Summary - Osterweil ’87

Newman Abstract: Radical solution - A radical solution to the problem of software process control, communication (hand down/around best practice knowledge), and predictability is described, based on concept of describing software processes by "programming" them much as we "program" computer applications. In comparison with using prose to describe processes it offers a formalism that lends itself to controlling, communicating and predicting process outcome, which have been demonstrated in preliminary tests, but it leaves a number of side effects to be addressed including the evolution of PDLs, execution engines, and integration to programming tools. Strategies are suggested for addressing these side effects.

Question (Feasibility): Is it possible to describe software processes in a language much like a programming language?
Result (Notation): Such a language, APPL/A, is described.
Validation (Evaluation – Descriptive model): This language lends itself to handing down of software processes (primary goal) and with further work might be useful for actually controlling and predicting process execution.
Significance: Created field of Process Programming
Let’s play “follow the authors”

- Identified all authors of the key process programming papers ('87 Osterweil -> '98 PROSYT)
- Identified all papers those authors wrote using ACM’s digital library
- Recorded all key words for those papers
- Tallied overlapping use of key words

“Follow the authors” – top hits

7 – Software process
5 – Process-centered Software Engineering Environments (PSEE), Business process, Decentralized, Inconsistency/deviation, Event-based middleware
4 – CASE, PML, Document management, Configuration management, Web services
3 – Middleware, Interoperability, Static analysis, Petri-nets

Related to:
- Process programming
- Process programming “in-the-world” problems
- Web services/SOA related
- Other

Expanding upon “follow the authors”

- Web-page scraping tool (Python) to gather:
  - Papers with abstracts and links to:
    - Authors
    - Institutions
    - Key words
    - Collaborators
    - Citations and Cited by relationships
  - Queries
  - Grouping
  - Weighting by citations (think Google)
  - Expand beyond zero degrees of separation (by Collaborators or Citations)

Notably missing is Measurement (only received 2 hits)
Software process definition languages

Timeline

Timeline

Communication style
- again two camps

UI
- two camps

Data modeling
- XML Schema

Flow/Process
- BPEL (also called WS-BPEL)

Significance:
(1) Broke event-based/distributed architectural infrastructure apart from classical PDL researcher processes.
(2) Identifies architectures/technologies.

Validation (Evaluation - Descriptive model): The classification framework is used to evaluate the author's model (ESM) against a variety of other architectures from CORBA to modern proprietary event-based infrastructures.

Significance:
(1) Breaks event-based/distributed architectural infrastructure apart from classical PDL researcher processes.
(2) Identifies classification scheme for event-based infrastructures.
(3) Classical PDL researcher moving on to web architectures work.

Newman Abstract: Enhanced model - Existing architectural style classification systems are deficient in dealing with the wide variety of technologies available for use in distributed and intermittently disconnected systems. An event-based architectural style classification system is described enables us to "publish" object access, "debounce" data and predictions of capabilities and limitations in distributed systems. The model has been tested by comparing analyses with empirically measured values of properties.

Question (Method for analysis): How do I distinguish between various architectural styles for distributed systems?

Result (Descriptive model): Two previous models are references and somewhat merged/designed that upon creating an elegant orthogonal classification framework based upon (1) event model, (2) subscription approach, (3) observation/notification model, and (4) event dispatch architecture.

Summary - Cupola '01

Newman Abstract: Enhanced model - Existing architectural style classification systems are deficient in dealing with the wide variety of technologies available for use in distributed and intermittently disconnected systems. An event-based architectural style classification system is described enables us to "publish" object access, "debounce" data and predictions of capabilities and limitations in distributed systems. The model has been tested by comparing analyses with empirically measured values of properties.

Question (Method for analysis): How do I distinguish between various architectural styles for distributed systems?

Result (Descriptive model): Two previous models are references and somewhat merged/designed that upon creating an elegant orthogonal classification framework based upon (1) event model, (2) subscription approach, (3) observation/notification model, and (4) event dispatch architecture.

Validation (Evaluation - Descriptive model): The classification framework is used to evaluate the author's model (ESM) against a variety of other architectures from CORBA to modern proprietary event-based infrastructures.

Significance:
(1) Breaks event-based/distributed architectural infrastructure apart from classical PDL researcher processes.
(2) Identifies classification scheme for event-based infrastructures.
(3) Classical PDL researcher moving on to web architectures work.

Current web services architecture

- Underlying - XML, and event-based/message-based architecture
- Flow/Process - BPEL (also called WS-BPEL)
- Remote process programming/PDLs, Petri nets, Pi Calculus, BPMN, ebXML, BPEL4WS
- Data modeling - XML Schema
- Interface description - WSDL
- Queries - XPath
- UI - two camps
  - XForms
- Roots: HTML Forms, Spreadsheet technology
- XUL (Mozilla et. al.), XAML (MS-Avalon), etc.
- Roots: MVC architecture, etc.
- Communication style - again two camps
  - SOAP/XML-RPC
  - Roots: Event-based architectures, CORBA
  - REST/RESTLETS
  - Roots: HTTP, CRUD/ACID
- Further from center
  - UDDI
  - WS-*
WS-* What a mess?

- Some of WS-* good
  - WSDL
  - WS-BPEL (MS, IBM and BEA all merged!)
  - WS-I (SOAP, WSDL, X-Path, XML-Schema)
- Much of it just layers of goop
  - What happens when you try to innovate with standards
  - Meta models of meta models
    - Meta models of interface definition (or coordination, or routing, etc.)
    - That are used to describe actual interface definition languages
    - That are used to describe actual interfaces
    - That are used by process definitions
    - That were defined in process definition languages
    - That were described in PDL meta models
- Conclusion – Just use WS-I for now and add others as layers if later required

UGG!!!