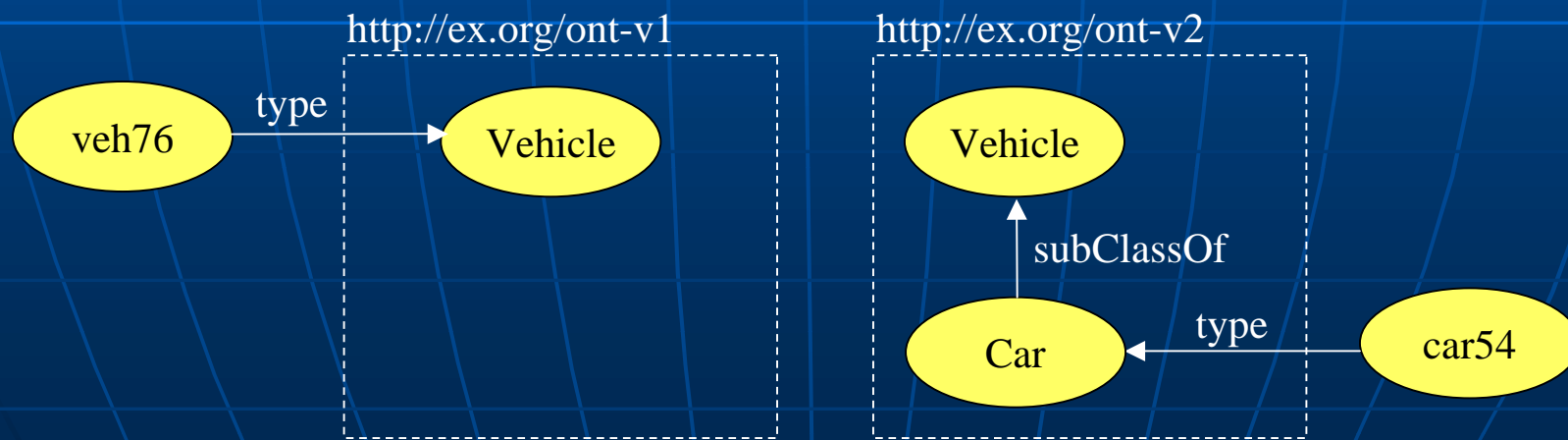
*not a DIONE Phase 1 focus (may be done under SATURN)

Agenda

- **Brief Overview**
- **Components – Approach and Plans**
  - ONTADEPT (Kettler)
  - VPI (Heflin)

Post-deployment Versioning

- Each new version has new URL
  - other users may have committed to your ontology
    - "point at" it using its URL
  - if you change the file at that location, then you change their commitment without their consent
- But what happens to data that committed to the older ontology?

http://ex.org/ont-v1

veh76 —type→ Vehicle

http://ex.org/ont-v2

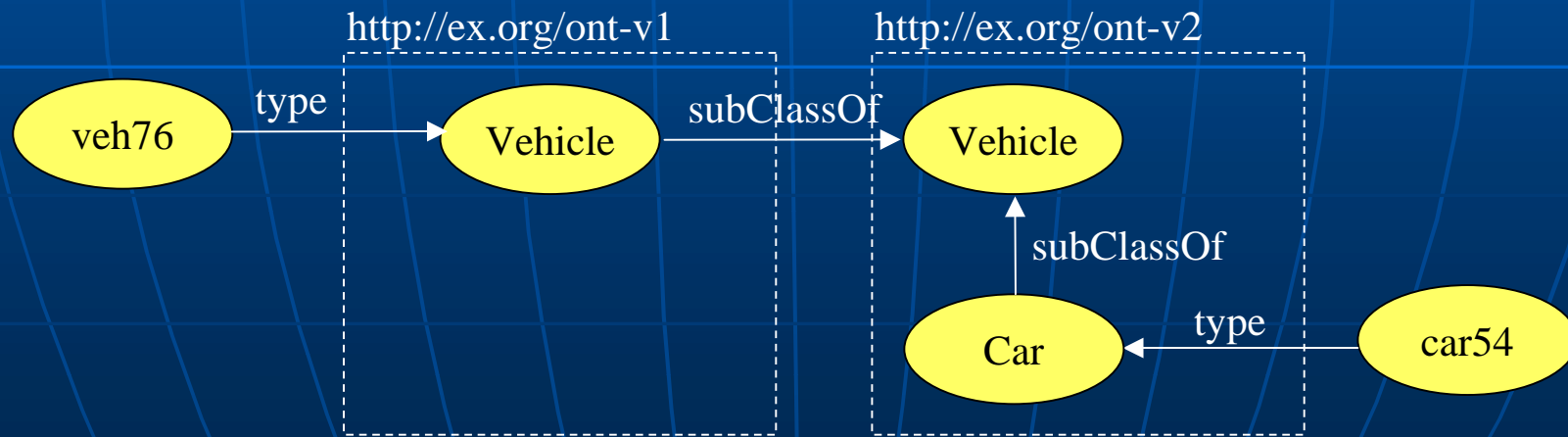Vehicle
↑ subClassOf
Car ←type— car54

Prospective Use of Data

- View old data through a new ontology
  - we want veh76 to be a v2:Vehicle
  - however, different versions have different namespaces
  - thus classes are distinct
- Solution?
  - relate classes/properties using subClassOf/subPropertyOf
  - all instances of v1:Vehicle are now instances of v2:Vehicle

Versioning in OWL

- **priorVersion**
  - indicates a previous version of an ontology
- **backwardCompatibleWith**
  - indicates a version with which ontology is backward compatible
- **DeprecatedClass**
  - used to signify that a class should no longer be used
- **DeprecatedProperty**
  - used to signify that a property should no longer be used
- **versionInfo**
  - used for CVS-like strings
- **incompatibleWith**
  - opposite of backwardCompatible with

OWL Versioning Syntax

```
<rdf:rdf xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<owl:Ontology rdf:about="">
    <owl:priorVersion rdf:resource="http://ex.org/schema-v1">
    <owl:backwardCompatibleWith rdf:resource="http://ex.org/schema-v1">
</owl:Ontology>
<owl:DeprecatedClass rdf:ID="Megalodon">

<owl:Class rdf:ID="Dolphin">
    <rdfs:subClassOf rdf:resource="#Mammal">
</owl:Class>
</rdf:rdf>

…
```
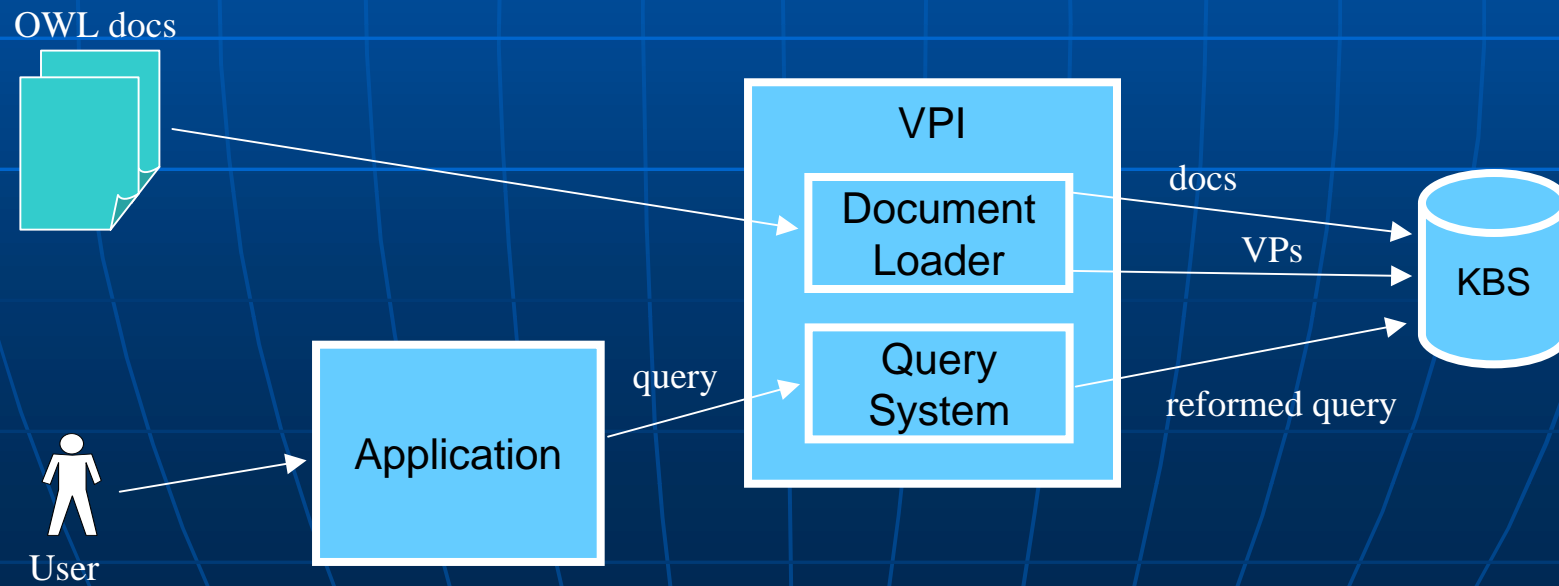
Semantics for Versioning

- OWL has little to no semantics for versioning properties
- Ontology Perspective Theory
  - originally developed for SHOE
  - allows multiple views of the same data
    - each view based on an ontology
  - an initial semantics for versioned ontologies
    - in particular, semantics for backward-compatible-with
  - prospective use of selected data
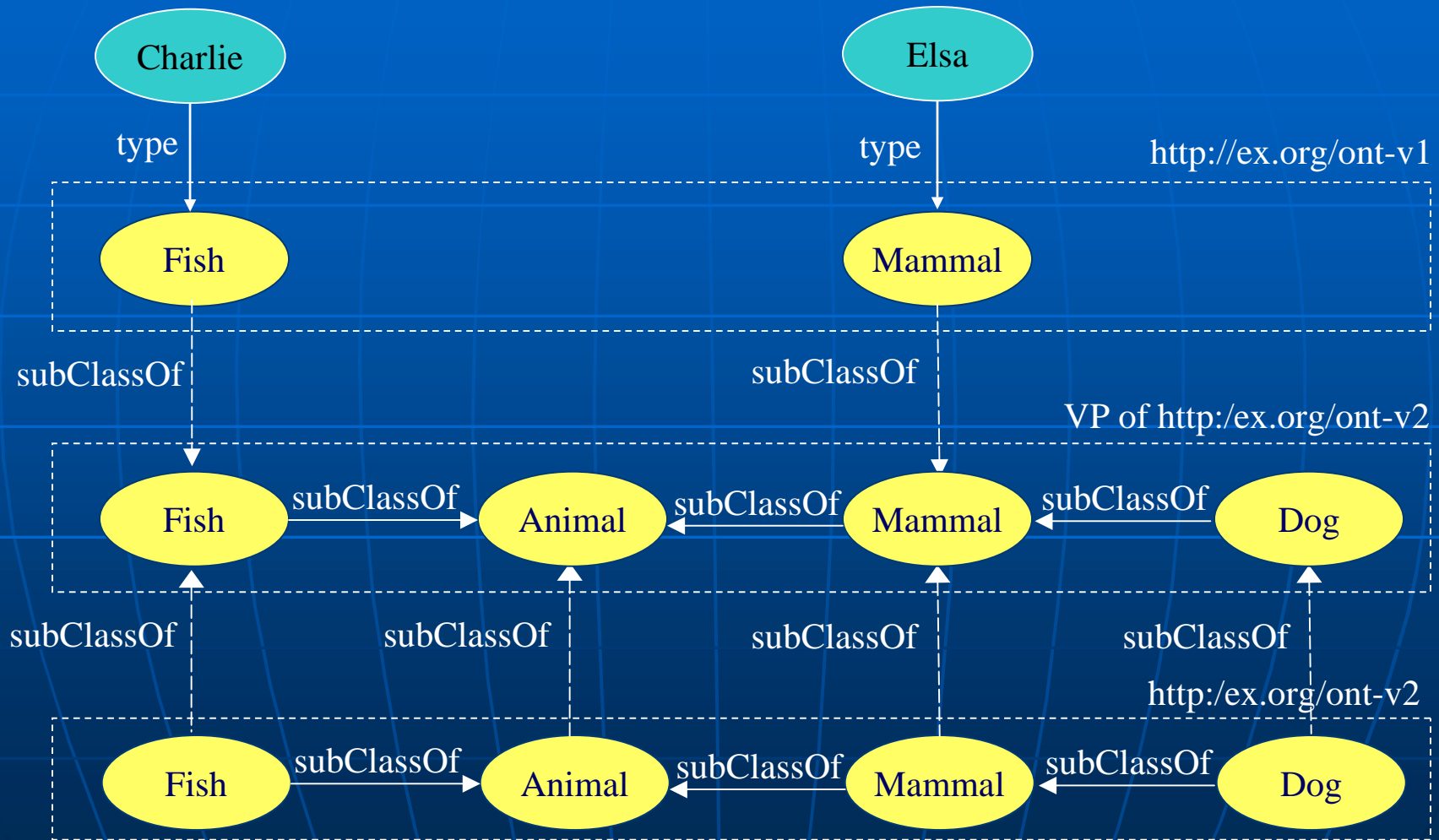
Virtual Perspective Interface

- Middle layer between user and any OWL KB system
- Allows KB to perform perspective-based entailment without modifying internals
- Works by creating a virtual OWL ontology that results in perspective entailments using only OWL entailment

Simple VPI Example

Challenge

What if change is to correct an error?

http:/ex2.org/myfile

Flipper

Flipper becomes both v2:Fish and v2:Mammal!

type

http://ex.org/ont-v1

Fish ← subClassOf ← Dolphin   Mammal

subClassOf   subClassOf   subClassOf

VP of http:/ex.org/ont-v2

Fish   Dolphin → subClassOf → Mammal
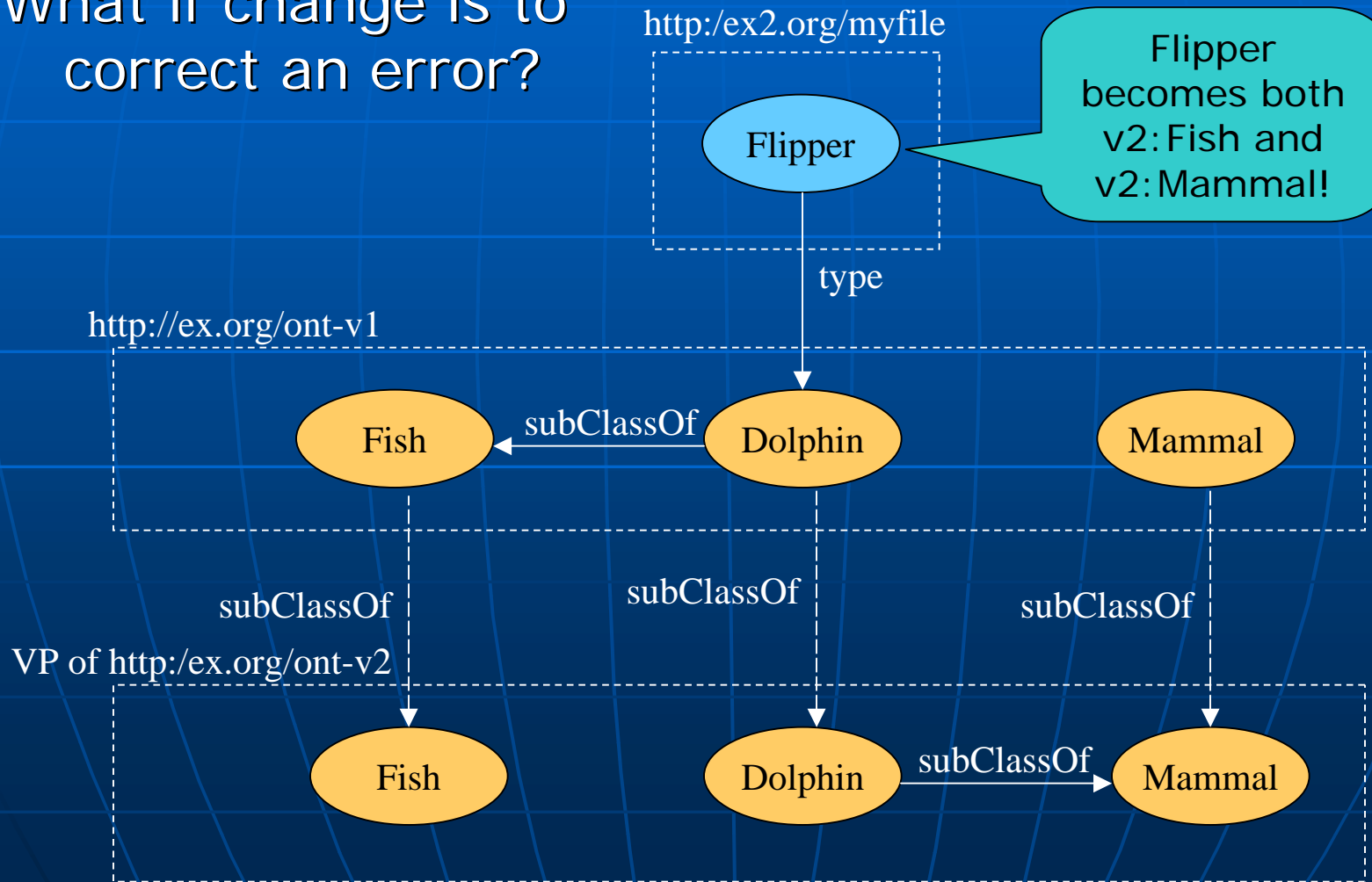
Questions?

- Brian Kettler, bkettler@isx.com
- Jeff Heflin, heflin@cse.lehigh.edu

Detail/Backup Slides

SATURN Requirements
(Year 1 and Beyond)

- Determine dependencies
  - within an ontology (consistency)
  - between ontology and
    - another ontology markup template
    - entity extraction rules/patterns
    - applications/components
    - instance data in a KB represented using the ontology
- Evaluate changes (actual or proposed)
  - to an ontology, template, etc.
  - determine changes (detect, represent, etc.)
  - assess impact (change propagation)
  - log changes (in a machine-processable way)
- Track agenda of tasks (changes to make)
  - track task status
  - assist in task (change) execution: e.g., launch editors, replace strings, etc.
  - interface with CM system for version file management, rollback, etc.
- GUI for above for a knowledge engineer
  - view/edit dependencies, changes, and tasks
- Interface with ontology/template authoring tools/IDE
  - e.g., Protégé for ontologies, XML Spy for markup templates (XML), etc
- Above should scale well and support eventual distributed ontology authorship

Other Ontology Modification Scenarios (1)

- **OMS-1: create a new concept C**
- **OMS-2: Add new property P**
- **Refine an existing concept C**
  - **OMS-3a: add "slot" (property/constraint) to C [explicative change]**
    - TBD
  - **OMS-3b: Divide C into subclasses S1...Sn**
- **Refine an existing property P**
  - **OMS-4a: Divide P into subprops S1...Sn**
    - e.g., siblingOf -> brotherOf and sisterOf (both subprops of siblingOf)
- **OMS-5: delete class C — hard (handle later) [might be same as merge with parent class]**
- **OMS-6: delete prop P — hard (handle later) [might be same as merge with parent class]**
- **OMS-7: create a parent class C for subclasses C1 and C2**
  - e.g., InanimateThing is new parent class for Vegetable and Mineral
- **OMS-8: create a parent prop P for subprops P1 and P2**
  - e.g., siblingOf is new parent prop for brotherOf and sisterOf

Need to select & prioritize for DIONE (e.g., based on SATURN needs, etc.)

Other Ontology Modification Scenarios (2)

- **OMS-9: merge classes C1 and C2 into C**
  - e.g., FixedWingAircraft & RotaryWingAircraft -> Aircraft
- **OMS-10: merge props P1 and P2 into P (delete P1 and P2)**
  - e.g., lengthOf & widthOf -> areaOf (delete lengthOf & widthOf)
- **OMS-11: split class C into C1 and C2 (delete C?) — hard (handle later)**
  - e.g., Aircraft -> FixedWingAircraft & RotaryWingAircraft (delete Aircraft)
- **OMS-12: split prop P into P1 and P2 (delete P?) — hard (handle later)**
  - e.g., areaOf -> lengthOf and widthOf (delete areaOf)
- **OMS-13: rename class C**
- **OMS-14: rename prop P**
- **OMS-15: replace class C with property P (i.e., attribute becomes a class)**
  - e.g., Class MotorizedVehicle -> Property propulsionMethod = Motorized
  - e.g., Class RedThing -> Property color = Red
- **OMS-16: replace property P with class C (i.e., class becomes an attribute)**
  - e.g., Property propulsionMethod = Motorized -> Class MotorizedVehicle
    - add additional props?
  - e.g., Property color = Red -> Class RedThing
    - OWL: use class expression since the property is definitional?
- **OMS-17: change a property's characteristics (e.g., its type, etc.) [BREAKDOWN]**
  - e.g., Datatype prop to Object Type prop
    - e.g., XSD date/time string to DAML Spatial date/time object
  - e.g., domain/range changes
  - e.g., transitive -> nontransitive (problem if inferences are cached)
    - make low priority since not used/supported in SATURN now?

Need to select & prioritize for DIONE (e.g., based on SATURN needs, etc.)

Other Ontology Modification Scenarios (3)

- **OMS-18a: add annotation property (of a class, property, individual, or ontology)**
  - e.g., labels?
- **OMS-18b: delete annotation property (of a class, property, individual, or ontology)**
- **OMS-18c: change annotation property (of a class, property, individual, or ontology)**
- **OMS-18d: change ontology name (URI)**
- **OMS-19: change enumerated values for a Class**
- **OMS-20a: add an Individual (to the ontology)**
- **OMS-20b: delete an Individual (from the ontology)**
- **OMS-20c: change an Individual (in the ontology)**
- **OMS-21: make a class C also an individual I (OWL Full only) – also vs. instead**
- **OMS-22: make an individual I also a class C (OWL Full only) – also vs. instead**
- **23/24 Deprecate Class/Prop**
- **OMS-xx: change Property units: e.g., maxRange: miles -> km**
- replace class C1 with class C2, where C2 now subsumes C1 – covered in above by Refine Class?
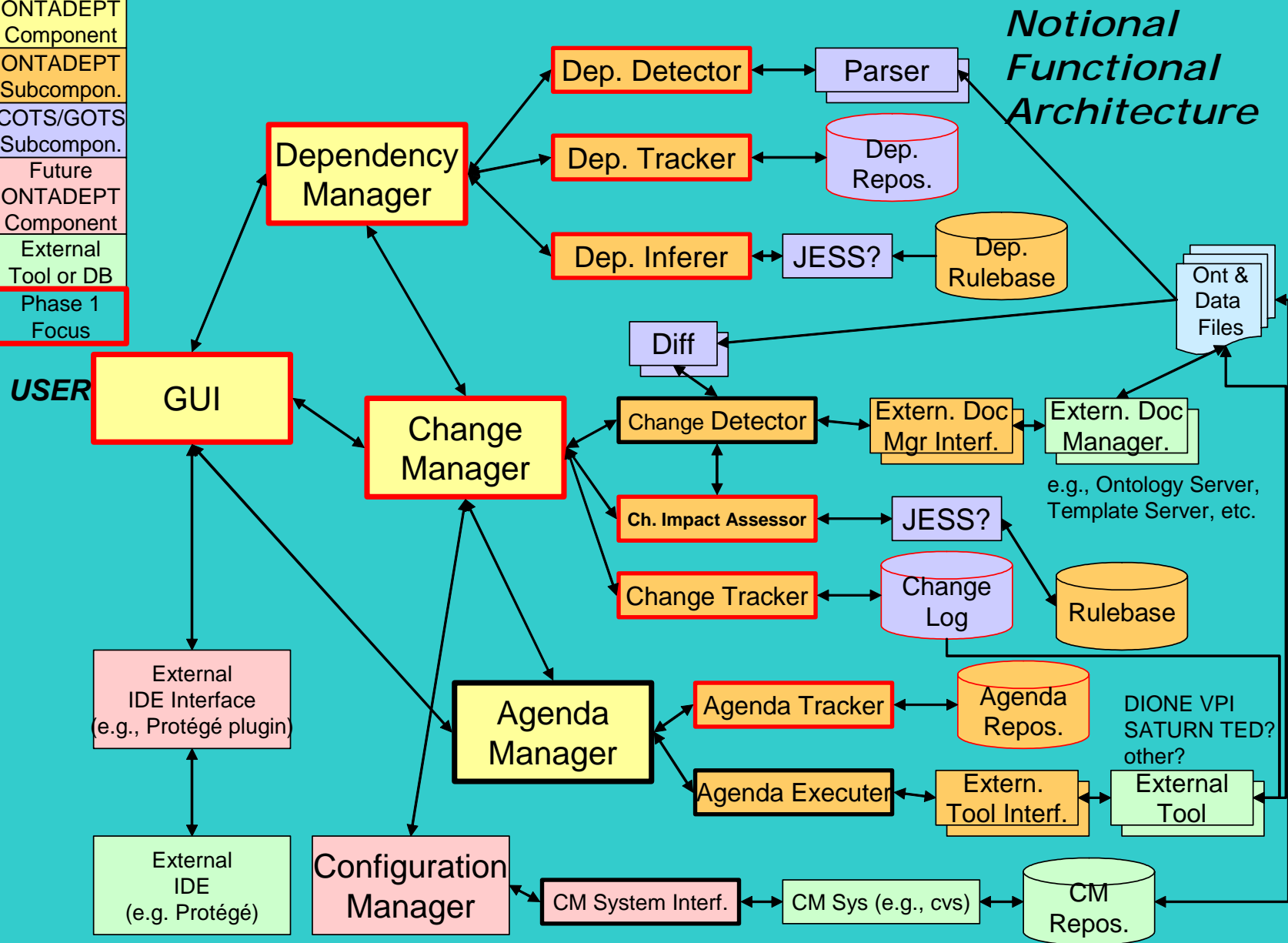- replace prop P1 with prop P2, where P2 now subsumes P1 – covered in above by Refine Class?

Need to select & prioritize for DIONE (e.g., based on SATURN needs, etc.)
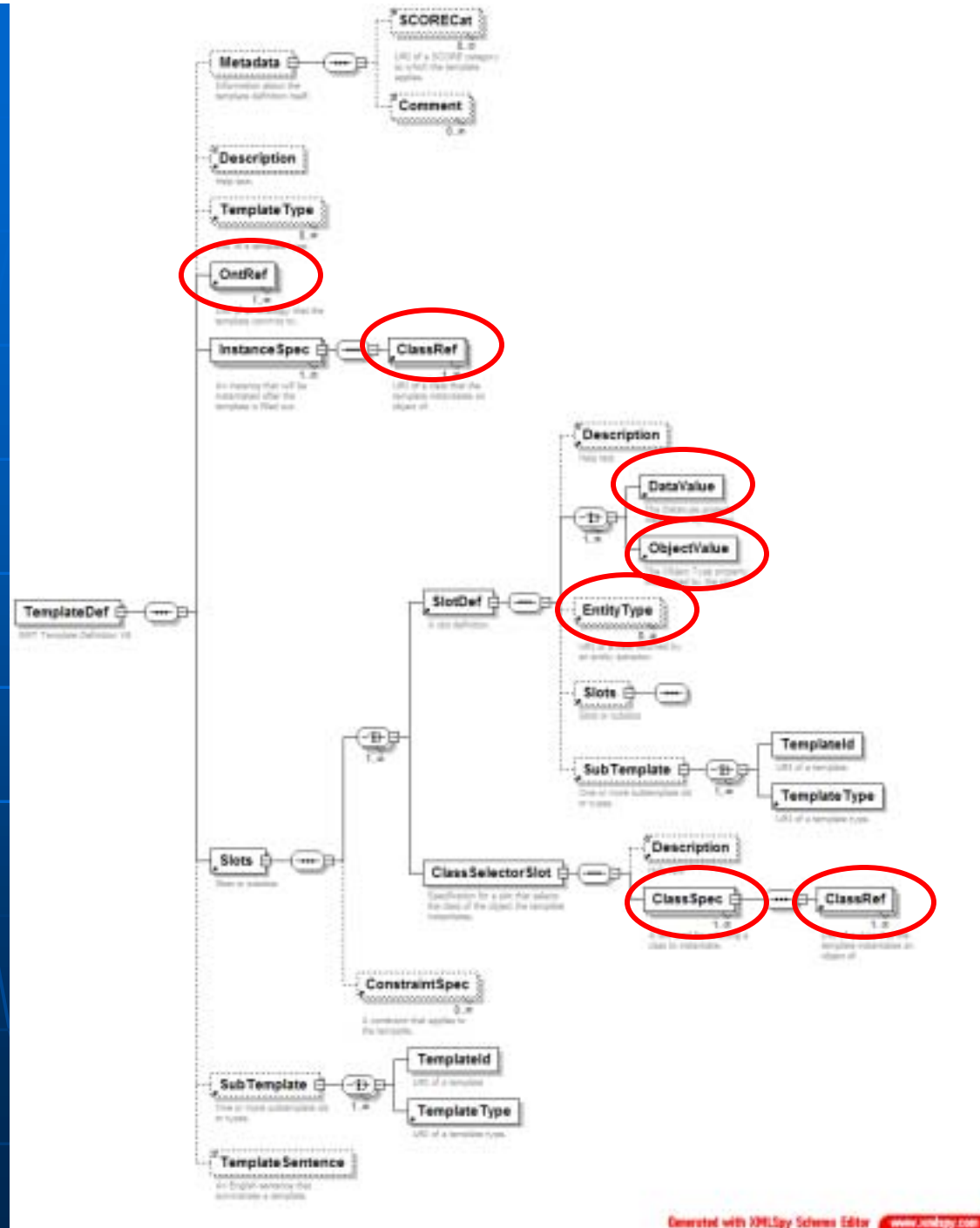
ONTADEPT Functional Architecture

Notional Functional Architecture

Legend:

ONTADEPT Component
ONTADEPT Subcompon.
COTS/GOTS Subcompon.
Future ONTADEPT Component
External Tool or DB
Phase 1 Focus

USER

- Dep. Detector → Parser
- Dep. Tracker → Dep. Repos.
- Dep. Inferer ↔ JESS? ← Dep. Rulebase
- Dependency Manager
- GUI
- Change Manager
- Diff
- Change Detector ↔ Extern. Doc Mgr Interf. → Extern. Doc Manager.
- e.g., Ontology Server, Template Server, etc.
- Ch. Impact Assessor ↔ JESS?
- Change Tracker ↔ Change Log → Rulebase
- Agenda Manager
- Agenda Tracker → Agenda Repos.
- Agenda Executer ↔ Extern. Tool Interf. ↔ External Tool
- DIONE VPI SATURN TED? other?
- Ont & Data Files
- External IDE Interface (e.g., Protégé plugin)
- External IDE (e.g. Protégé)
- Configuration Manager ↔ CM System Interf. ↔ CM Sys (e.g., cvs) ↔ CM Repos.

SATURN Markup Template Representation: XML Schema smt.xsd

Ontology Reference

```xml
<TemplateDef id="SatPersonAffil" label="Person Affilitation" defSchema="&smtxsd;"
    defSchemaVersion="8.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="C:\BPK\Work\Proj\SATURN\Dev\Ontologies\Templates\smt.xs
    d">
    <Metadata author="Brian Kettler" organization="ISX Corp" versionDate="2004-03-25"
    versionId="1.8">
            <SCORECat uri="&score;foo"/>
            <SCORECat uri="&score;bar"/>
            <Comment>Subtemplate for various other templates (e.g., MeetingEvent)</Comment>
    </Metadata>
    <Description>This template describes a person's affilitations</Description>
    <TemplateType uri="&smt;Person"/>
    <OntRef uri="&sont;"/>
    <InstanceSpec id="Person1" isMainInstance="true">
            <ClassRef uri="&sont;Person1"/>
    </InstanceSpec>
    <InstanceSpec id="Org1" isMainInstance="false">
            <ClassRef uri="&sont;Organization"/>
    </InstanceSpec>
    <InstanceSpec id="Country1" isMainInstance="false">
            <ClassRef uri="&sont;Country"/>
    </InstanceSpec>
    <Slots>
            <SlotDef id="PersTitle" label="Title or Rank" datatypeURI="&xsd;string"
    minOccurs="0">
                    <Description>Person's title or rank</Description>
                    <DataValue subjectId="Person1" propURI="&sont;hasTitle"/>
            </SlotDef>
            <SlotDef id="PersOrg" label="Organization Affiliation"
    datatypeURI="&xsd;string" minOccurs="0">
                    <Description>Organization person is affilitated with</Description>
                    <ObjectValue subjectId="Person1" propURI="&sont;memberOf"
    objectId="Org1" isNewInstance="true"/>
            </SlotDef>
```
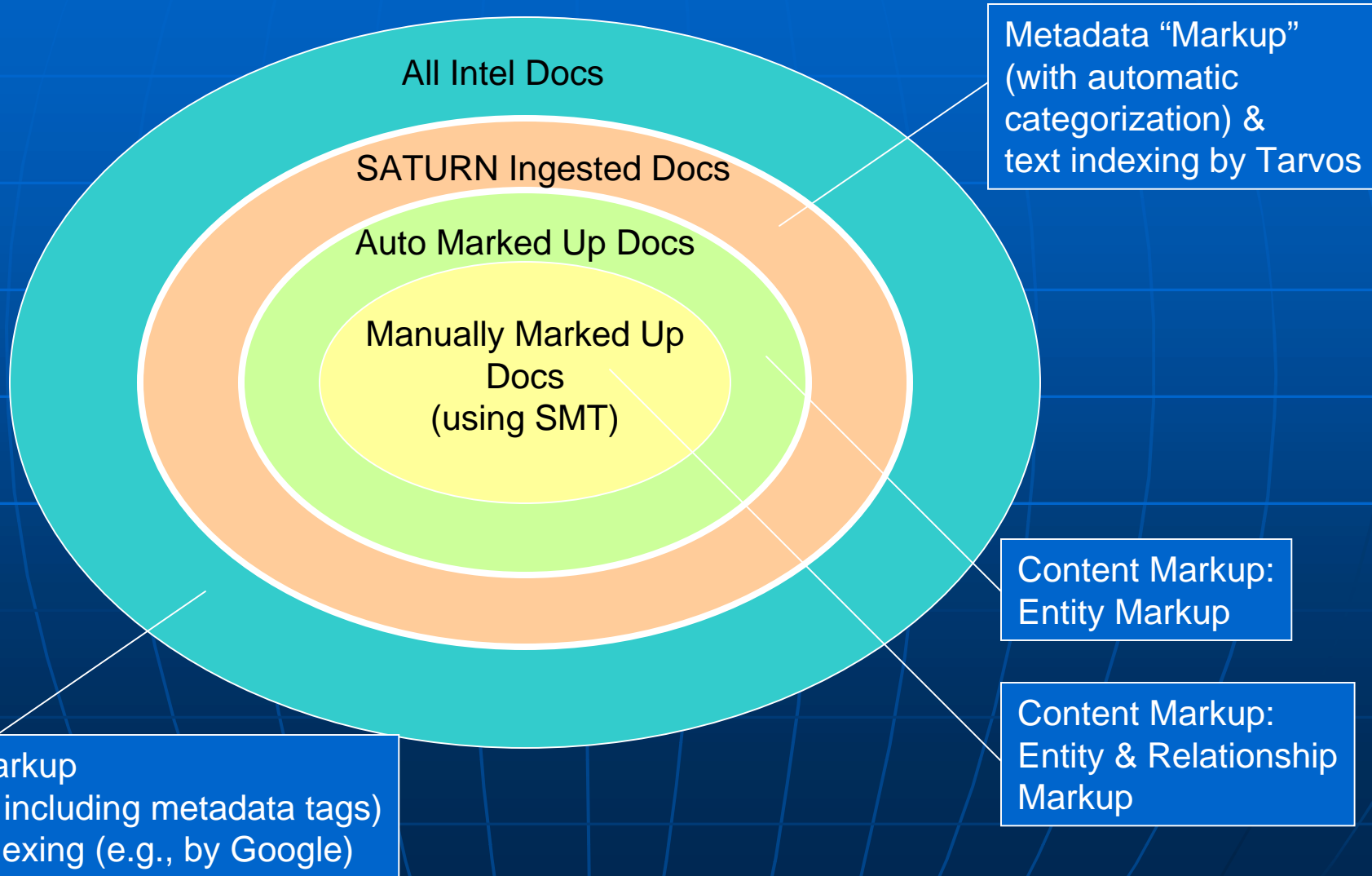
Kinds of Document Markup in SATURN

All Intel Docs

SATURN Ingested Docs

Auto Marked Up Docs

Manually Marked Up Docs
(using SMT)

Metadata "Markup"
(with automatic categorization) &
text indexing by Tarvos

Content Markup:
Entity Markup

Content Markup:
Entity & Relationship
Markup

HTML markup
(possibly including metadata tags)
& text indexing (e.g., by Google)

VPI Problem Definition

- Reduce ontology perspective entailment to OWL entailment

- Define functions T_{per} and T_{query} such that
  - KB $\models_O \alpha$ iff KB $\cup$ $T_{per}(O)$ $\models_{OWL}$ $T_{query}(\alpha)$
    - $\models_O$ means entailment using O as the ontology perspective
    - $\models_{OWL}$ means OWL-DL entailment

- Additional constraint
  - want $T_{per}(P)$ to produce a small number of axioms
    - scalability for large KBs

Complex VPI Example