BridgePoint Roadmap Possibilities



October 2018

BridgePoint Roadmap - Background

- Need for excellent xtUML tools…forever
- Balancing
  - Needs of existing projects
  - Community growth
  - Technology convergence
- OneFact Vision
  - Balancing demands
  - Leveraging technological insight
  - Providing a starting point for discussion
  - Seeking input
  - Driven by contracts
  - **Incremental, always improving**

BridgePoint Roadmap (last 12 months)

- 6.8 (November 2017)
  - Enhanced OAL Editor (phase 1)
    - OAL Autocomplete
  - UI/UX improvements
  - Started the move to Python-based generator
  - Various defect repairs
- 6.10 (February 2018)
  - Project Rapping (MASL improvements)
    - Introduced new MASL reference models
    - Many MASL enhancements and fixes
  - Several OAL content assist improvements
  - Various defect repairs



BridgePoint Roadmap (last 12 months cont'd)

- 6.12 (March 2018)
  - Intrduced MASL coverage tool
  - More MASL reference models
  - Introduced set operations in OAL
  - Various defect repairs and enhancements



- 6.14 (June 2018)
  - Introduced GPS Watch MASL Reference Model
  - Saab project Sortie-1
    - Search of xtUML structural elements
    - Improve datatype scoping allow name reuse
    - Other tool and UI/UX improvements
  - Update MC3020 to pyrsl2
  - Improved element ordering consistency between views
  - Various defect repairs and enhancements

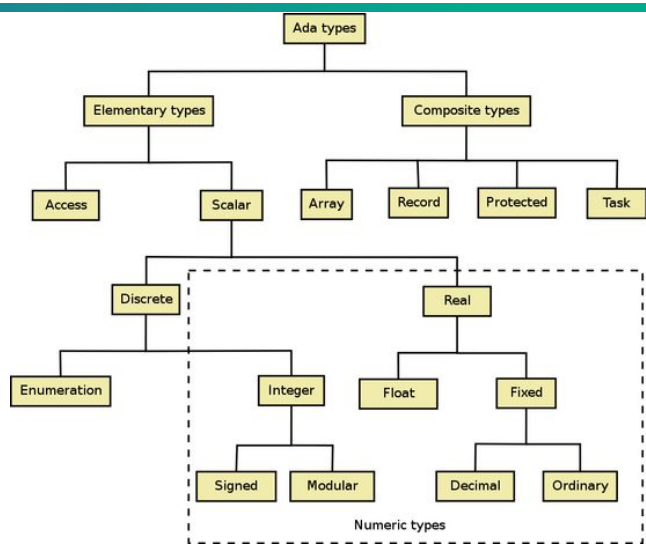
Some Possible BridgePoint Roadmap Additions

- Advanced type system (Ada, Darwen and Date)
- More User Experience / User Interface improvements
- ALS Dialect Support
- OAL automatic reference maintenance
- Package references, Phase 2
- Persistence layer update
- Load Improvements (Proxy removal?)
- Compare and merge
- Activity signature editor
- Multi-body activity editor
- Verifier for MASL
- Inspector integration
- Marking solution
- Pre-existing instance solution



Advanced Type System

- Based on: Ada, Darwen and Date
- Easy, natural type conversions
- Collection types with various properties
  - Ordered/unordered
  - Unique elements/repeat elements
  - Defined bounds/boundless
- Flexible user defined type hierarchies
  - Add constraints on existing types
  - Convert between user defined types in a hierarchy
- Type references and anonymous types
  - Facilitate better type sharing
  - Enable textual persistence of signatures



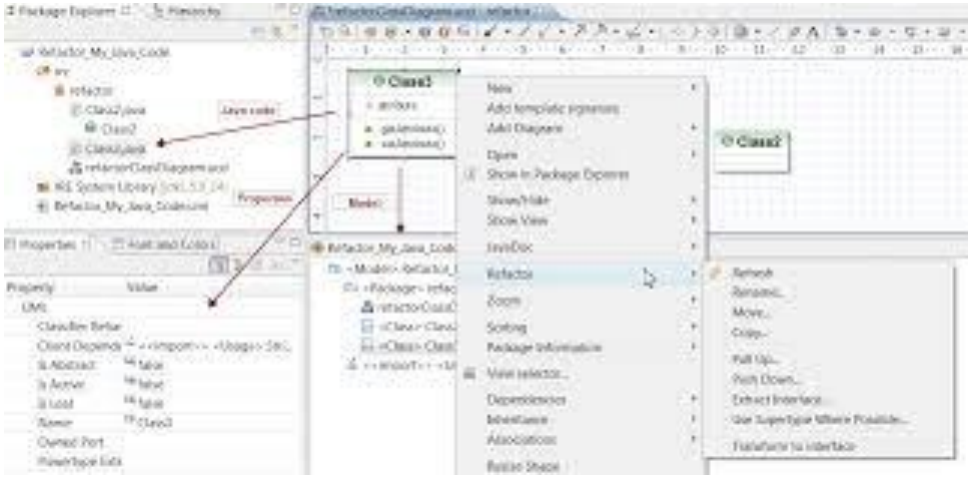
User Experience / User Interface Improvements

- More filtering
  - "Properties View is massively painful!" (Tower)
  - Palette
- Task-specific workflows
  - Less clicks, less options for entry
- In-place editing
  - Ability to edit in the canvas
- Consistency across available views
- Enhanced Model Explorer
  - Tabbed View
- Measured Improvement
  - Scorecard?

ASL Dialect Support

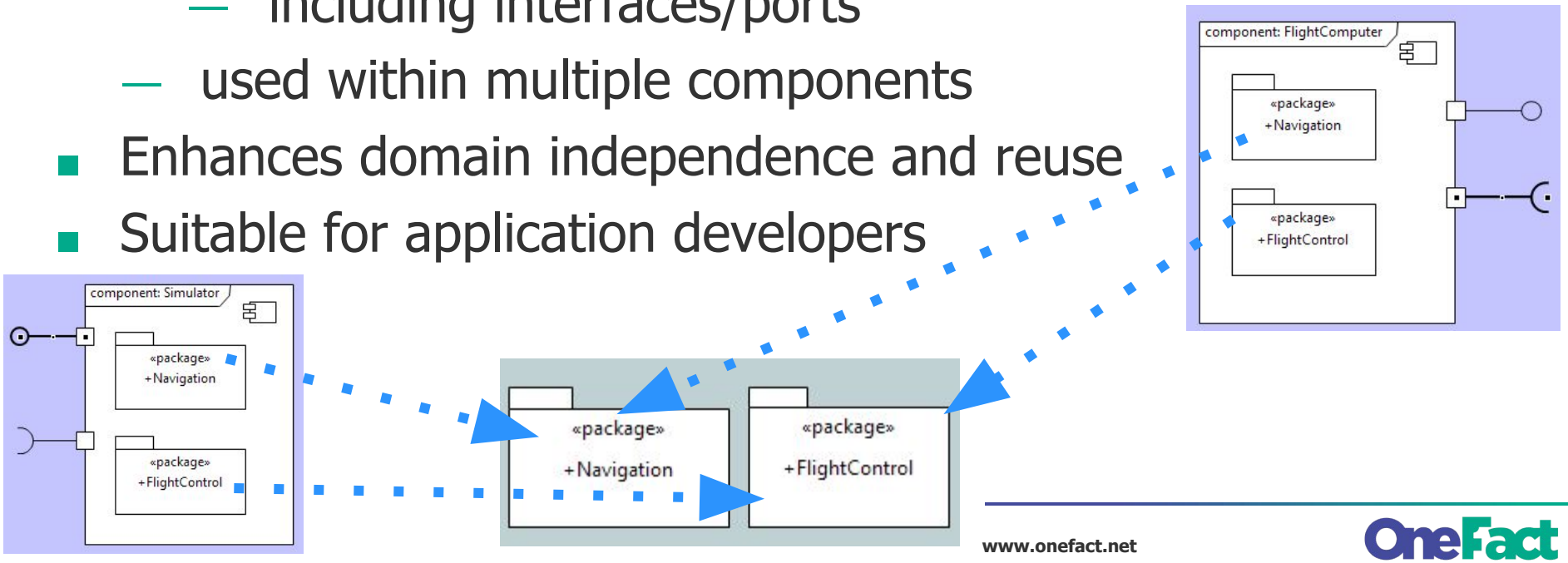
OAL Automatic Reference Maintenance

- Additional intelligence in editor
- Automatically updates OAL references to identifiers
- Normalizes model, maintaining one fact in one place
- Intersects with MASL

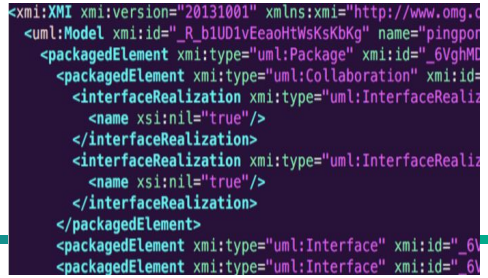


Package References, Phase 2

- BP 5.8, Package References Phase 1:
  - Reuse passive class models (e.g., meta-model)
  - Mainly for model compilers and tooling
- Phase 2 allows entire domains to be:
  - defined within packages, outside components
    - including interfaces/ports
  - used within multiple components
- Enhances domain independence and reuse
- Suitable for application developers



Persistence Layer Updates




```
object Group is
  group_id   : preferred unique integer;
  group_name : string;

end object;
pragma id (3);
pragma key_letter ("G");

object Group_For_Session is
  group_id   : preferred referential (R2.is_member_of.
  session_id : preferred referential (R2.current_membe

end object;
```

- Persist in a modeling language
  - BP currently stores models as:
    - meta-model instances, rendered in SQL
  - EMF persistence would store models as:
    - meta-model instances, rendered in XML
  - Model persisted as meta-model instances:
    - are large, hard to read and repair
    - significantly complicate compare/merge
  - Models persisted in a modeling language
    - are more concise, easily read and repaired
    - simplify compare/merge
- Persist graphics separately

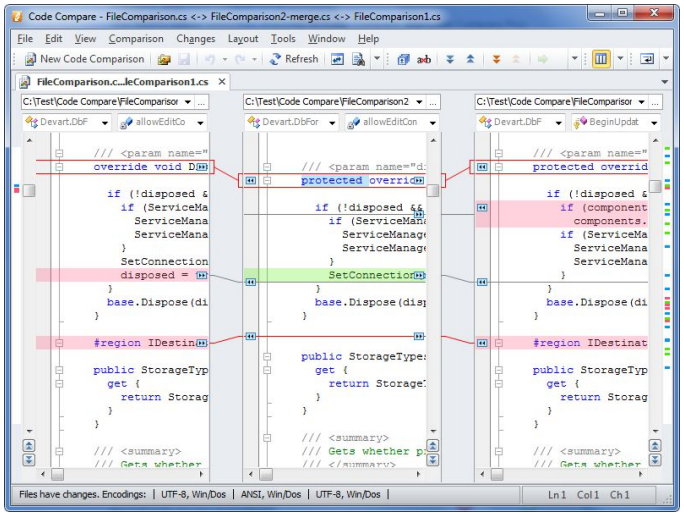
Load Improvements (Proxy Removal?)

- Current BP architecture relies on proxies between:
  - referring elements and referred-to elements
  - example: classes carry proxies for data types
- Introduced years ago to reduce model load times
  - no longer necessary
  - complicates tool enhancement
  - examples: model element move, compare/merge
- Removal requires significant work
  - perhaps best done after persistence layer is updated



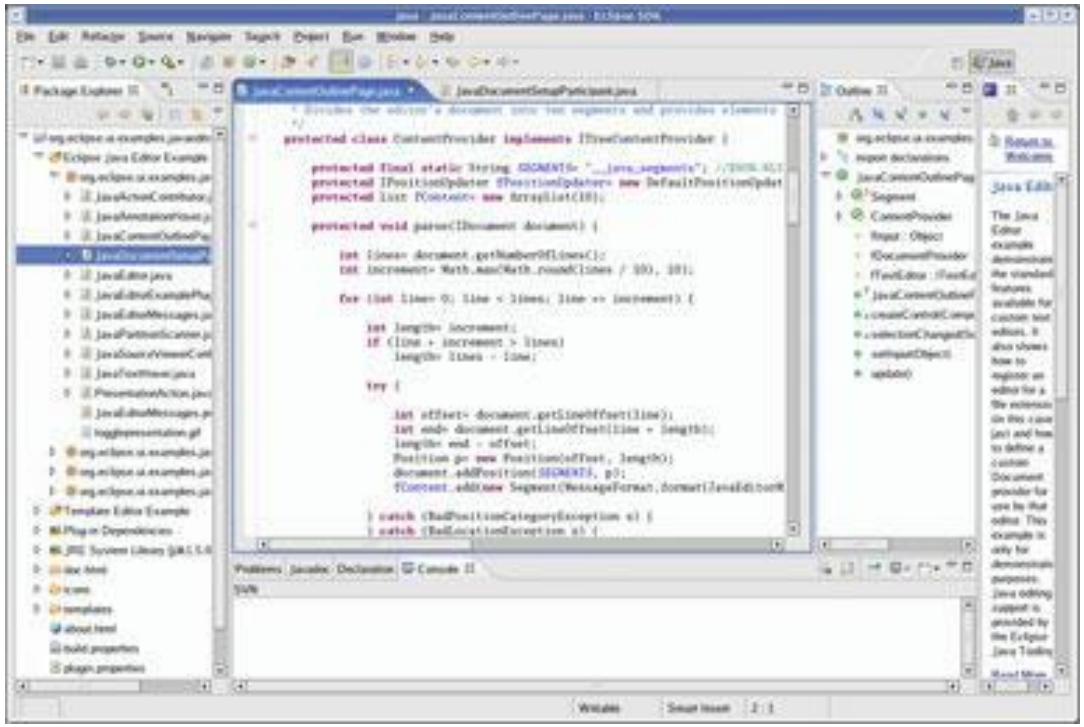
Compare and Merge

- Current BP support works in many circumstances
  - requires disciplined, process-focused use
  - failures can be difficult to untangle
- Robust support is required to
  - support existing users
  - appeal to mainstream users
- Perhaps best tackled after persistence layer is updated



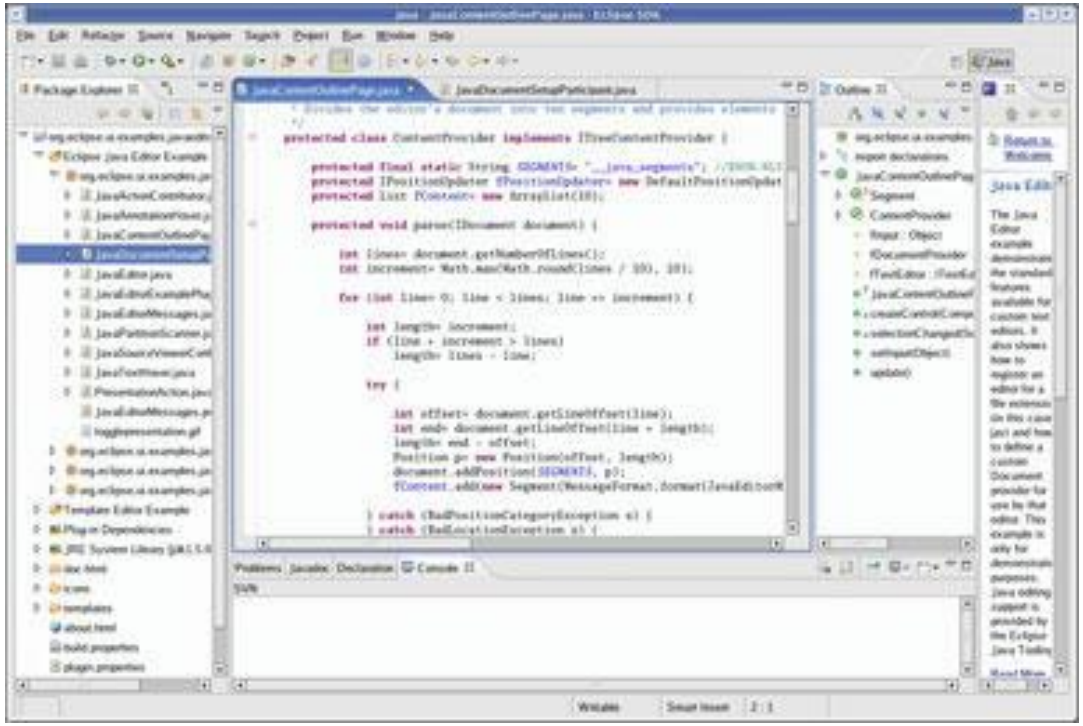
Activity Signature Editor

- Current approach requires many menu clicks
- Add ability to edit activity signatures within a file



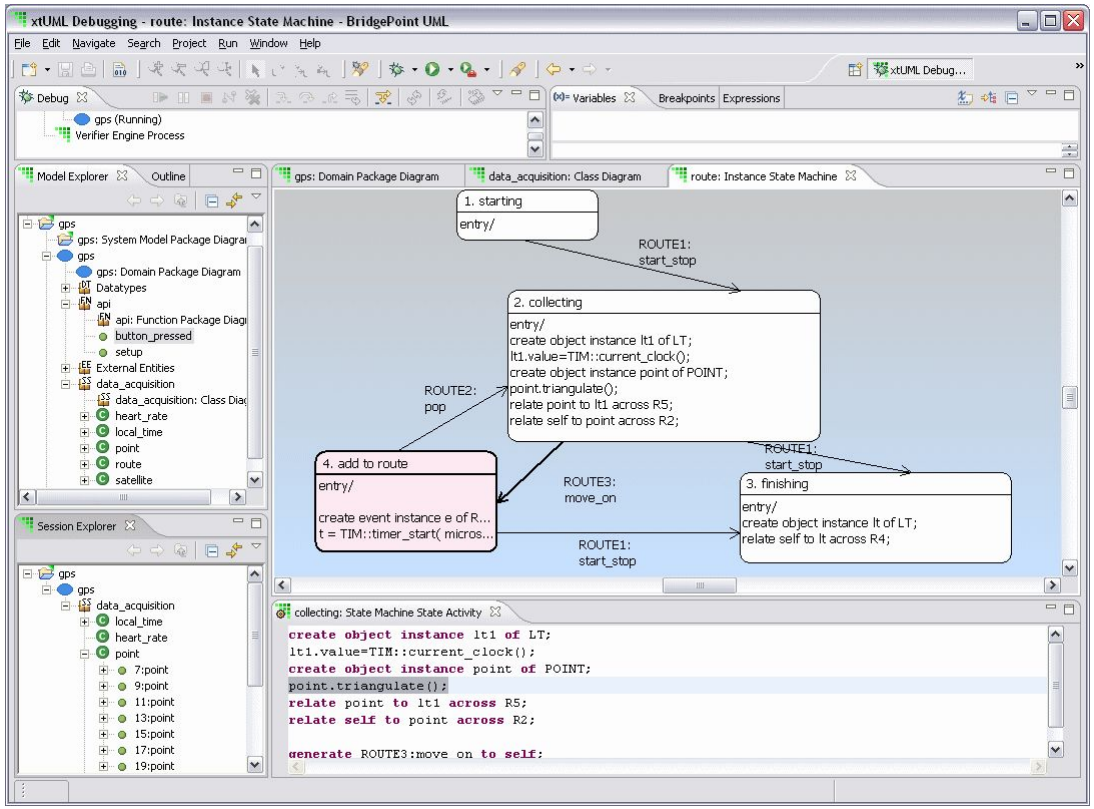
Multi-body Activity Editor

- Edit multiple activities within a single editor session
- Eases refactoring and restructuring of activities
- Increases mainstream appeal



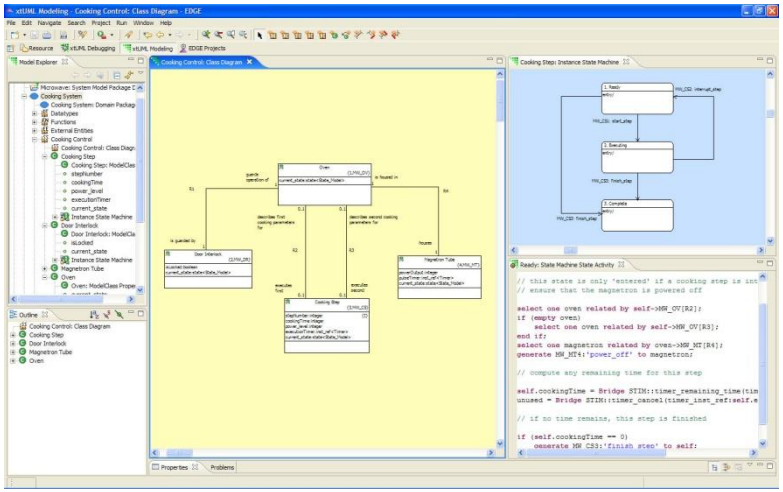
Verifier for MASL

- Verifier provides interpretive execution of models
  - presently supports only OAL-infused xtUML models



Inspector Integration

- Provides interactive debugging of generated code
  - presently supports only MASL models
  - stand-alone tool, not integrated with Eclipse or BP
- Integration could include
  - support for OAL and MASL
  - integration with Eclipse and BridgePoint
  - support for multiple model compilers
  - integration with Verifier (hybrid execution)



Marking Solution

- BP presently provides little support (mainly for MASL)
- Some model compilers rely on text files
  - maintained separately from models ☺
  - difficult to correlate with models being marked ☹
- Some solutions intermix marking and model data ☹
- Requirements
  - maintain marking and model data separately
  - easily view, edit marking data within model context
  - support multiple model compilers





Pre-existing Instance Solution

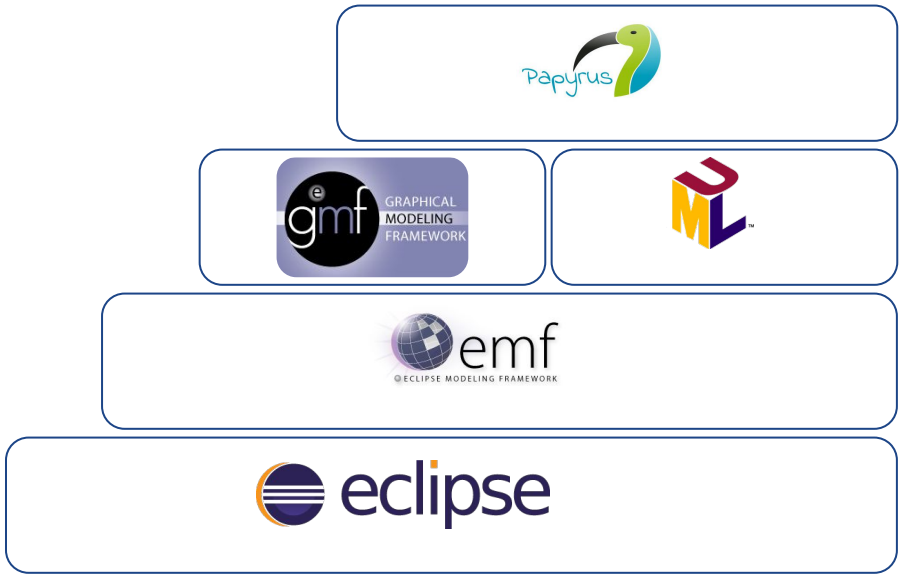
- BP presently provides no support
- Action-language based idiom
  - single definition for Verifier and generated code ☺
  - current tool support has…rough edges ☹
- Other approaches considered:
  - tabular
  - network
- Requirements?



Migration to Papyrus Platform

- Adopt mainstream OSS packages
  - focus on adding capabilities, not duplicating them
- Leverage growing community of developers and users
- Benefit from Papyrus Industry Consortium marketing







onefact.net