Product Lines for Service Oriented Applications - PL for SOA

Maurice H. ter Beek
ISTI–CNR, Pisa, Italy

joint work with
Mercy N. Njima and Stefania Gnesi
IMT Lucca and ISTI–CNR

WWV 2011

Reykjavík, Ísland
9 June 2011
Content

1. Background and research aim
2. Motivating example: Smart Grids
3. Variability modeling of Smart Grid
4. Conclusions and future work
SOA to enable the assembly, orchestration and maintenance of enterprise solutions to quickly react to changing business requirements

SPL to systematically capture and exploit commonality among a set of related systems, while managing variations for specific customers or market segments
Background

**SOA** to enable the assembly, orchestration and maintenance of enterprise solutions to quickly react to changing business requirements

**SPL** to systematically capture and exploit commonality among a set of related systems, while managing variations for specific customers or market segments

⇒ ‘How can the use of product line practices support service-oriented applications?’
Service Product Lines

- PLE technology increasingly finds its way to software sector
- SPL can be considered as the most successful approach to intra-organizational reuse of software
- Combining SPL and SOA could become a new development paradigm that can help provide answers to the need for agility, versatility and economies
- Would achieve flexibility of network-based systems through service orientation, but still manage product variations through PLE techniques
- Service features are selected and/or parameterized at runtime by a user or by a product itself when a certain contextual change or a new service provider is recognized
PLE technology increasingly finds its way to software sector
SPLE can be considered as the most successful approach to intra-organizational reuse of software
Combining SPL and SOA could become a new development paradigm that can help provide answers to the need for agility, versatility and economies
Would achieve flexibility of network-based systems through service orientation, but still manage product variations through PLE techniques
Service features are selected and/or parameterized at runtime by a user or by a product itself when a certain contextual change or a new service provider is recognized
⇒ We adapt a feature-oriented PLE approach to SOC
Feature diagrams

- A family of popular modeling languages used for engineering requirements in SPL

- Represented as the nodes of a tree, with the product family being the root and have the following features:
Feature diagrams

- A family of popular modeling languages used for engineering requirements in SPL

- Represented as the nodes of a tree, with the product family being the root and have the following features:
  - **optional** features may be present in a product only if their parent is present
  - **mandatory** features are present in a product if and only if their parent is present
  - **alternative** features are a set of features among which one and only one is present in a product if their parent is present
With additional inter-feature constraints a feature diagram results in a feature model:
Feature models

With additional inter-feature constraints a feature diagram results in a feature model:

**requires** is a unidirectional relation between two features indicating that the presence of one feature requires the presence of the other

**excludes** is a bidirectional relation between two features indicating that the presence of either feature is incompatible with the presence of the other
Motivating example: Smart Grid

Definition

Next-generation, managed electrical power system that leverages increased use of communications and information technology in the generation, delivery and consumption of electrical energy
Motivating example: Smart Grid

Definition

Next-generation, managed electrical power system that leverages increased use of communications and information technology in the generation, delivery and consumption of electrical energy

⇒ Consists of solutions based on both current and future telecommunication technologies for command and control, metering and charging
The Smart Grid (adopted from smartgrid.epri.com)
Properties of the Smart Grid

- Self-heal: enables the problematic elements to be restored with little or no human intervention
- Motivate and include the consumer in energy decisions
- Security: resists attack
- Provide power quality for 21st century needs
- Accommodate all generation and storage options
- Enable markets
- Optimize assets and operate efficiently
Electric utilities, in a reactive or proactive answer to these new challenges, are adding more intelligence and complexity in their distribution networks.

As the grid becomes more intelligent and more complex, the tools to operate it become increasingly important.
Future Smart Grids

- Electric utilities, in a reactive or proactive answer to these new challenges, are adding more intelligence and complexity in their distribution networks.
- As the grid becomes more intelligent and more complex, the tools to operate it become increasingly important.
- Hence the need for interoperability (SOA), flexibility and variability (SPL).
Future Smart Grids

- Electric utilities, in a reactive or proactive answer to these new challenges, are adding more intelligence and complexity in their distribution networks.

- As the grid becomes more intelligent and more complex, the tools to operate it become increasingly important.

- Hence the need for interoperability (SOA), flexibility and variability (SPL).

  ⇒ End result: electricity provision as a service and the Smart Grid as a service product line.
Variability Modeling of Smart Grids

The generic Smart Grid will be modeled as a family with basic components for basic products and specialized properties for some of the products such as:

1. Storage
2. Renewables: varies with weather, time, season and other intermittent effects
3. Vehicle to Grid (V2G): establishing a viable transparent business model, accurate forecasting of renewable energy supply and demand

Load shifting and V2G can reduce the energy storage capacity required to maintain power quality
Variability Modeling of Smart Grids

- The generic Smart Grid will be modeled as a family with basic components for basic products and specialized properties for some of the products such as:

1. Storage
2. Renewables: varies with weather, time, season and other intermittent effects
3. Vehicle to Grid (V2G): establishing a viable transparent business model, accurate forecasting of renewable energy supply and demand
The generic Smart Grid will be modeled as a family with basic components for basic products and specialized properties for some of the products such as:

1. **Storage**
2. **Renewables**: varies with weather, time, season and other intermittent effects
3. **Vehicle to Grid (V2G)**: establishing a viable transparent business model, accurate forecasting of renewable energy supply and demand

Load shifting and V2G can reduce the energy storage capacity required to maintain power quality
Feature Model of the Smart Grid family

SMART GRID

GRID MONITORING
  MANAGEMENT
    SELF HEALING
    SMART METERS
    INTEGRATED COMMUNICATIONS

DEMAND RESPONSE
    LOAD SHIFTING
    EFFICIENT MARKETS
    INCORPORATES CUSTOMER

INTEGRATION OF RENEWABLES
    ELECTRIC VEHICLES
    STORAGE
    VEHICLE TO GRID
A derived product without integration of renewables

Mandatory

SMART GRID

GRID MONITORING
  MANAGEMENT

SELF HEALING
SMART METERS
INTEGRATED COMMUNICATIONS

DEMAND RESPONSE

LOAD SHIFTING
EFFICIENT MARKETS
INCORPORATES CUSTOMER

M.H. ter Beek et al.  PL for SOA
Smart Grid Demand Response

- Demand response initiatives seek to reduce peak loads and defer additional generation capacity
Demand response initiatives seek to reduce peak loads and defer additional generation capacity

1. Offer flexible tariffs like critical-peak and real-time pricing
2. Two-way communications allow for pricing information to be transmitted to customers based on price changes each day and at timed intervals, determined by software at the enterprise level to allow real-time or day-ahead management
3. Exception pricing as well as price changes associated with system emergency conditions and quantity available to enable the customer to either buy or sell depending on their capacity
Smart Grid Demand Response

- Demand response initiatives seek to reduce peak loads and defer additional generation capacity

1. Offer flexible tariffs like critical-peak and real-time pricing
2. Two-way communications allow for pricing information to be transmitted to customers based on price changes each day and at timed intervals, determined by software at the enterprise level to allow real-time or day-ahead management
3. Exception pricing as well as price changes associated with system emergency conditions and quantity available to enable the customer to either buy or sell depending on their capacity

⇒ When the utility has excess supply of electricity, it will take advantage of existing agreements with their customers to sell and allow the system to get back to a state of equilibrium
MTS for the demand response function

Virtual Power Plant

Low Supply

High Supply

Aggregator

Market

Quantity

Price

Day ahead

Realtime

Buy

Sell

Equilibrium

Agreement

Load Shift

M.H. ter Beek et al. PL for SOA
Demand response behaviour when supply is high

- Virtual Power Plant
- High Supply
- Aggregator
- Market
- Load Shift
- Agreement
- Pricing
- Day ahead
- Sell
- Equilibrium
Modal Transition Systems (MTSs)

Originally introduced by Larsen & Thomsen @ LICS 1988

MTSs are now an accepted model to formalize a product family’s
- underlying behaviour, shared among all products, and
- variation points, differentiating between products

MTS is an LTS distinguishing between may and must transitions
(modelling optional or mandatory features, resp.)
Modal Transition Systems (MTSs)

Originally introduced by Larsen & Thomsen @ LICS 1988

MTSs are now an accepted model to formalize a product family’s
- underlying behaviour, shared among all products, and
- variation points, differentiating between products

MTS is an LTS distinguishing between may and must transitions
(modelling optional or mandatory features, resp.)

MTS cannot model constraints regarding alternative features (only one may be present) nor those regarding inter-feature relations (a feature’s presence requires or excludes that of another feature)
Modal Transition Systems (MTSs)

Originally introduced by Larsen & Thomsen @ LICS 1988

MTSs are now an accepted model to formalize a product family’s
- underlying behaviour, shared among all products, and
- variation points, differentiating between products

MTS is an LTS distinguishing between may and must transitions
(modeling optional or mandatory features, resp.)

MTS cannot model constraints regarding alternative features (only one may be present) nor those regarding inter-feature relations (a feature’s presence requires or excludes that of another feature)

We will model such advanced variability constraints by means of an associated set of logical formulae expressed in the variability and action-based branching-time temporal logic vACTL
### Definition of MTS

$(Q, A, \bar{q}, \delta^{\square}, \delta^{\Diamond})$ is an MTS with

- **underlying LTS** $(Q, A, \bar{q}, \delta^{\square} \cup \delta^{\Diamond})$
- **may transition relation** $\delta^{\Diamond} \subseteq Q \times A \times Q$ (*possible transitions*)
- **must transition relation** $\delta^{\square} \subseteq Q \times A \times Q$ (*mandatory transitions*)

By definition, mandatory transitions must also be possible: $\delta^{\square} \subseteq \delta^{\Diamond}$

Reasoning on 3-valued logic; truth values *true, false* and *unknown*
(Q, A, q, δ□, δ◊) is an MTS with

- underlying LTS (Q, A, q, δ□ ∪ δ◊)
- may transition relation δ◊ ⊆ Q × A × Q (possible transitions)
- must transition relation δ□ ⊆ Q × A × Q (mandatory transitions)

By definition, mandatory transitions must also be possible: δ□ ⊆ δ◊

Reasoning on 3-valued logic; truth values true, false and unknown

σ = q_1a_1q_2a_2q_3⋯ is a must path σ□ if q_i \xrightarrow{a_j} □ q_{i+1}, for all i > 0

The set of all must paths from q_1 is denoted by □-path(q_1)
Definition of MTS

\((Q, A, \overline{q}, \delta^\square, \delta^\Diamond)\) is an MTS with

- underlying LTS \((Q, A, \overline{q}, \delta^\square \cup \delta^\Diamond)\)
- may transition relation \(\delta^\Diamond \subseteq Q \times A \times Q\) (possible transitions)
- must transition relation \(\delta^\square \subseteq Q \times A \times Q\) (mandatory transitions)

By definition, mandatory transitions must also be possible: \(\delta^\square \subseteq \delta^\Diamond\)

Reasoning on 3-valued logic; truth values true, false and unknown

\[\sigma = q_1 a_1 q_2 a_2 q_3 \cdots\] is a must path \(\sigma^\square\) if \(q_i \xrightarrow{a_i} q_{i+1}\), for all \(i > 0\)

The set of all must paths from \(q_1\) is denoted by \(\square\)-path\((q_1)\)

Subfamilies/products obtained by preserving all must transitions, turning some may transitions into must transitions, and removing some/all remaining ones
Syntax of \( \mathcal{vACTL} \): variability and action-based branching-time temporal logic

A temporal logic based on the “Hennessy-Milner logic with until”, but augmented with deontic \( O \) (obligatory) and \( P \) (permitted) operators, CTL’s path operators \( E \) and \( A \) and ACTL’s action-based Until operator, both with and without a deontic interpretation
Syntax of νACTL: variability and action-based branching-time temporal logic

A temporal logic based on the “Hennessy-Milner logic with until”, but augmented with deontic $O$ (obligatory) and $P$ (permitted) operators, CTL’s path operators $E$ and $A$ and ACTL’s action-based Until operator, both with and without a deontic interpretation

$\phi ::= true \mid \neg \phi \mid \phi \land \phi' \mid \langle a \rangle \phi \mid [a] \phi \mid \langle a \rangle^\square \phi \mid [a]^\square \phi \mid E \pi \mid A \pi$

$\pi ::= \phi \{ \varphi \} U \{ \varphi' \} \phi' \mid \phi \{ \varphi \} U^\square \{ \varphi' \} \phi'$

Defines state formulae $\phi$, path formulae $\pi$ and action formulae $\varphi$ (boolean compositions of actions) over atomic actions $\{a, b, \ldots\}$
A temporal logic based on the “Hennessy-Milner logic with until”, but augmented with deontic $O$ (obligatory) and $P$ (permitted) operators, CTL’s path operators $E$ and $A$ and ACTL’s action-based Until operator, both with and without a deontic interpretation.

\[
\phi ::= \text{true} \mid \neg \phi \mid \phi \land \phi' \mid \langle a \rangle \phi \mid [a] \phi \mid \langle a \rangle \Box \phi \mid [a] \Box \phi \mid E \pi \mid A \pi
\]

\[
\pi ::= \phi \{ \varphi \} U \{ \varphi' \} \phi' \mid \phi \{ \varphi \} U^{\Box} \{ \varphi' \} \phi'
\]

Defines state formulae $\phi$, path formulae $\pi$ and action formulae $\varphi$ (boolean compositions of actions) over atomic actions \{a, b, \ldots\}

$\langle a \rangle \Box$ and $[a] \Box$ represent classic deontic modalities $O$ and $P$, resp.
\( q \models \text{true} \) always holds

\( q \models \neg \phi \) iff not \( q \models \phi \)

\( q \models \phi \land \phi' \) iff \( q \models \phi \) and \( q \models \phi' \)

\( q \models \langle a \rangle \phi \) iff \( \exists q' \in Q \) such that \( q \xrightarrow{a} q' \), and \( q' \models \phi \)

\( q \models [a] \phi \) iff \( \forall q' \in Q \) such that \( q \xrightarrow{a} q' \), we have \( q' \models \phi \)

\( q \models \langle a \rangle [\square] \phi \) iff \( \exists q' \in Q \) such that \( q \xrightarrow{a} q' \), and \( q' \models \phi \)

\( q \models [a] [\square] \phi \) iff \( \forall q' \in Q \) such that \( q \xrightarrow{a} q' \), we have \( q' \models \phi \)

\( q \models E \pi \) iff \( \exists \sigma' \in \text{path}(q) \) such that \( \sigma' \models \pi \)

\( q \models A \pi \) iff \( \forall \sigma' \in \text{path}(q) \) such that \( \sigma' \models \pi \)

\( \sigma \models \phi \{ \varphi \} U \{ \varphi' \} \phi' \) iff \( \exists j \geq 1 : \sigma(j) \models \phi' \), \( \sigma\{j\} \models \varphi' \), and \( \sigma(j + 1) \models \phi' \), and \( \forall 1 \leq i < j : \sigma(i) \models \phi \) and \( \sigma\{i\} \models \varphi \)

\( \sigma \models \phi \{ \varphi \} U^\square \{ \varphi' \} \phi' \) iff \( \sigma \) is a must path \( \sigma^\square \) and \( \sigma^\square \models \phi \{ \varphi \} U \{ \varphi' \} \phi' \)
\begin{itemize}
  \item $q \models \text{true}$ always holds
  \item $q \models \neg \phi$ iff not $q \models \phi$
  \item $q \models \phi \land \phi'$ iff $q \models \phi$ and $q \models \phi'$
  \item $q \models \langle a \rangle \phi$ iff $\exists q' \in Q$ such that $q \xrightarrow{a} q'$, and $q' \models \phi$
  \item $q \models [a] \phi$ iff $\forall q' \in Q$ such that $q \xrightarrow{a} q'$, we have $q' \models \phi$
  \item $q \models \langle a \rangle \Box \phi$ iff $\exists q' \in Q$ such that $q \xrightarrow{a} q'$, and $q' \models \phi$
  \item $q \models [a] \Box \phi$ iff $\forall q' \in Q$ such that $q \xrightarrow{a} q'$, we have $q' \models \phi$
  \item $q \models \exists \pi \in \text{path}(q)$ such that $\sigma' \models \pi$
  \item $q \models \forall \pi \in \text{path}(q)$ such that $\sigma' \models \pi$
  \item $\sigma \models \phi \{\varphi\} \cup \{\varphi'\} \phi'$ iff $\exists j \geq 1 : \sigma(j) \models \phi'$, $\sigma(j) \models \varphi'$, and $\sigma(j + 1) \models \phi'$, and $\forall 1 \leq i < j : \sigma(i) \models \phi$ and $\sigma(i) \models \varphi$
  \item $\sigma \models \phi \{\varphi\} \cup^\Box \{\varphi'\} \phi'$ iff $\sigma$ is a must path $\sigma^\Box$ and $\sigma^\Box \models \phi \{\varphi\} \cup \{\varphi'\} \phi'$
\end{itemize}

Abbreviations: $EF \phi = E(\text{true} \{\text{true}\} \cup \{\text{true}\} \phi)$; $AG \phi = \neg EF \neg \phi$; $EF^\Box \phi = E(\text{true} \{\text{true}\} \cup^\Box \{\text{true}\} \phi)$; $EF \{\varphi\} \text{true} = E(\text{true} \{\text{true}\} \cup \{\varphi\} \text{true})$; $EF^\Box \{\varphi\} \text{true} = E(\text{true} \{\text{true}\} \cup^\Box \{\varphi\} \text{true})$; $AG^\Box \phi = \neg EF^\Box \neg \phi$; etc.
Advanced variability management with $\nu\text{ACTL}$

Complement behavioural description of MTSs by expressing constraints over products of a family that MTSs cannot model.
Complement behavioural description of MTSs by expressing constraints over products of a family that MTSs cannot model.

Template ALT: Features F1 and F2 are alternative

\((\mathsf{EF}\{F1\} \text{ true} \lor \mathsf{EF}\{F2\} \text{ true}) \land \neg(\mathsf{EF}\{F1\} \text{ true} \land \mathsf{EF}\{F2\} \text{ true})\)
Advanced variability management with $\nu$CTL

Complement behavioural description of MTSs by expressing constraints over products of a family that MTSs cannot model

**Template ALT:** Features F1 and F2 are *alternative*

$\left( EF^{\square} \{F1\} \text{true} \lor EF^{\square} \{F2\} \text{true} \right) \land \neg \left( EF \{F1\} \text{true} \land EF \{F2\} \text{true} \right)$

**Template EXC:** Feature F1 *excludes* feature F2

$\left( \left( EF \{F1\} \text{true} \right) \implies \left( AG \neg \langle F2 \rangle \text{true} \right) \right) \land \left( \left( EF \{F2\} \text{true} \right) \implies \left( AG \neg \langle F1 \rangle \text{true} \right) \right)$
Advanced variability management with $\forall\text{CTL}$

Complement behavioural description of MTSs by expressing constraints over products of a family that MTSs cannot model

**Template ALT:** Features F1 and F2 are *alternative*

\[(EF\Box \{F1\} \text{ true} \lor EF\Box \{F2\} \text{ true}) \land \neg(EF \{F1\} \text{ true} \land EF \{F2\} \text{ true})\]

**Template EXC:** Feature F1 *excludes* feature F2

\[
((EF \{F1\} \text{ true}) \implies (AG \neg\langle F2\rangle \text{ true})) \land ((EF \{F2\} \text{ true}) \implies (AG \neg\langle F1\rangle \text{ true}))
\]

**Template REQ:** Feature F1 *requires* feature F2

\[(EF \{F1\} \text{ true}) \implies (EF\Box \{F2\} \text{ true})\]
## Advanced variability management with vACTL

Complement behavioural description of MTSs by expressing constraints over products of a family that MTSs cannot model

### Template ALT: Features F1 and F2 are alternative

\[
(\text{EF} \{F1\} \text{ true} \lor \text{EF} \{F2\} \text{ true}) \land \neg(\text{EF} \{F1\} \text{ true} \land \text{EF} \{F2\} \text{ true})
\]

### Template EXC: Feature F1 excludes feature F2

\[
((\text{EF} \{F1\} \text{ true}) \Rightarrow (\text{AG} \neg \langle F2 \rangle \text{ true})) \land ((\text{EF} \{F2\} \text{ true}) \Rightarrow (\text{AG} \neg \langle F1 \rangle \text{ true}))
\]

### Template REQ: Feature F1 requires feature F2

\[
(\text{EF} \{F1\} \text{ true}) \Rightarrow (\text{EF} \{F2\} \text{ true})
\]

No temporal ordering among related features: duty of behavioural LTS/MTS description of product/family, verifiable with vACTL
The SOA of the utility industry

We incorporate feature modeling into a SOA framework
The SOA of the utility industry

We incorporate feature modeling into a SOA framework

The major expectations and requirements are as follows

- The smart grid must provide all consumers with a highly reliable, flexible, accessible and cost-effective power supply
- End users will interact more with both markets and grids
- Electricity will be generated by centralized and dispersed sources
- Grid systems will become more interoperable to enhance security and cost-effectiveness
Like the Internet, the Smart Grid needs a standard layered architecture and needs to be distributed. It will deliver electricity over a two-way protocol from supplier to consumer, utilizing independent components that must cooperate. SOA will provide a framework for integration and flexibility for the services of the Smart Grid.
Like the Internet, the Smart Grid needs a standard layered architecture and needs to be distributed.

It will deliver electricity over a two-way protocol from supplier to consumer, utilizing independent components that must cooperate.

SOA will provide a framework for integration and flexibility for the services of the Smart Grid.

⇒ To model service product lines, we merge feature modeling and service-oriented concurrency calculus Orc.
Orc

- High-level orchestration of services that coordinates interactions among basic subsystems, called sites, by use of a small number of combinators.
- Language for task orchestration which can manage time-outs, priorities, failure of service and communication, which are features that are required for the Smart Grid.
- Since developing a distributed system is notoriously difficult, this high-level language will make development and maintenance easier.
- Combines a functional style with ideas from process algebra to aid the software engineer.
Why Orc?

- Dynamic nature of the various components and services of the Smart Grid: dynamic service, market management and pricing are basic building blocks of a Smart Grid system.

- Orc allows for the dynamic combination of services and the dynamic reconfiguration of software systems: idea is to invoke published services instead of developing isolated functions.

- The semantics is operational, asynchronous and based on LTSs, which we hope to extend to MTSs and create a PL for SOA calculus in the future.
Orc outlined

Allows integration of components and assumes that structured concurrent programs should be developed much like structured sequential programs, by decomposing a problem and combining the solutions with the site combinators.
Orc outlined

Allows integration of components and assumes that structured concurrent programs should be developed much like structured sequential programs, by decomposing a problem and combining the solutions with the site combinators.

- Sites are nondeterministic: a site may publish either nothing, or a value, but typically the value itself may not be uniquely determined.

- When sites are combined independently in parallel, even if each is deterministic, the result is not, because the order of publication is not determined and so interleaving occurs.

- Each site, and the result of each Orc computation, is modeled as the set of what it may publish (i.e. as a set of sequences of publishable values).
Orc combinators

- The *independent parallel combinator*, \((A|B)\), of expressions \(A\) and \(B\), publishes anything published by \(A\) or \(B\) independently.
- The sites called by \(A\) and \(B\) individually are called by \((A|B)\) and the values published by \(A\) and \(B\) are published by \((A|B)\).
The **sequential combinator**, \((A \ x \ B)\), initiates a new instance of \(B\) for every value published by \(A\) whose value is bound to name \(x\) in that instance of \(B\).

- The values published by \((A \ x \ B)\) are all instances of those published by \(B\).
- If \(A\) fails to publish, then so does \((A \ x \ B)\).
- If \(x\) is not used in \(B\), combinator is abbreviated by \((A \gg B)\).
The *asymmetric parallel combinator*, \((A \prec x \prec B)\), evaluates \(A\) and \(B\) independently, but the site calls in \(A\) that depend on \(x\) are suspended until \(x\) is bound to a value.

The first value from \(B\) is bound to \(x\), evaluation of \(B\) is then terminated and suspended calls in \(A\) are resumed; the values published by \(A\) are those published by \((A \prec x \prec B)\).

If \(B\) fails to publish, then so does \((A \prec x \prec B)\); but, otherwise, \((A \prec x \prec B)\) publishes the result of \(A\)’s invocation with the first publication of \(B\).

If \(x\) is not used in \(B\), combinator is abbreviated by \((A \ll B)\).
The *otherwise combinator*, \((A; B)\), executes \(A\) and if it completes and has not published any values, then \(B\) executes.

If \(A\) publishes one or more values, then \(B\) is ignored; the publications of \((A; B)\) are thus those of \(A\) if \(A\) publishes, and those of \(B\) otherwise.
Orc combinators for features

Orc combinators have an almost one-to-one correspondence with the feature relations of feature diagrams:

- The independent parallel combinator, \((A|B)\), can be used to specify mandatory features
- The sequential combinator, \((A >x> B)\), can be used to specify required features
- The asymmetric parallel combinator, \((A <x< B)\), can specify optional features
- The otherwise combinator, \((A ; B)\), can be used to specify alternative features, especially when there is a preferred feature or priority
Example: demand response in Orc

To demonstrate the functionality of Orc for our purposes we go back to our model for demand response.
Example: demand response in Orc

- To demonstrate the functionality of Orc for our purposes we go back to our model for demand response.

- We utilize the site `let (x, y)`, which returns values as a tuple when it receives the second value.
Example: demand response in Orc

- To demonstrate the functionality of Orc for our purposes we go back to our model for demand response.

- We utilize the site `let (x, y)`, which returns values as a tuple when it receives the second value.

```
let (u, v) < Load_shift < (real_time|day_ahead) < Agreement < (sell|buy)
```

- The values published by this expression are those contained in `let`, which acts as container for the first result published and releases both when the second value is received.
We have proposed that services can be modeled in a new way by incorporating variability notions from SPLs.
Conclusions and future work

We have proposed that services can be modeled in a new way by incorporating variability notions from SPLs.

The results give an indication that Orc could be the specification language to which we could add variability features in order to formalize service product lines.
We have proposed that services can be modeled in a new way by incorporating variability notions from SPLs.

The results give an indication that Orc could be the specification language to which we could add variability features in order to formalize service product lines.

We intend to extend the LTS-based semantics of Orc to an MTS-based semantics in order to utilize the $\nu$CTL logic for verification.
Publicity

Service-Oriented Architectures and Programming SOAP track @ 27th Annual ACM Symposium on Applied Computing (SAC 2012)

⇒ http://www.itu.dk/acmsac2012-soap/

IMPORTANT DATES (strict)

August 31, 2011: Paper submission
October 12, 2011: Author notifications
November 2, 2011: Camera-ready copy
March 25-29, 2012: SOAP @ SAC 2012