A Component-Based Approach to Feature Modelling

Pablo Parra[1], Óscar R. Polo[1], Segundo Esteban[2],
Agustín Martínez[1] and Sebastián Sánchez[1]

[1] Space Research Group
University of Alcalá

[2] ISCAR Research Group
Complutense University of Madrid

Universidad de Alcalá

SRG

Summary

Introduction

Our proposal

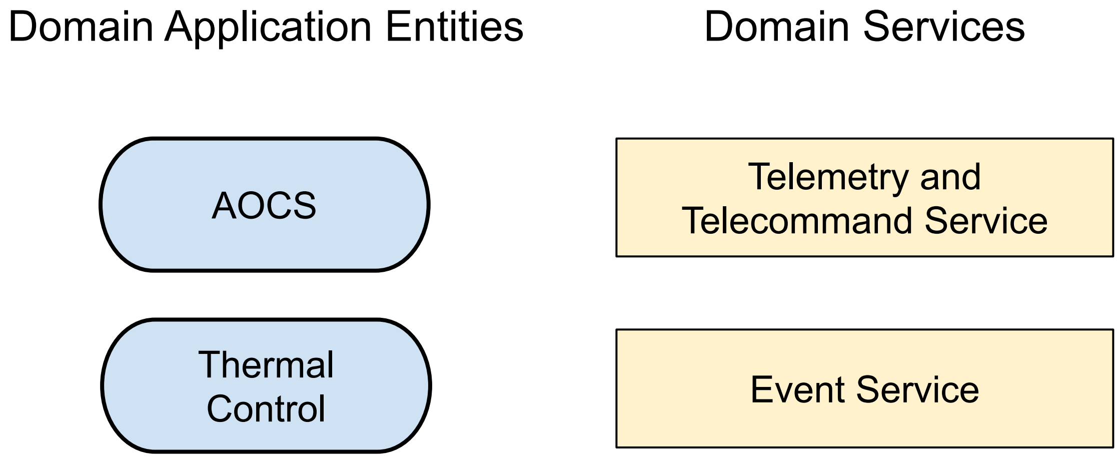
An approach to feature modeling inspired by the artifacts characteristic of component-based software design

- The models allow establishing feature hierarchies, clearly differentiating between the features themselves and their variants or realisations
- It enables the definition of complex dependency relationships between the different feature realisations
- It allows the modelling of product configurations as a set of interconnected and configured feature realisations
- The on-board satellite software domain has been used as an example of the proposed approach

On-board software domain

- The OBSW of a satellite is in charge of controlling the main procedures of the spacecraft:
  - Managing the transmission and reception of information to and from the ground
  - Performing housekeeping operations
  - Controlling and executing the different processes required by the payload
- The overall set of procedures may vary, depending on the type of satellite, e.g. scientific, communication, etc.
- An OBSW is conceptually divided into *applications*
  - An application is a software product that has their own specification and validation procedures
  - Each one is in charge of one aspect of the system
- Applications may use different *services* during their execution

On-board software domain


Domain Application Entities
Domain Services
AOCS
Thermal Control
Telemetry and Telecommand Service
Event Service

Component-based feature modelling

Feature ≡ Component Type or Classifier

A prominent or distinctive characteristic of a software system that is susceptible to having different realisations or variants

Feature realisation ≡ Component

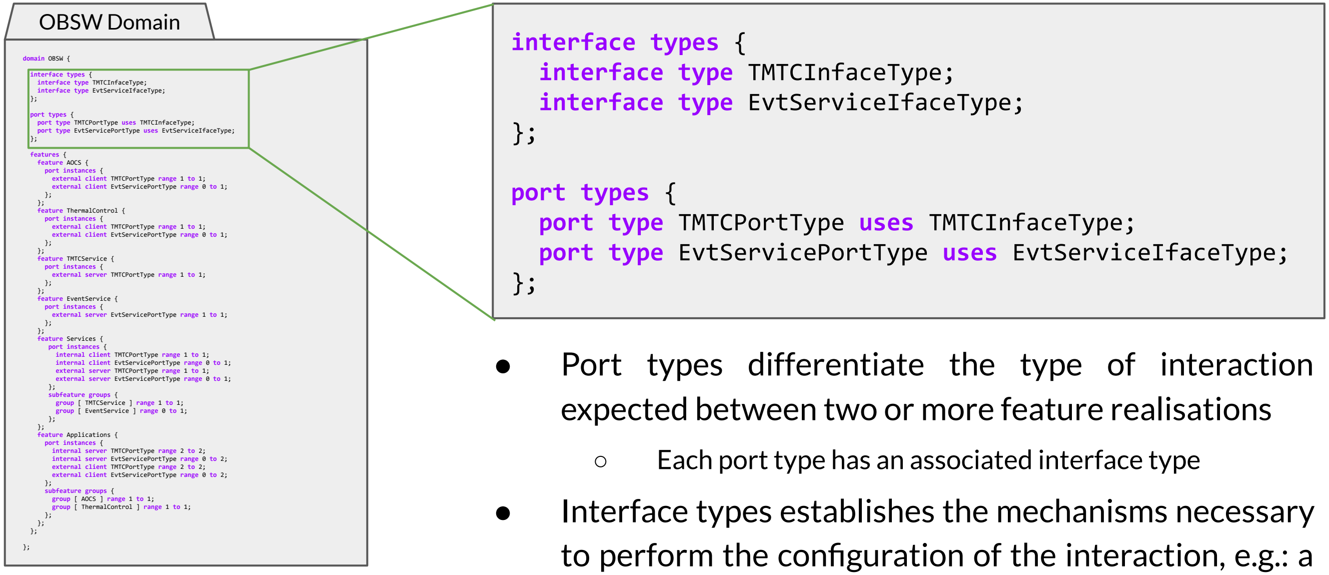
Each one of the possible alternatives of implementing a feature. They can define interaction points called *ports* through which they can require (*client ports*) or provide (*server ports*) services to other realisations

Product configuration ≡ Component assembly

A set of interconnected and configured feature realisations that model a single product within the software product line

Component-based feature modelling

Product features


OBSW Domain

```
domain OBSW {

  interface types {
    interface type TMTCInfaceType;
    interface type EvtServiceIfaceType;
  };

  port types {
    port type TMTCPortType uses TMTCInfaceType;
    port type EvtServicePortType uses EvtServiceIfaceType;
  };
  features {
    feature AOCS {
      port instances {
        external client TMTCPortType range 1 to 1;
        external client EvtServicePortType range 0 to 1;
      };
    };
    feature ThermalControl {
      port instances {
        external client TMTCPortType range 1 to 1;
        external client EvtServicePortType range 0 to 1;
      };
    };
    feature TMTCService {
      port instances {
        external server TMTCPortType range 1 to 1;
      };
    };
    feature EventService {
      port instances {
        external server EvtServicePortType range 1 to 1;
      };
    };
    feature Services {
       port instances {
         internal client TMTCPortType range 1 to 1;
         internal client EvtServicePortType range 0 to 1;
         external server TMTCPortType range 1 to 1;
         external server EvtServicePortType range 0 to 1;
       };
       subfeature groups {
         group [ TMTCService ] range 1 to 1;
         group [ EventService ] range 0 to 1;
       };
    };
    feature Applications {
      port instances {
        internal server TMTCPortType range 2 to 2;
        internal server EvtServicePortType range 0 to 2;
        external client TMTCPortType range 2 to 2;
        external client EvtServicePortType range 0 to 2;
      };
      subfeature groups {
        group [ AOCS ] range 1 to 1;
        group [ ThermalControl ] range 1 to 1;
      };
    };
  };

};
```

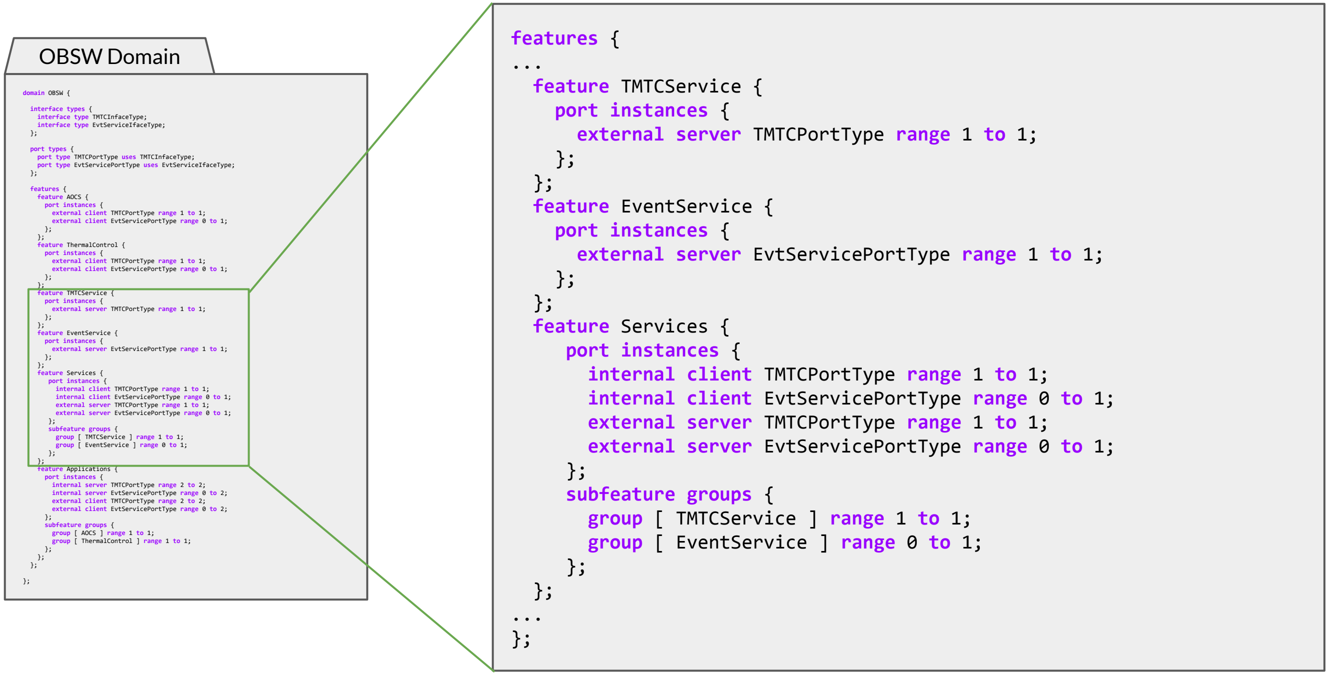
```
interface types {
  interface type TMTCInfaceType;
  interface type EvtServiceIfaceType;
};

port types {
  port type TMTCPortType uses TMTCInfaceType;
  port type EvtServicePortType uses EvtServiceIfaceType;
};
```

- Port types differentiate the type of interaction expected between two or more feature realisations
    - Each port type has an associated interface type
- Interface types establishes the mechanisms necessary to perform the configuration of the interaction, e.g.: a meta-model or an IDL

Component-based feature modelling

Product features


OBSW Domain

```
domain OBSW {

  interface types {
    interface type TMTCInfaceType;
    interface type EvtServiceIfaceType;
  };

  port types {
    port type TMTCPortType uses TMTCInfaceType;
    port type EvtServicePortType uses EvtServiceIfaceType;
  };

  features {
    feature AOCS {
      port instances {
        external client TMTCPortType range 1 to 1;
        external client EvtServicePortType range 0 to 1;
      };
    };
    feature ThermalControl {
      port instances {
        external client TMTCPortType range 1 to 1;
        external client EvtServicePortType range 0 to 1;
      };
    };
    feature TMTCService {
      port instances {
        external server TMTCPortType range 1 to 1;
      };
    };
    feature EventService {
      port instances {
        external server EvtServicePortType range 1 to 1;
      };
    };
    feature Services {
       port instances {
         internal client TMTCPortType range 1 to 1;
         internal client EvtServicePortType range 0 to 1;
         external server TMTCPortType range 1 to 1;
         external server EvtServicePortType range 0 to 1;
       };
       subfeature groups {
         group [ TMTCService ] range 1 to 1;
         group [ EventService ] range 0 to 1;
       };
    };
    feature Applications {
      port instances {
        internal server TMTCPortType range 2 to 2;
        internal server EvtServicePortType range 0 to 2;
        external client TMTCPortType range 2 to 2;
        external client EvtServicePortType range 0 to 2;
      };
      subfeature groups {
        group [ AOCS ] range 1 to 1;
        group [ ThermalControl ] range 1 to 1;
      };
    };
  };

};
```

```
features {
...
  feature TMTCService {
    port instances {
      external server TMTCPortType range 1 to 1;
    };
  };
  feature EventService {
    port instances {
      external server EvtServicePortType range 1 to 1;
    };
  };
  feature Services {
     port instances {
       internal client TMTCPortType range 1 to 1;
       internal client EvtServicePortType range 0 to 1;
       external server TMTCPortType range 1 to 1;
       external server EvtServicePortType range 0 to 1;
     };
     subfeature groups {
       group [ TMTCService ] range 1 to 1;
       group [ EventService ] range 0 to 1;
     };
  };
...
};
```

Component-based feature modelling

Feature realisations

PUSTMTCService feature realisation

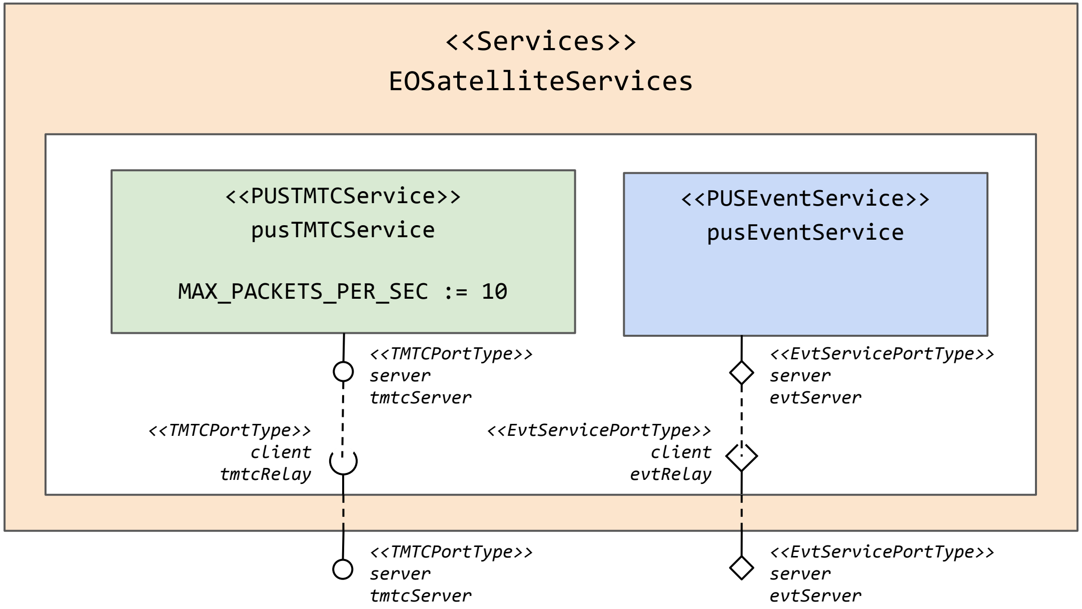
```
realisation PUSTMTCService of TMTCService {
    attributes {
        integer MAX_PACKETS_PER_SEC;
    };
    ports {
        external server TMTCPortType tmtcServer;
    };
};
```

EOSServices feature realisation

```
realisation EOSatelliteServices of Services {
  ports {
    internal client TMTCPortType tmtcRelay;
    external server TMTCPortType tmtcServer;
    internal client EvtServicePortType evtRelay;
    external server EvtServicePortType evtServer;
  };
  subfeature configurations {
    configuration PUSTMTCService pusTMTCService {
      MAX_PACKETS_PER_SEC := 10;
    };
    configuration PUSEventService pusEventService { };
  };
  connections {
    connection this.tmtcRelay <-> this.tmtcServer;
    connection this.evtRelay <-> this.evtServer;
    connection this.tmtcRelay <->
        pusTMTCService.tmtcServer;
    connection this.evtRelay <->
        pusEventService.evtServer;
  };
};
```

Component-based feature modelling

Feature realisations

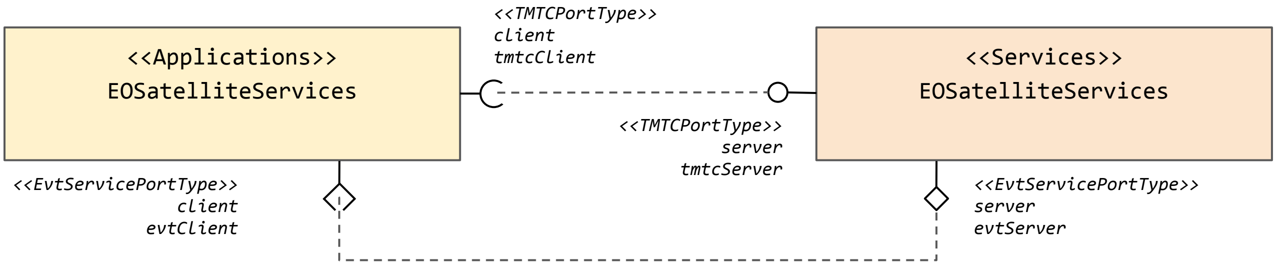


Component-based feature modelling

Product configurations

EOSatellite Product

```
product EOSatellite {
  configurations {
    configuration EOSApplications eosApplications { };
    configuration EOSatelliteServices eosServices { };
  };
  connections {
    connection eosApplications.tmtcClient <-> eosServices.tmtcServer;
    connection eosApplications.evtClient <-> eosServices.evtServer;
  };
};
```



Conclusions and future work

- An approach to feature modelling based on the use of constructs from the component-based software development domain has been introduced
  - It allows establishing features hierarchies, making a clear distinction between the feature them- selves and their realisations or variants
  - It enables the definition of complex dependency relationships between feature realisation
- The approach allows the modelling of product configurations as a set of interconnected and configured feature realisations

Future goal

To define a model-based software product line of on-board satellite applications that uses as inputs the feature models defined in this approach

Thank you very much for your attention

Any questions?