AC-CONTRACT: RUN-TIME VERIFICATION OF CONTEXT-AWARE APPLICATIONS

Marina Mongiello[1],Patrizio Pelliccione[2], Massimo Sciancalepore[1]

[1] Dipartimento di Ingegneria Elettrica e dell'Informazione – Politecnico di Bari, Italy

[2] Department of Computer Science and Engineering - Chalmers University of Technology and University of Gothenburg, Sweden

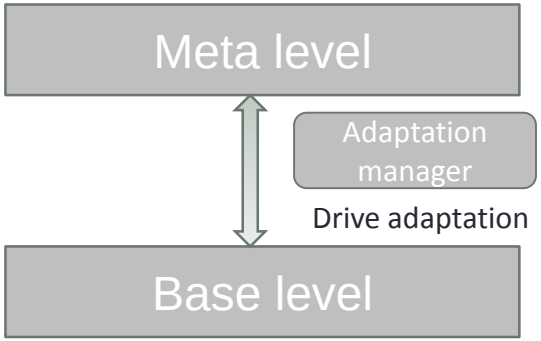
Outline of the talk

- Schema and frame: cognitive psicology
- AdaptableCode-contract: AC-contract
- Instantiation of AC-contract
- Derivation of operational requirements
- Reflection and contract checking
- Android implementation: Traveller
- Conclusion and future work

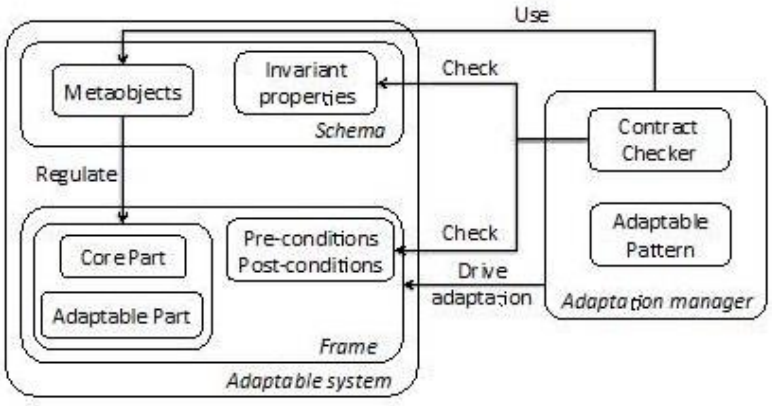
AC-contract: formal definition

Tuple

- Schema
- Frame
- Adaptable Pattern
- Contract checking



AC-contract tuple

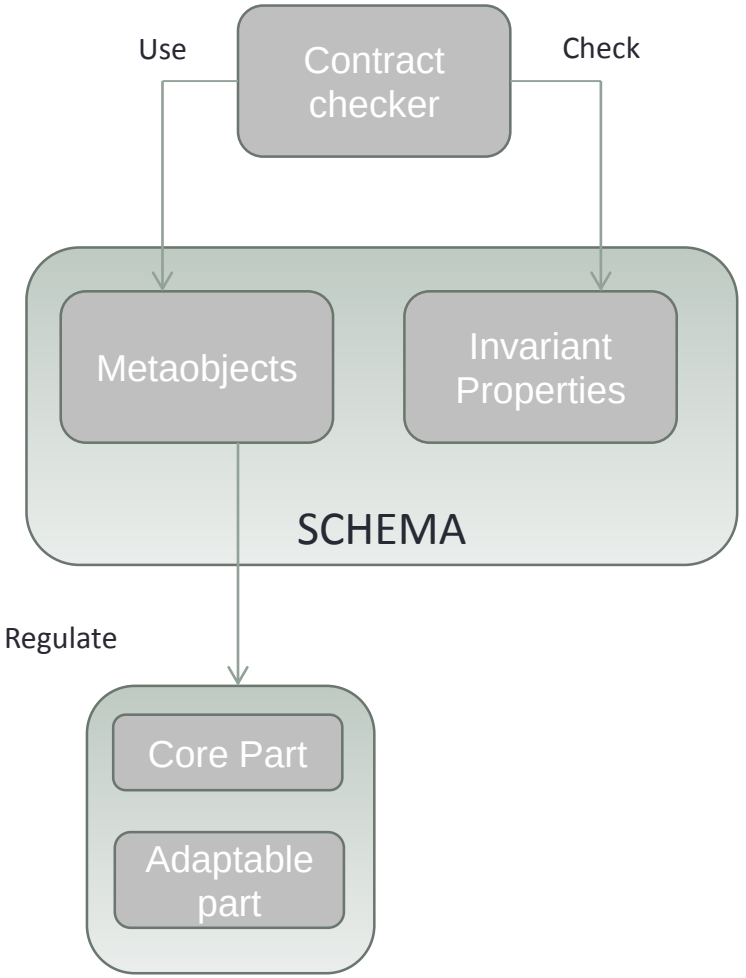


Embeds logical propositions in the source code

Executes the annotation for run-time verification

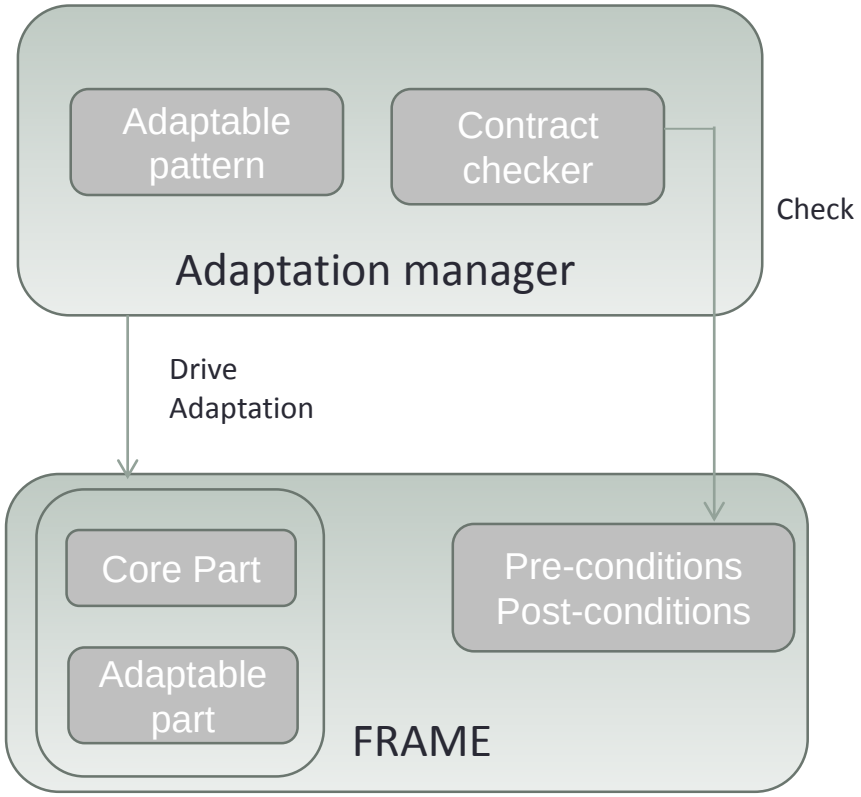
Schema

- Models the structure of information
  - Tacit knowledge to use to interpret ambiguous situations
  - Models properties to maintain despite adaptation



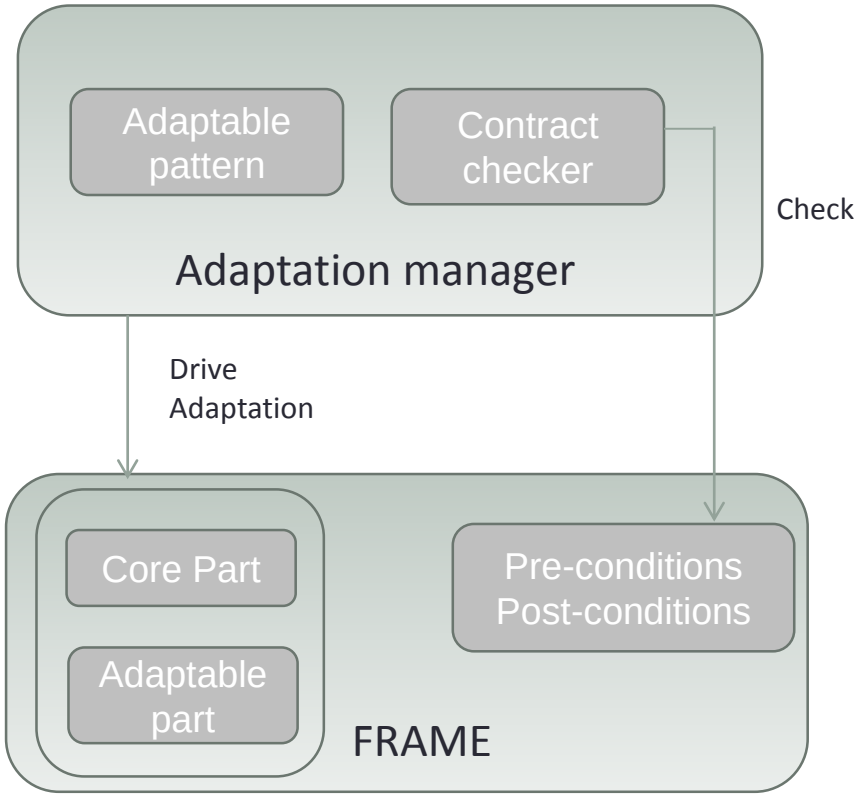
Frame

- The set of core and adaptable components
- Contains the *pre* and *post conditions* of the contract



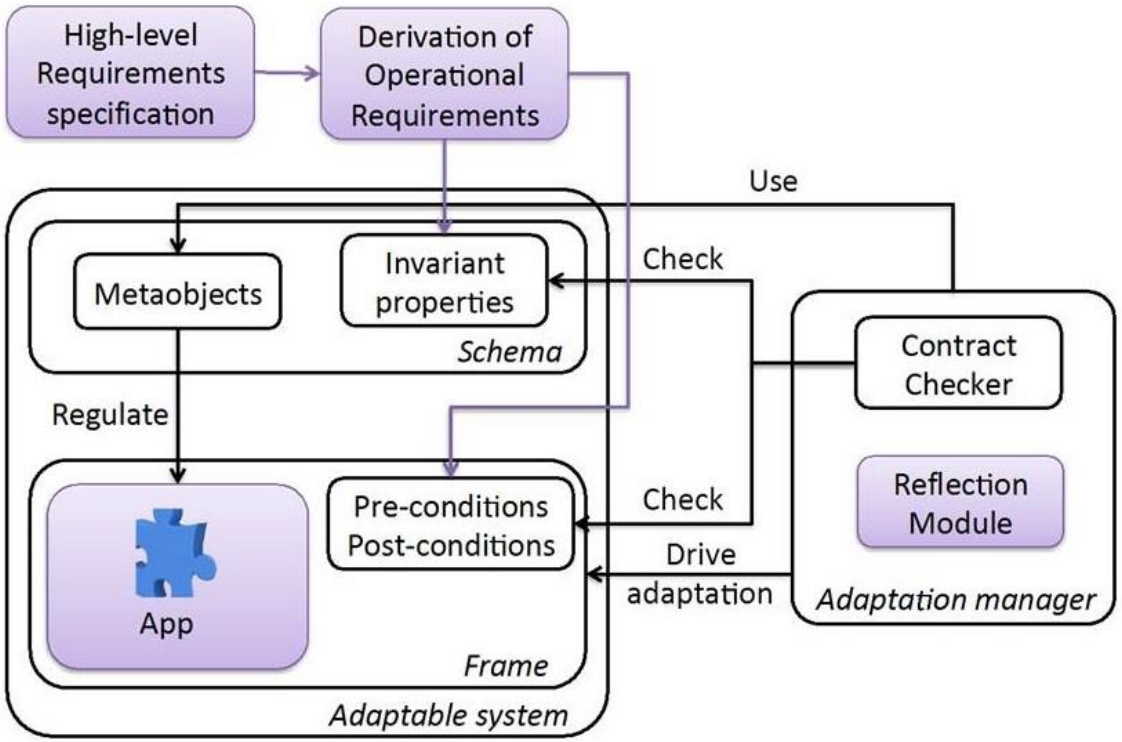
Adaptation manager

- Embeds the contract checker managing metaobjects at runtime
- Contains the adaptable mechanism to manage metaobjects



Instantiated approach

Instantiation of AC-contract based on user's requeriments specification



Requirements

High-level requirements

- Expressed according to structured english grammar in term of specification patterns

Operational requirements

- Formalized exploiting the structure of specification patterns

Derivation of operational requirements

- $\mathcal{E}$={Events/states}: Local properties
- SP: Specification patterns
  - Absence(P)
  - Universality(P)
  - BoundedExistence(P,n)
  - Response(P,S)
  - Precedence(P,S)
  - $\text{ResponseChain}_{1N}$(P,S,$T_1$,..$T_N$)
  - $\text{ResponseChain}_{N1}$(S,$T_1$,..$T_N$,P)
  - $\text{PrecedenceChain}_{1N}$ (S, T_1,..T_N,P)
  - $\text{PrecedenceChain}_{N1}$ (P,S,T_1,..T_N)
- SC: Scope
  - Globally
  - Before(R)
  - After(Q)
  - Between(Q,R)
  - After(Q,R)

Requirement specification

- $req_{spec} = (inst_{sc}(sc, loc_{prop}) , inst_{sp}(sp, loc_{prop}))$
  - $inst_{sc}(sc, loc_{prop})$
    - maps a scope into a list of local properties
    - instantiate the scope according to state/events in loc_{prop}
  - $inst_{sp}(sp, loc_{prop})$
    - maps a pattern into a list of local properties
    - instantiates the pattern according to states/events in loc_{prop}

Requirement specification: an example

High-level requirement

*"**The system creates an album with the name of signalled points** of interest only when **the multimedia files have been already stored**"*

Operational requirement

- $\mathcal{E}=\{e_1,e_2\}$
  - e_1= *"the system creates an album with the name of signalled points of interest"*
  - e_2= *"the multimedia files have been already stored"*
- *SC: Globally*
- *SP: Precedence (P,S)*

Structured english grammar for req

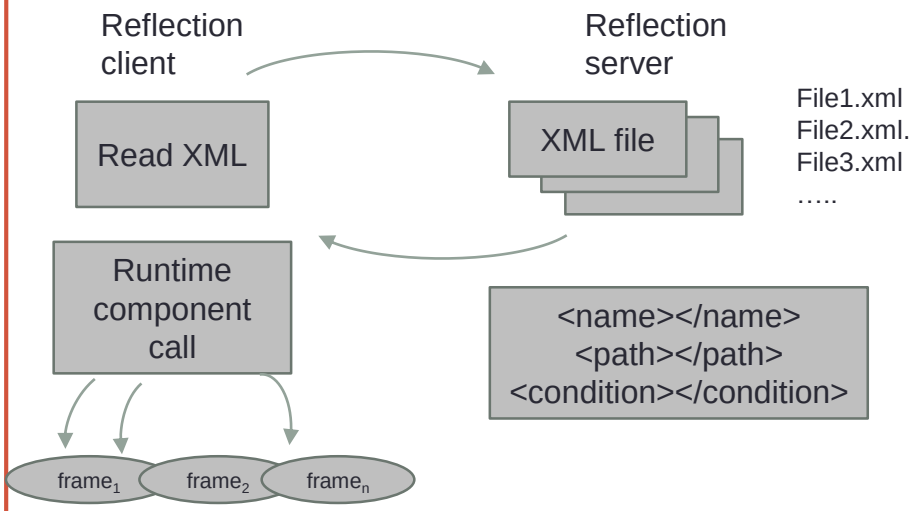
- **Globally, if** *'the system creates an album with the name of signalled points of interest"* **then it must have been the case that** *"the multimedia files have been already stored* **before** *'the system creates an album with the name of signalled points of interest "*

Reflection Module

Main features

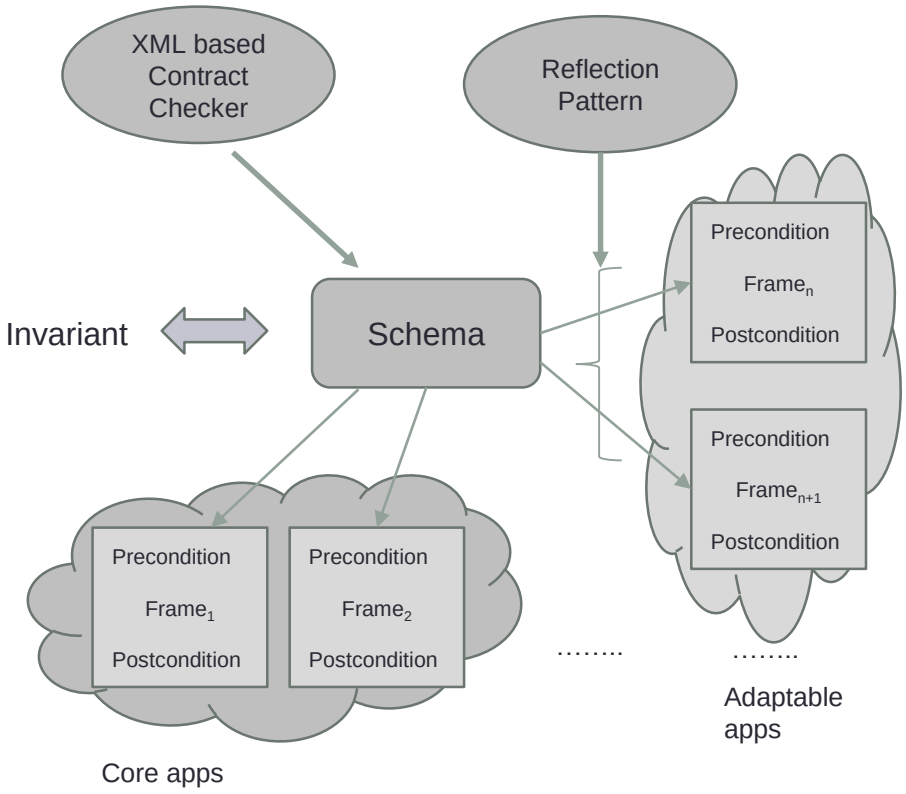
- Implements Reflection pattern
- Deploys and executes adaptable components
- Uses metainformation about the extensions

Control flow of reflection module



Contract checking module

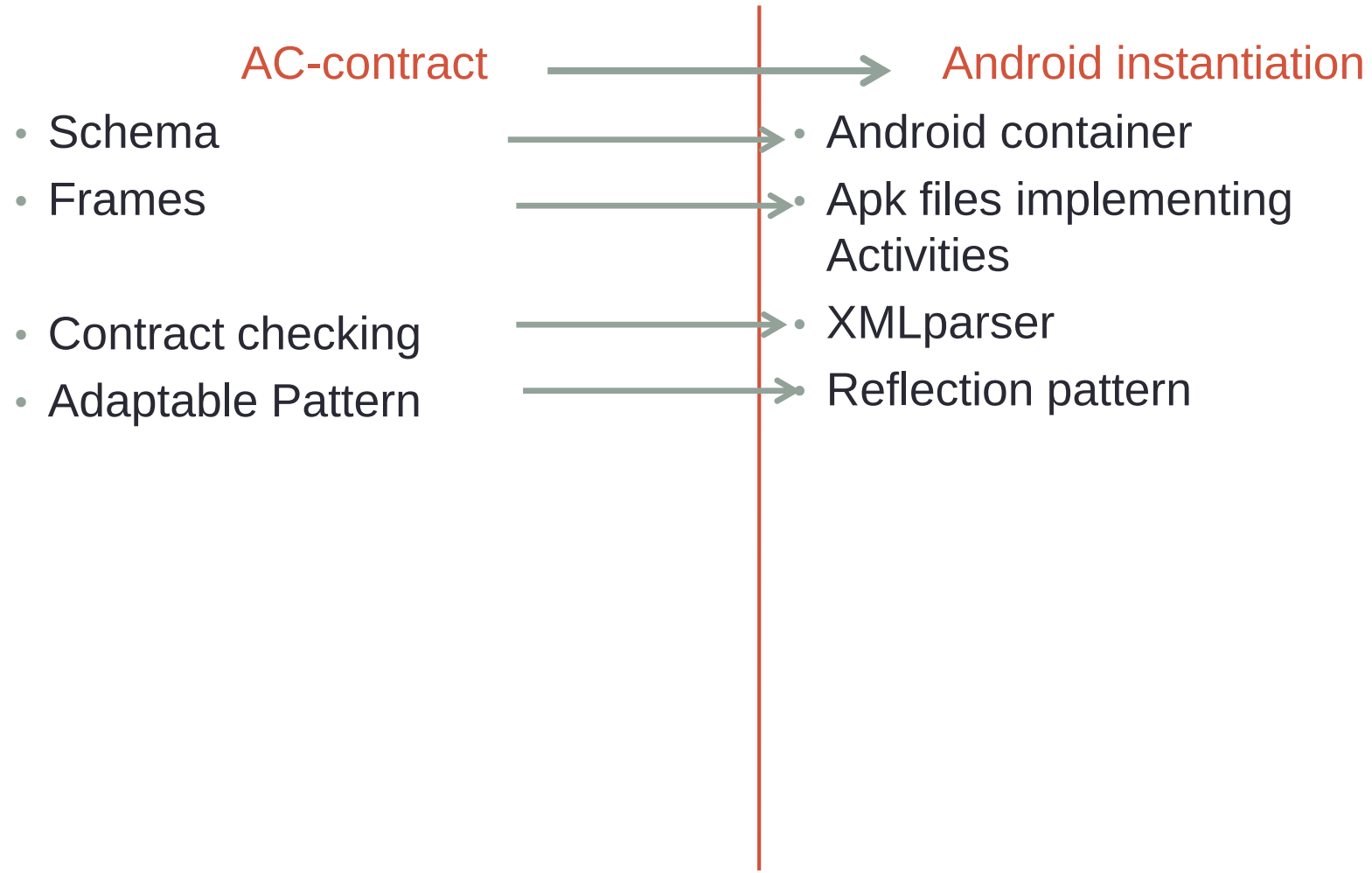
- Checks the contract triple
- Core components of the schema embed the invariant of the contract
- Each component encodes pre/postcondition pair



Mobile Adaptable architecture

- Traveller: Android application for traveller assistance
  - Architectural issues:
    - Native/hybrid mobile architecture
    - Dynamic deployment
    - Executable code downloaded and executed on the device
  - Innovative issues:
    - Implementation of contract verification on Android platform
    - Use of android Intents to manage events pre and post conditions.

Traveller


AC-contract
Android instantiation
Schema
Android container
Frames
Apk files implementing Activities
Contract checking
XMLparser
Adaptable Pattern
Reflection pattern

Derivation of operational requirements

- **Globally, if** *‘’the system creates an album with the name of signalled points of interest”* **then it must have been the case that** *”the multimedia files have been already stored* **before** *‘’the system creates an album with the name of signalled points of interest “*

- precondition
  - P= *‘’the system creates an album with the name of signalled points of interest”*
- postcondition
  - S= *’the multimedia files have been already stored”*

Precondition P

- P is managed using Android Intent: image capture
- Precondition P is split in:
  - The user wants to take a photo
  - The photo is taken
  - The user accepts the photo
- Low level precondition encoding :
  - Event Photo ← onClick()
  - setAction (i←Intent(Action_Image_capture))
  - startActivity(i,Capture_active_request_code)

Precondition check

- User activates the app
- If some event occurs that satisfies a requirement
  - Precondition is checked to find an app satisfying the requirement
  - The app is downloaded and executed by Reflection on the device

Postcondition S

- Is encoded in the deployed app and checked after app execution
- Low level postcondition encoding:
  - the number of stored multimedia files on the device has increased
  - Implemented using a user defined function to count the multimedia files stored on the device’s memory

Postcondition check

- S is checked after the app execution:
  - The album with multimedia data has been created
  - S is verified if creation of album has been performed

AC-contract reasoning algorithm

App activation and execution

- Life-cycle of android app: run from Activity
- Intent activates Events:
  - A component requires the execution of an Action by another component

Algorithm AC-Contract

- begin
  - Intent Definition
    - Event Photo
    - setAction(Intent)
    - StartActivity(Capture_image)
    - If result
      - If XML ContractChecking Then
        - DownloadAPK
        - ReflectionCall
        - CreateAlbum
        - CheckPostCondition
- End

Contract checking

XML file descriptor managing

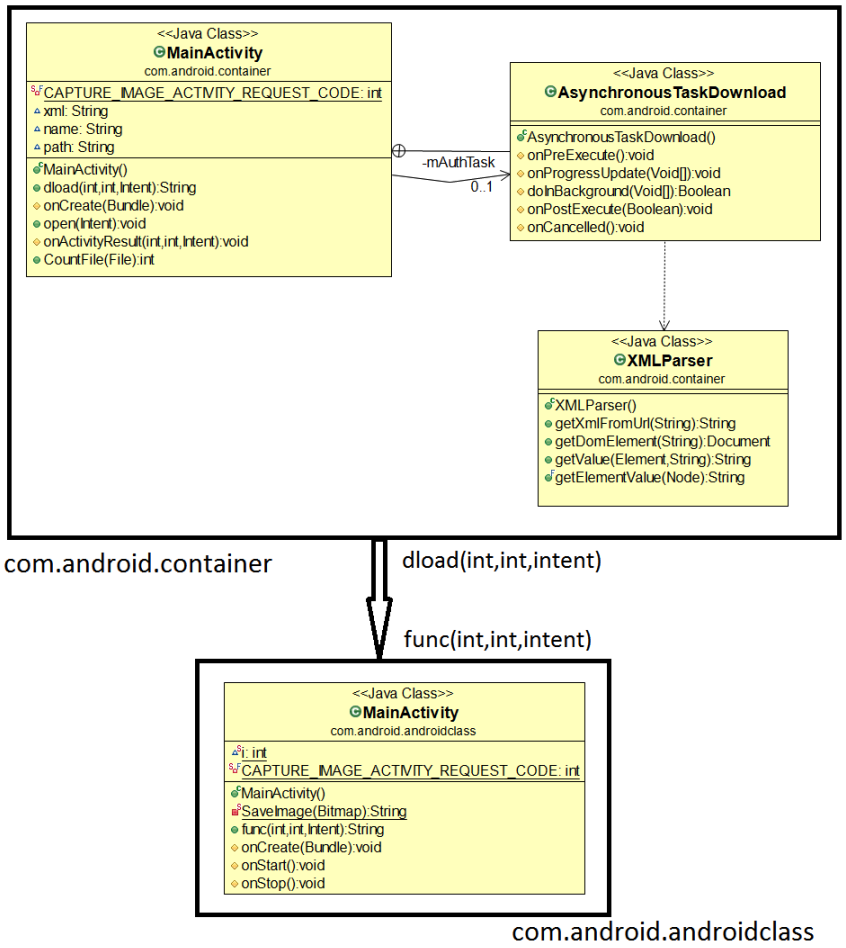
- Contract checking is managed through an XML file descriptor:
  - Each app that can be invoked by the main container has a XML file descriptor
  - A FTP server stores the XML file descriptors

Contract checking algorithm

Begin
XMLParser
While nextFile XML do
getXML fromURL
For i=1..nodelistlength
Getitem
Precondition← getvalue
If CheckPrecondition then
getnameclassTag
getpathnameTag
Download

end

Reflection Module



- Reflection enable the call of methods belonging to different Android classes
- Container connects to the address of the app
- Address is returned from the XML parser
- App is deployed to device container and executed on the device

Conclusion

- We propose AC-contract:
  - A run-time verification approach for modeling and verifying run-time requirements of adaptable software systems
  - Based on:
    - Design-by-contract
    - Reflection pattern
  - Models operational requirements using
    - Specification patterns
    - Local properties
- Starting from high-level requirements identifies properties that locally hold on single parts of the system
- Methodology is validated on a mobile application

Future work

- We are currently working to extend the approach on the theoretical basis and on experimental application:
  - Extend the approach for compositional and incremental verification
  - Instantiate the approach in implementative platform:
    - Sensor networks
    - Internet of things