**Introduction**

Distributed nature and change over time of service computing impose new challenges on identifying the quality of services for effective service selection and composition.

We propose a technique for identifying distributed features by mining scattered dynamic call-trees.

- **Complexities of distributed features:**
  - Feature locations may change due to change of input parameters.
  - Execution traces are scattered among different service provider platforms.
  - Trace files contain interleaving of execution traces related to different concurrent service users.

- **Proposed solution:**
  - Define different sets of feature-specific scenarios and execute on SOA.
  - Collect and aggregate relevant distributed execution traces.
  - Mine the resulting dynamic call trees to spot: "feature-specific", "omnipresent", and "noise" patterns.
  - Use metrics to identify the structural properties of the services.

---

**Framework for Feature Identification in SOA**

- **Feature of interest**: in SOA environment, an enterprise application consists of one or more service operations. A service operation can be a specific feature of interest, e.g., op1 of s1.
- **Goal**: to run feature-specific scenario sets to generate distributed execution traces, and identify the execution pattern of the feature.
- **Feature-specific scenario set**: a group of task scenarios that all share a specific feature. Example: in banking service "deposit into a bank account", the operation "entering the amount of money" can be the specific feature (op1).

---

**Challenges in Dynamic Analysis of SOA**

- **Deterministic vs. Non-deterministic features:**
  - Deterministic: behavior of the feature is independent of input or state of the system. This feature always generates the same call tree.
  - Non-deterministic: produces different call trees depending on the state and input of the system. We define different Cases, where in each case some conditions are imposed such that the feature shows the same behavior. E.g., withdraw money when account exists and balance is enough.

- **Distribution of traces**: execution of a scenario may involve several services, and the traces are scattered among different platforms.

- **Concurrency of events**: a service is used by several concurrent users.

---

**Mining Dynamic Call-trees: “Mining Frequent Sub-tree in a Forest”**

Running feature-specific scenario sets produces “Forest of Dynamic Call Trees”

A frequent subtree is a subtree T’ such that the “cardinality” of its super-trees T’s (namely, “support set” of T’) is greater than or equal to a given “threshold value” (i.e., minimum support).

- **Frequent sub-tree mining algorithm builds bottom-up sub-trees.**
- **Forest of trees** is represented by a two dimensional array.

- **Example with four iterations** of the mining process:
  - Each array entry (e.g., P1[1]) consists of a pointer to the root of a subtree (P1) and the subtree’s support set ([1]).
  - Tree T’ is represented as string 716105203004.
  - Minimum support threshold is two.
Analysis of Extracted Patterns

- Traces associated with two feature-specific scenario sets and their corresponding patterns. For simplicity, we have serialized each trace based on the time of method entries.

- Feature-Specific Patterns
  - (feature 1: P3, deterministic)
  - (feature 2: P4 & P5, non-deterministic)

- Omnipresent patterns are common among all scenario sets:
  - (pattern P1 in both sets)

- Noise patterns are random:
  - (pattern P2 in both sets)

Case Study: Service Oriented Banking

- Two features: Withdraw & Email-money
  - Scenarios in each feature-specific set are partitioned into a number of cases, where each case examines a service operation in a specific context.
  - Scenarios W9 and W10 examine operation "withdraw" in the case that the account to be withdrawn from exists and the amount to withdraw is less than or equal to available funds.

Structural Analysis of SOA

- Service Utilization: assessing a service based on how its operations utilize other SOA services. $SU(op) = \sum_{s_i \in \text{all methods in } \text{service } s_i} \frac{|M_s^C \cap M_{op}|}{|M_s^C|}$
- Call Frequency: communication overhead of a service. $CF(p_i) = \sum_{s_j \in \text{all services contributing to } p_i} \frac{|M_{s_j}^C \cap M_{op}|}{|M_{s_j}^C|}$

Service Utilization & Call Frequency

- Execution pattern mining and analysis for operation "withdraw".
- Service utilization of operation "withdraw" from "Mybank Account Access Service" and "Banking Service".

Feature-specific Patterns

- Feature specific patterns extracted from the execution of "withdraw" on the first system in page 8.
- For simplicity, we serialized each pattern by noting only the sequence of method entries.
- Each pattern is a sequence of method invocations separated by ",".
- Each method is specified as: package-name/class-name.method-name.
Identification and maintenance of software features in distributed web services are more complex than in monolithic systems. We focus on identifying distributed features through dynamic call trees and explore non-deterministic execution-related properties. Our proposed approach involves collecting and mining distributed traces to analyze SOA-based systems. The effectiveness of the approach is demonstrated through a case study of a service-oriented banking application.