# Tasking Framework in Nutshell: Motivation and Basic Features

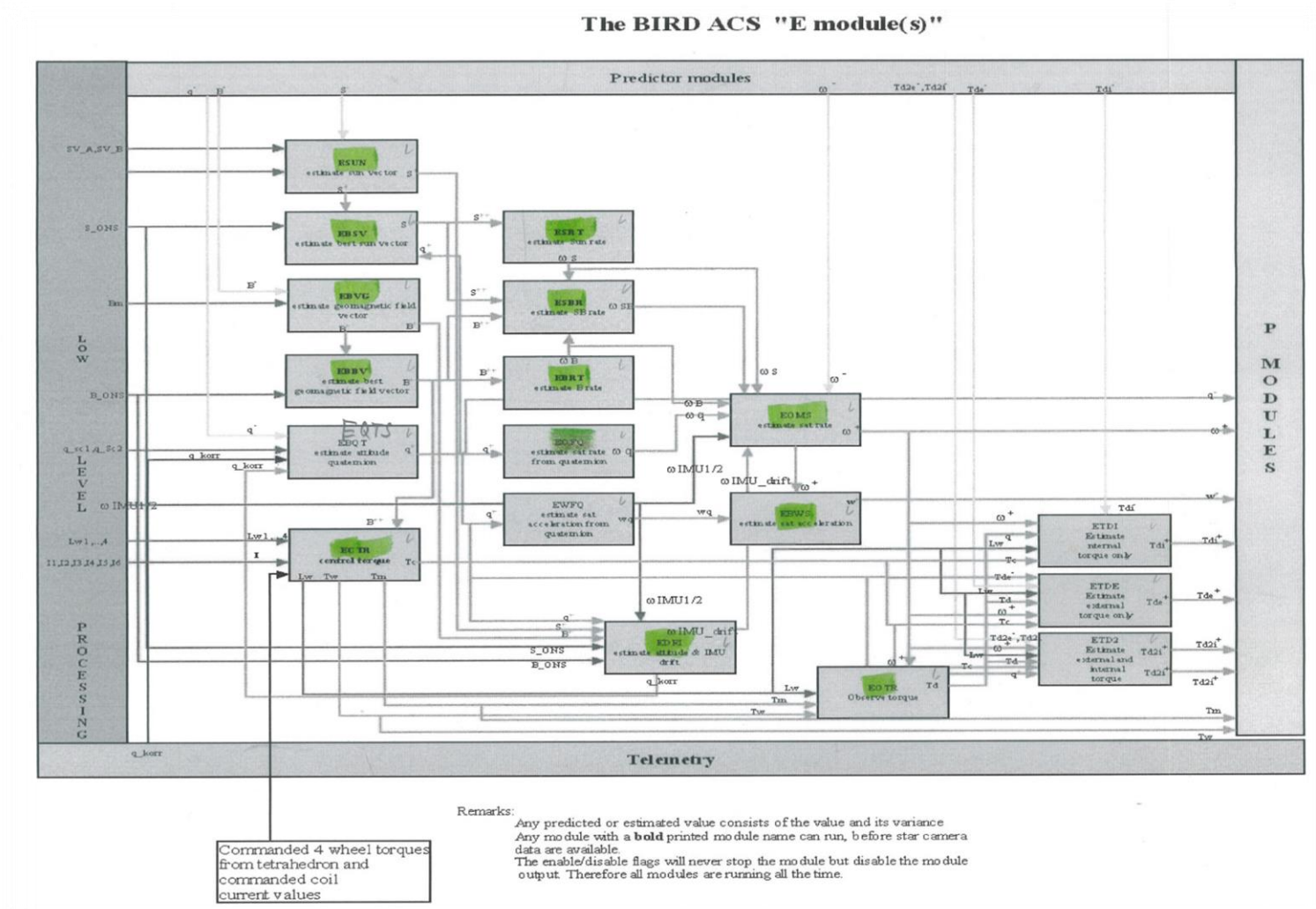
Embedded Systems Week 2020

Olaf Maibaum

# Small Satellite BIRD (Bisprectal Infrared Detection)

- BIRD
  - Mass 94 kg
  - Size ~ 1x1x1 m
  - Launch 22.10.2001
  - Monitoring of high temperature events

- Attitude Control System
  - 3 axis stabilized
  - 9 sensors, 5 different types
  - 6 actuators, two different types
  - Distributed Kalman Filter
    - 14 Estimator Modules
    - 4 Predictor Modules



# Lessons learned BIRD Attitude Control System

- Controller use a predefined time schedule
- Hard real-time condition on reaction wheel commanding
- Complex data flow in Distributed Kalman Filter
- Slowest sensor determine start time of estimation, prediction and state controller
- Not enough computing time for:
  - Computation of high fidelity attitude state
  - Attitude state machine
  - Control torque computation
  - Reaction wheel command message preparation
- Deadline misses at times with high work load from other satellite bus applications
- Endless loops in thread bodies a nightmare for testing

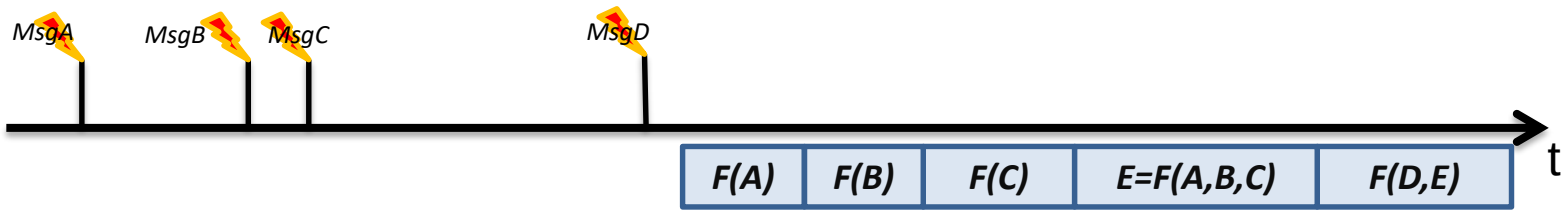
→ Need for ASAP scheduling in Distributed Kalman Filter

→ No endless loops for computing tasks

BIRD (© DLR)

# ASAP Scheduling

Scheduling in BIRD:

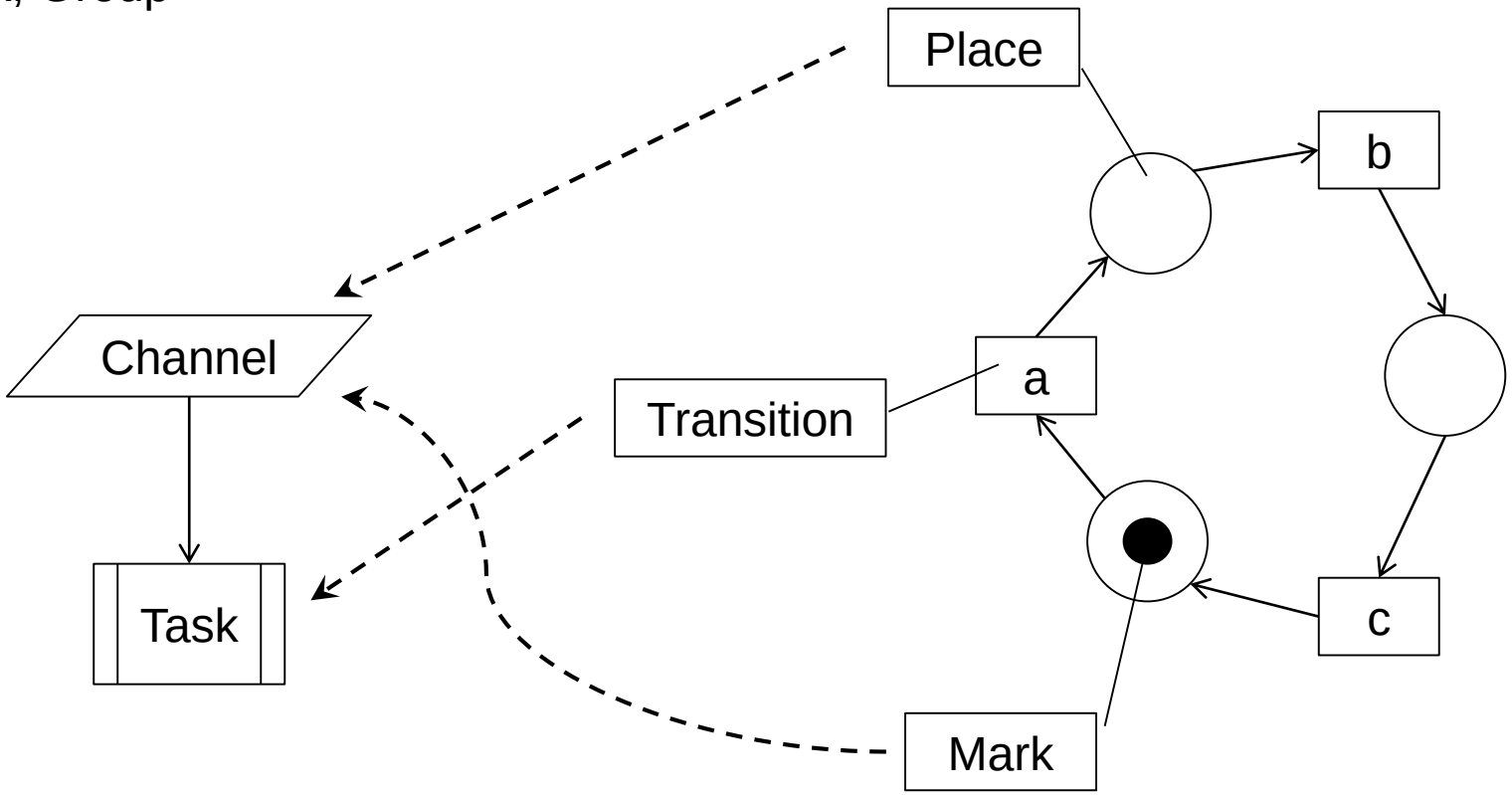



Tasks with ASAP scheduling:

# Time Line of the Tasking Framework: Prototype (~2008)

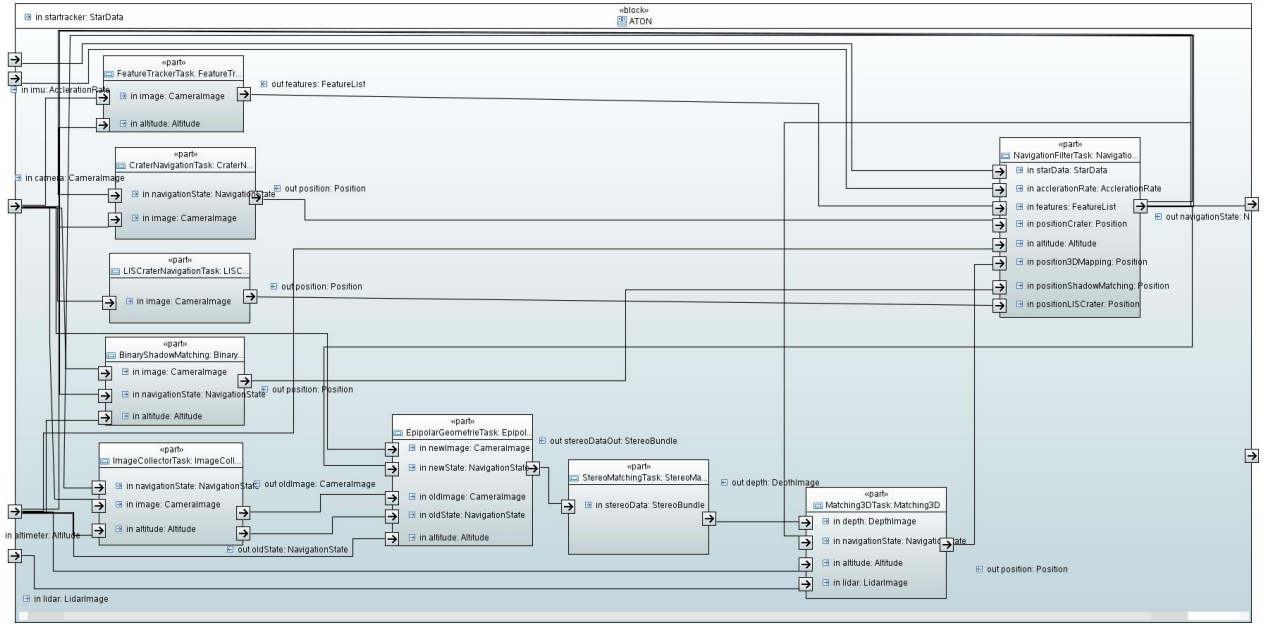
- Inspired by places (Channel) and transitions (Task) in Petri Nets

- Application SW overload Channel, Task, Group
- Class Channel control activations
- Global Scheduler
- Dynamic memory allocation



# Time Line of the Tasking Framework: ATON (2010)

- First application in ATON (Autonomous Terrain-based Optical Navigation)
- Simulation experiment
- First call from S-function in MatLab/Simulink up to integrated in helicopter controller
- Application SW overload Channel, and Task
- Global Scheduler
- Dynamic memory allocation

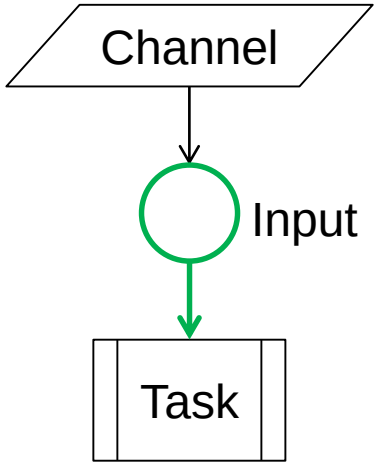



- Drawbacks
  - Collision between timing MatLab/Simulink and pthread-Executors
    - Solution: Step processing
  - All channel activations becomes optional by multiple receiver tasks
  - Default behavior of Input and running condition in Channel::reset

# Time Line of the Tasking Framework: Eu-CROPIS (2012)

- Application SW overload Channel, Event, Include, and Task
- Global scheduler with watchdog and reporting functionality based on Outpost
- Static memory allocation by memory provider class
- Specialized channels support PUS-Service 3 (Housekeeping)
- Specialized channels to support device specific functionalities
- Barriers introduced

- Drawbacks:
  - Global scheduler and unit tests by parallelity
    - Instrumentation of global scheduler for unit test support
    - Class UnitTestScheduler
      - to clean up global scheduler
      - Clock controlled by step processing
  - Memory provider class source of failures
  - Complexity to configure system by constructors

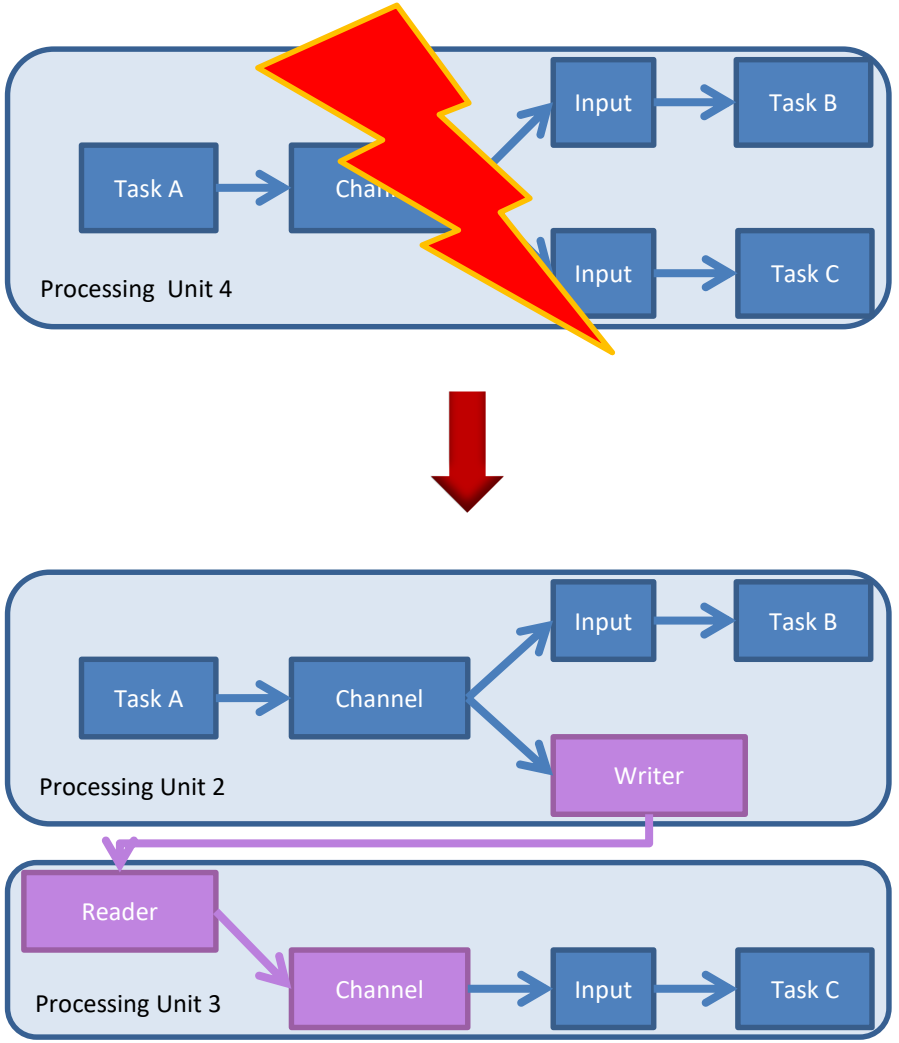


# Time Line of the Tasking Framework: OBC-NG (2013) / ScOSA (2016)

- Application on Distributed Systems
  - OBC-NG (On-Board Computer - Next Generation)
  - ScOSA (Scalable On-board Software Architecture)

- Application SW overload Channel, Include, Event, and Task
- Global scheduler on each computing node
- Static memory allocation by memory provider class

- Drawbacks:
  - Memory provider class source of failures
  - Complexity to configure system by constructors



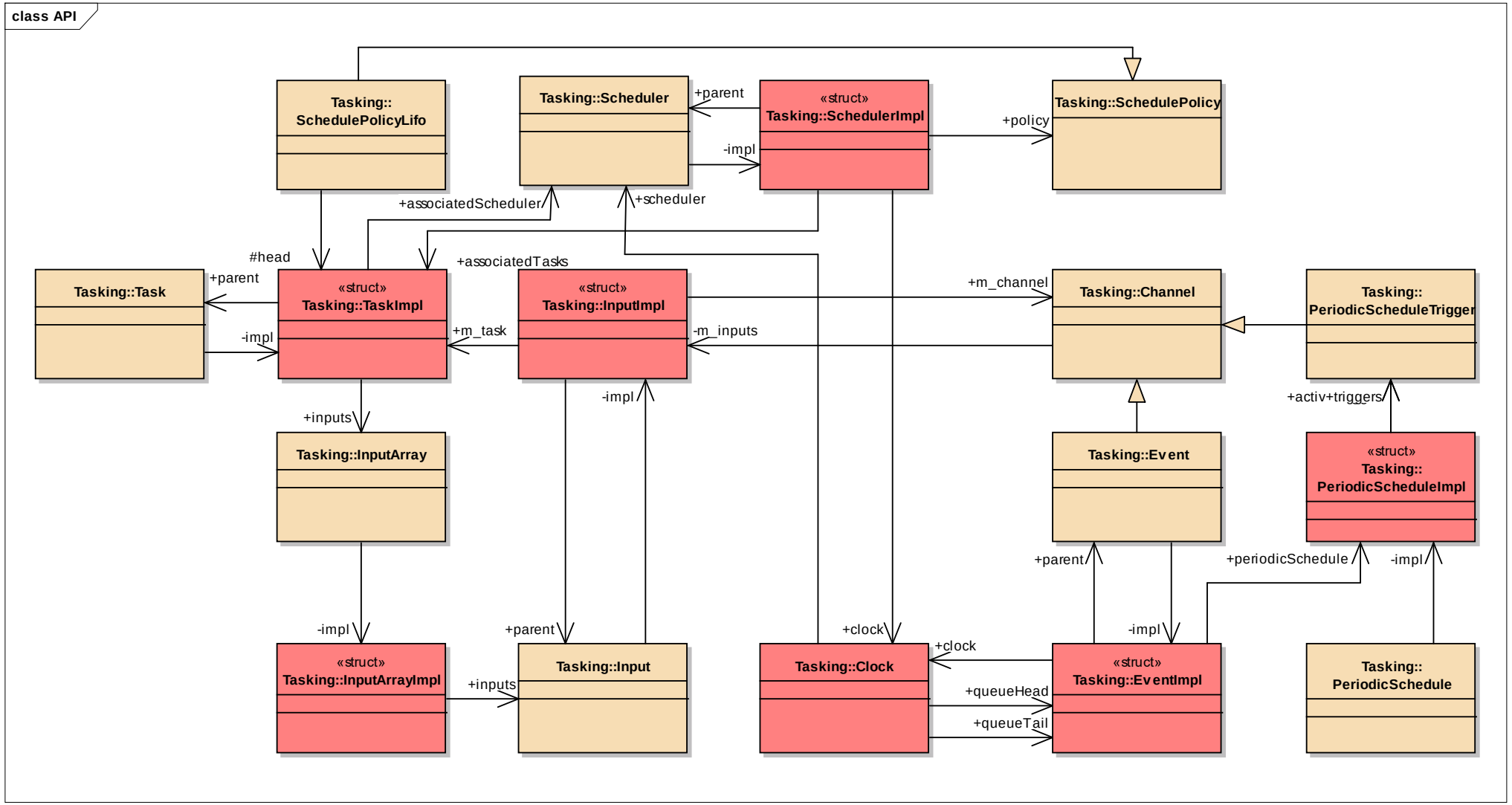
# Time Line of the Tasking Framework: Reorganization (2017-2019)

- Requirements:
  - No global objects
  - Simplify set-up process
  - Low complexity of programming for platform specific schedulers
  - Overtake further lessons learned from applications

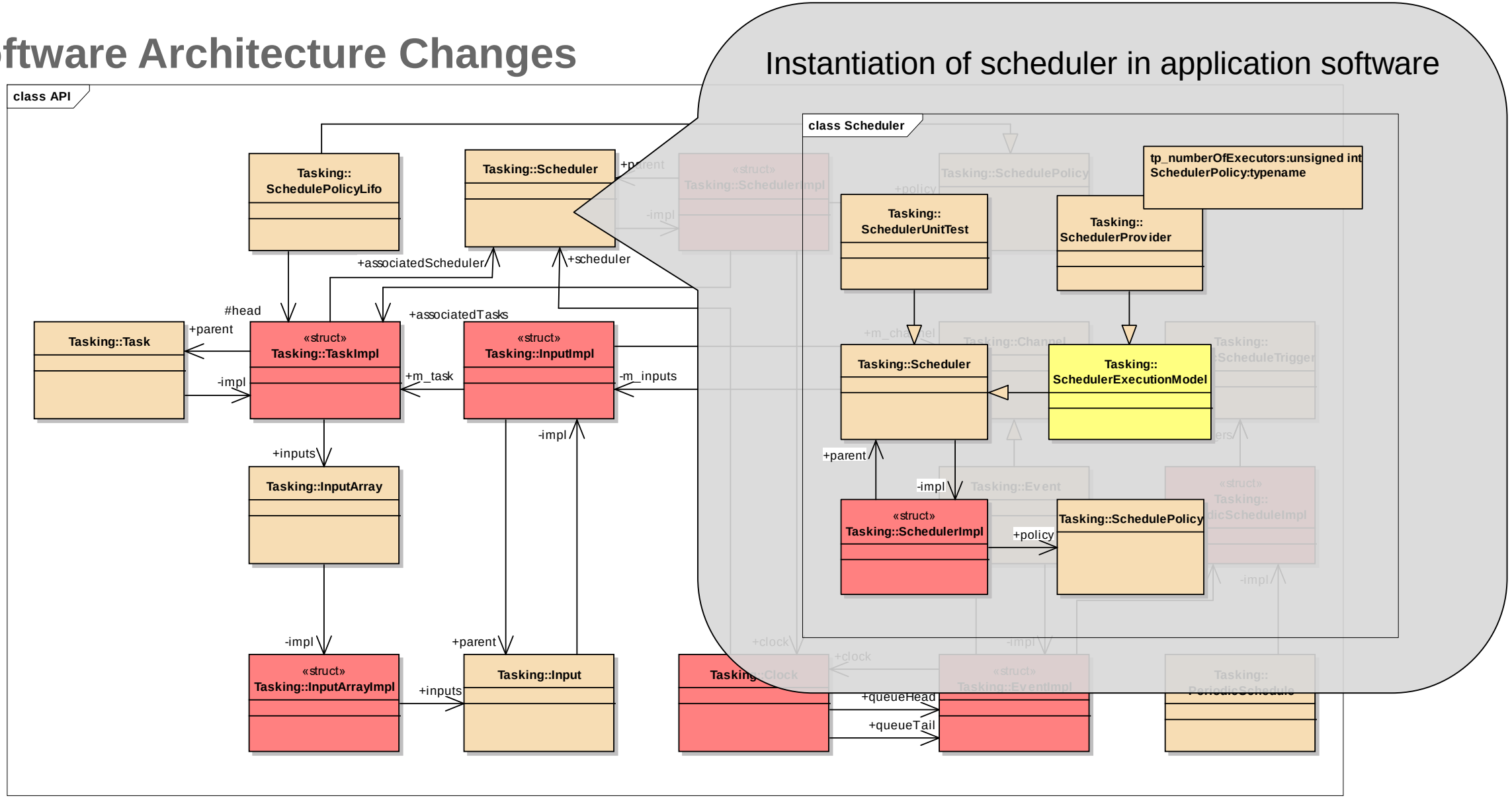
- By the requirements API needs an reorganization
  - Can not hold compatibility to old version



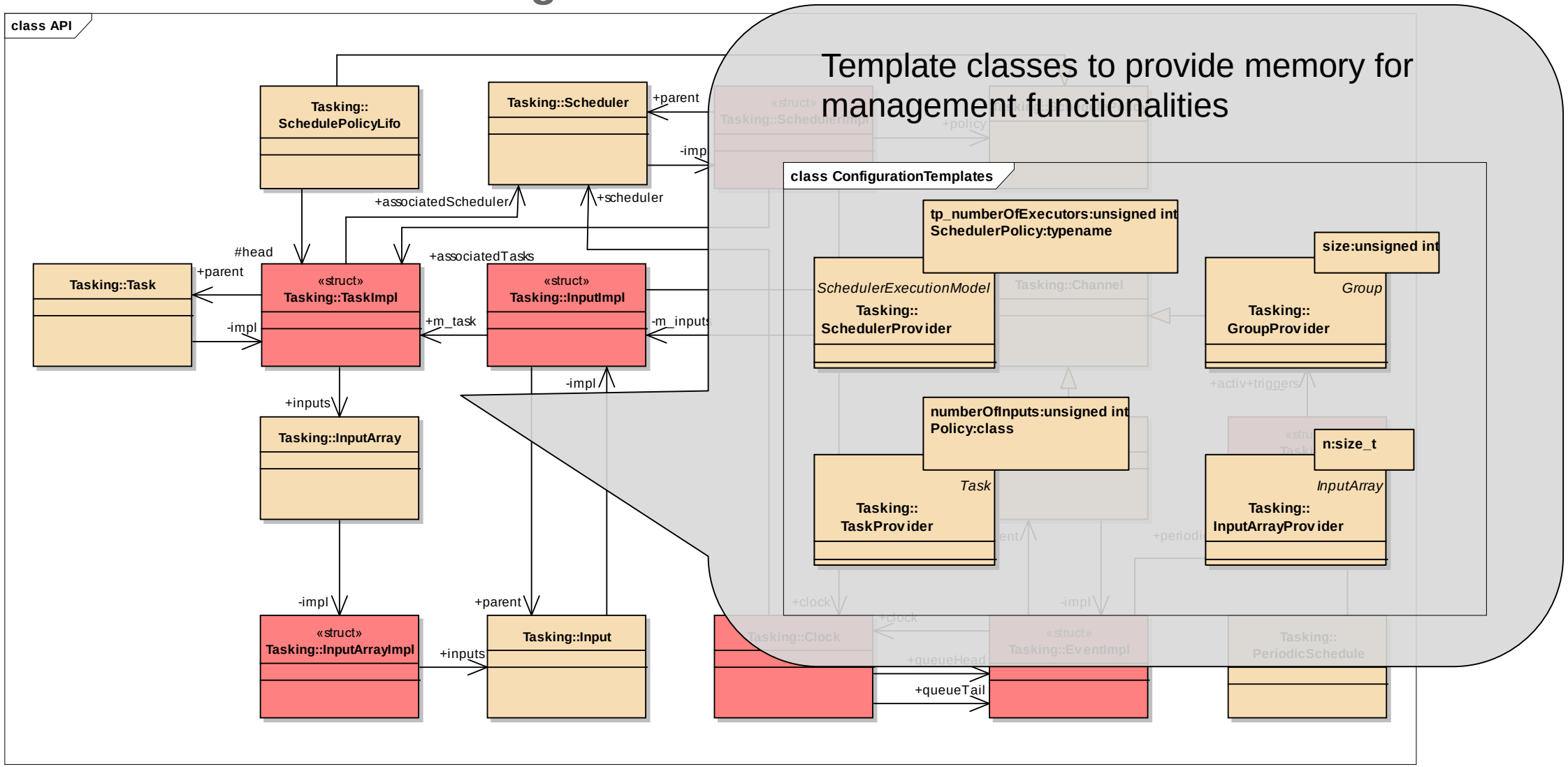
# Software Architecture



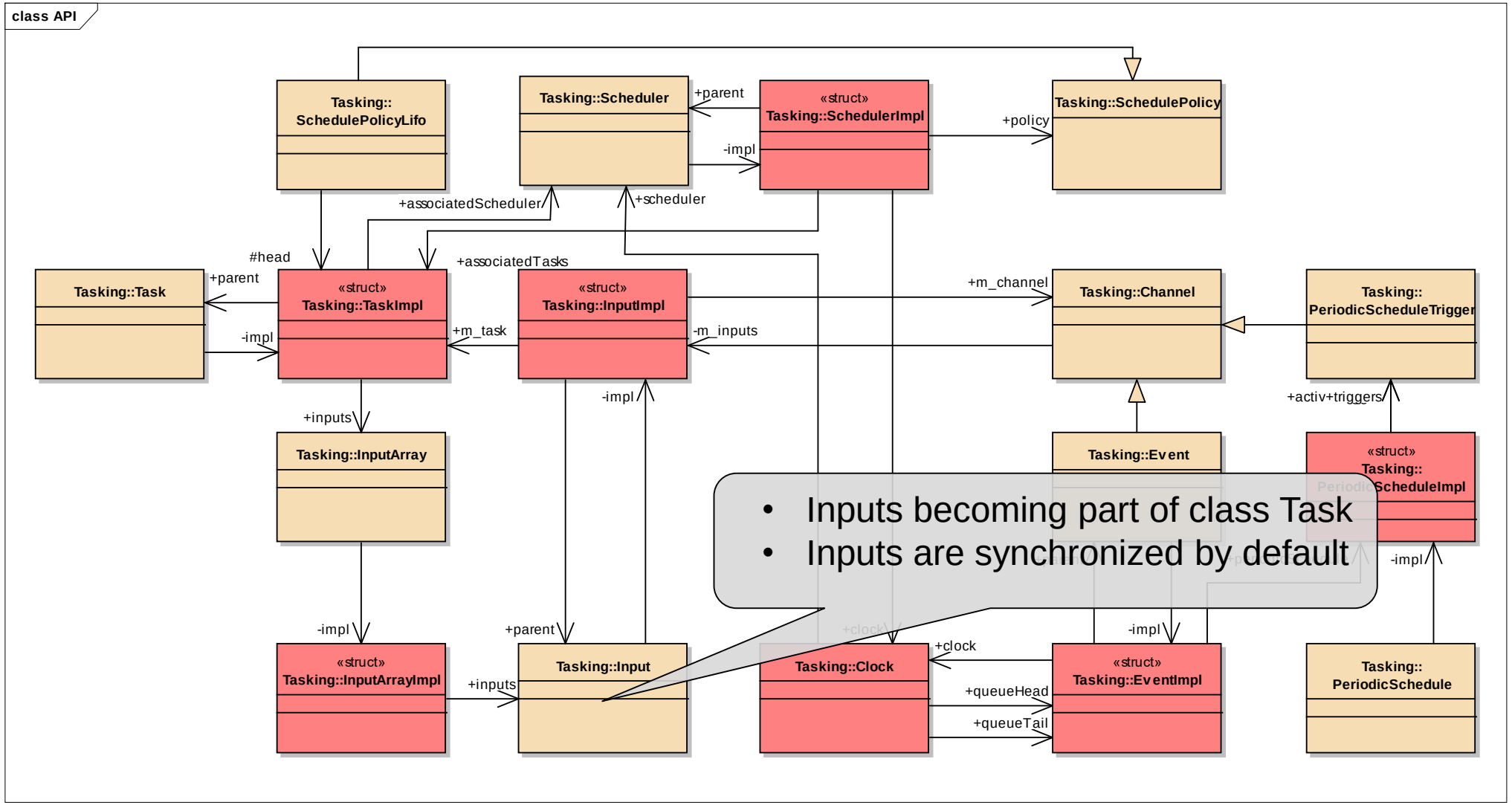
# Software Architecture Changes



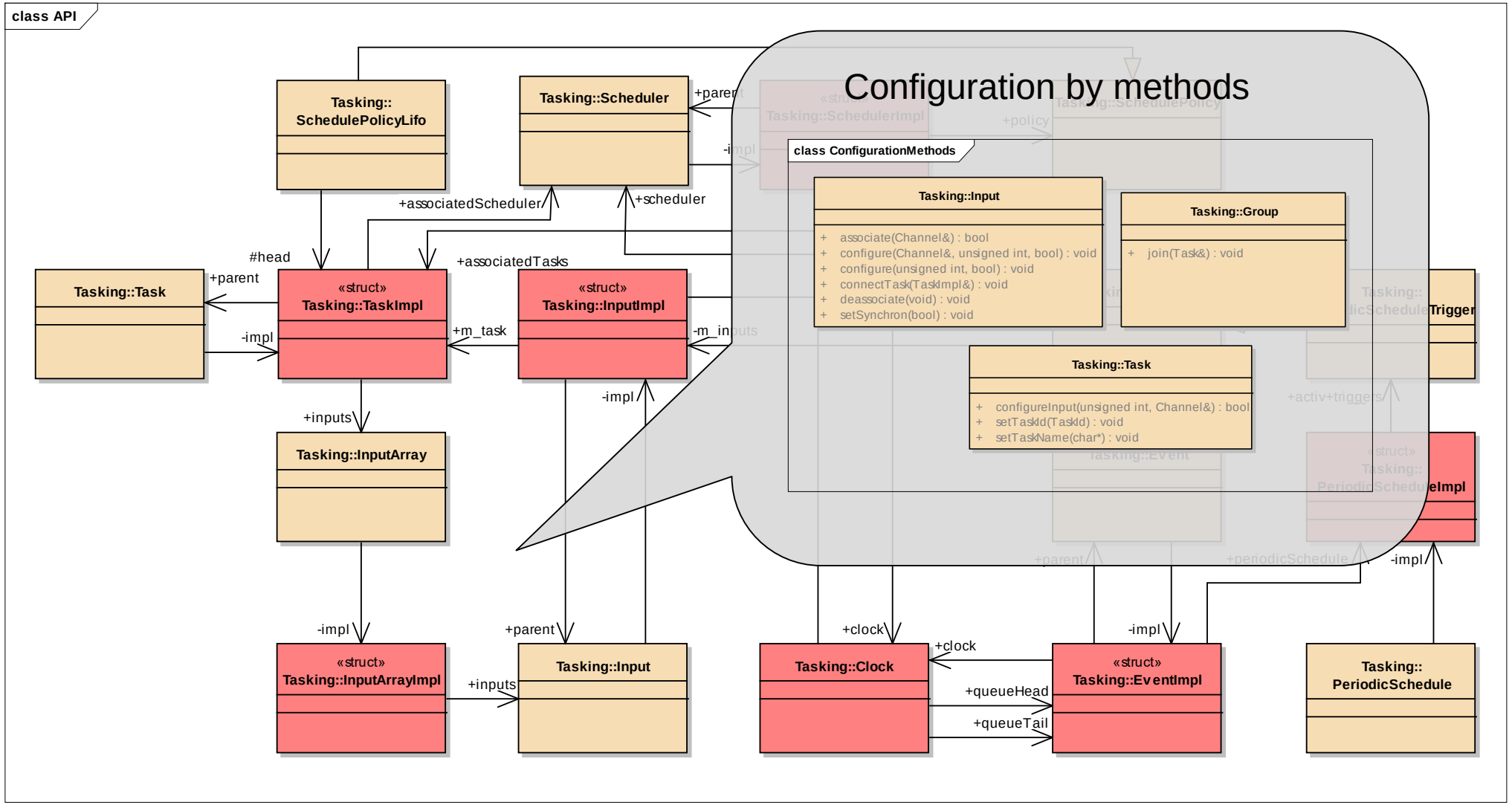
# Software Architecture Changes



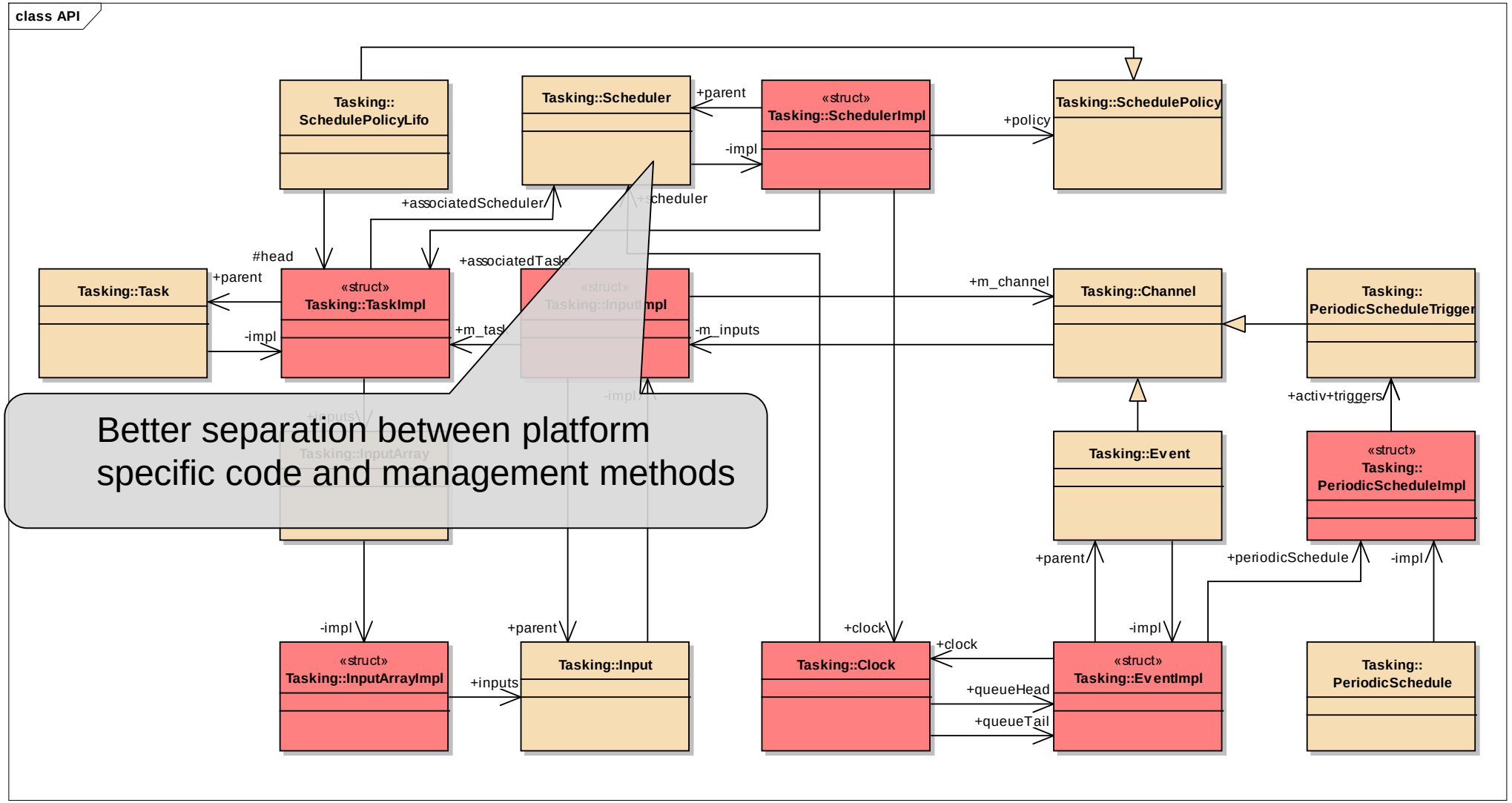
# Software Architecture Changes



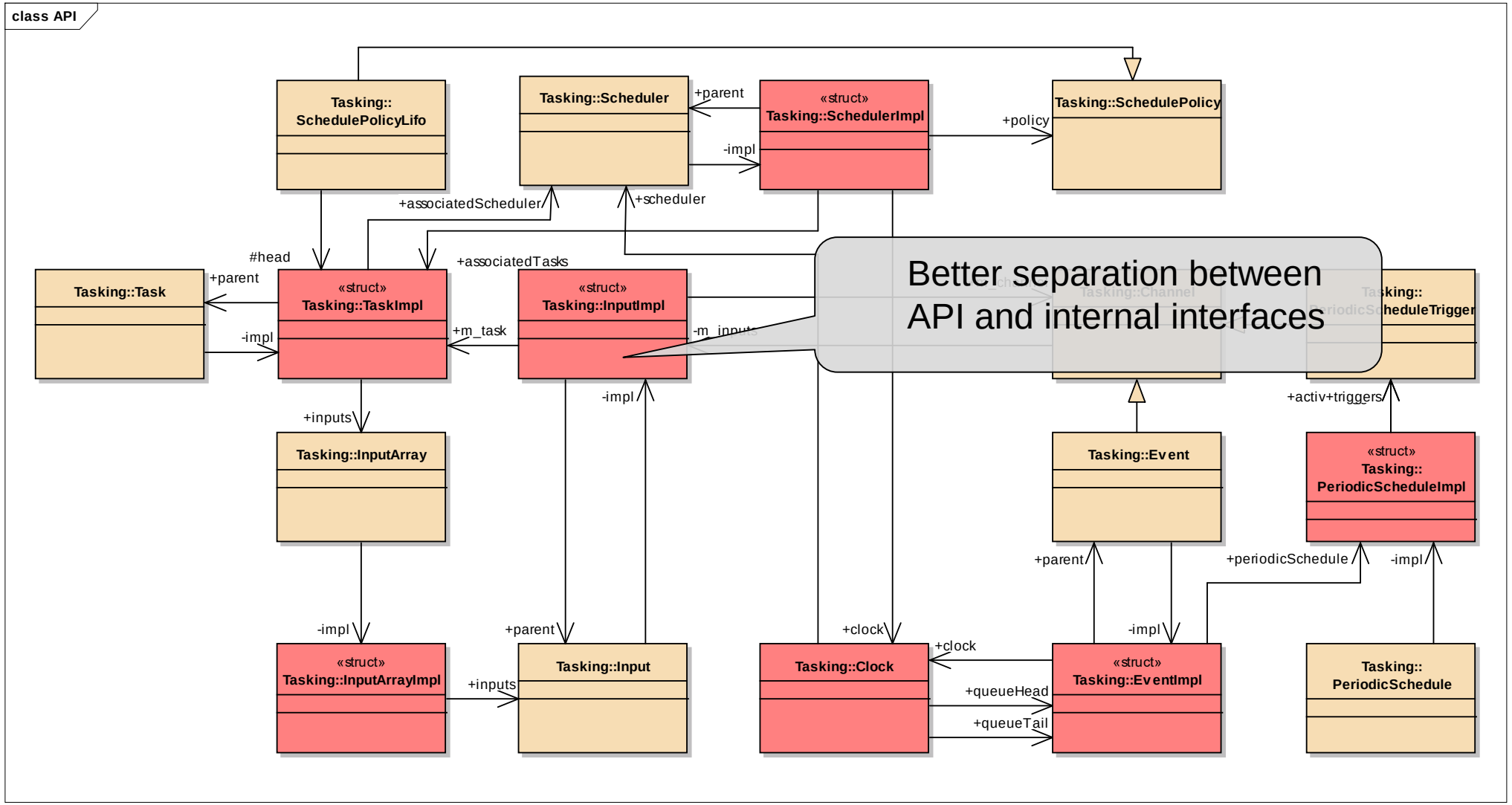
# Software Architecture Changes



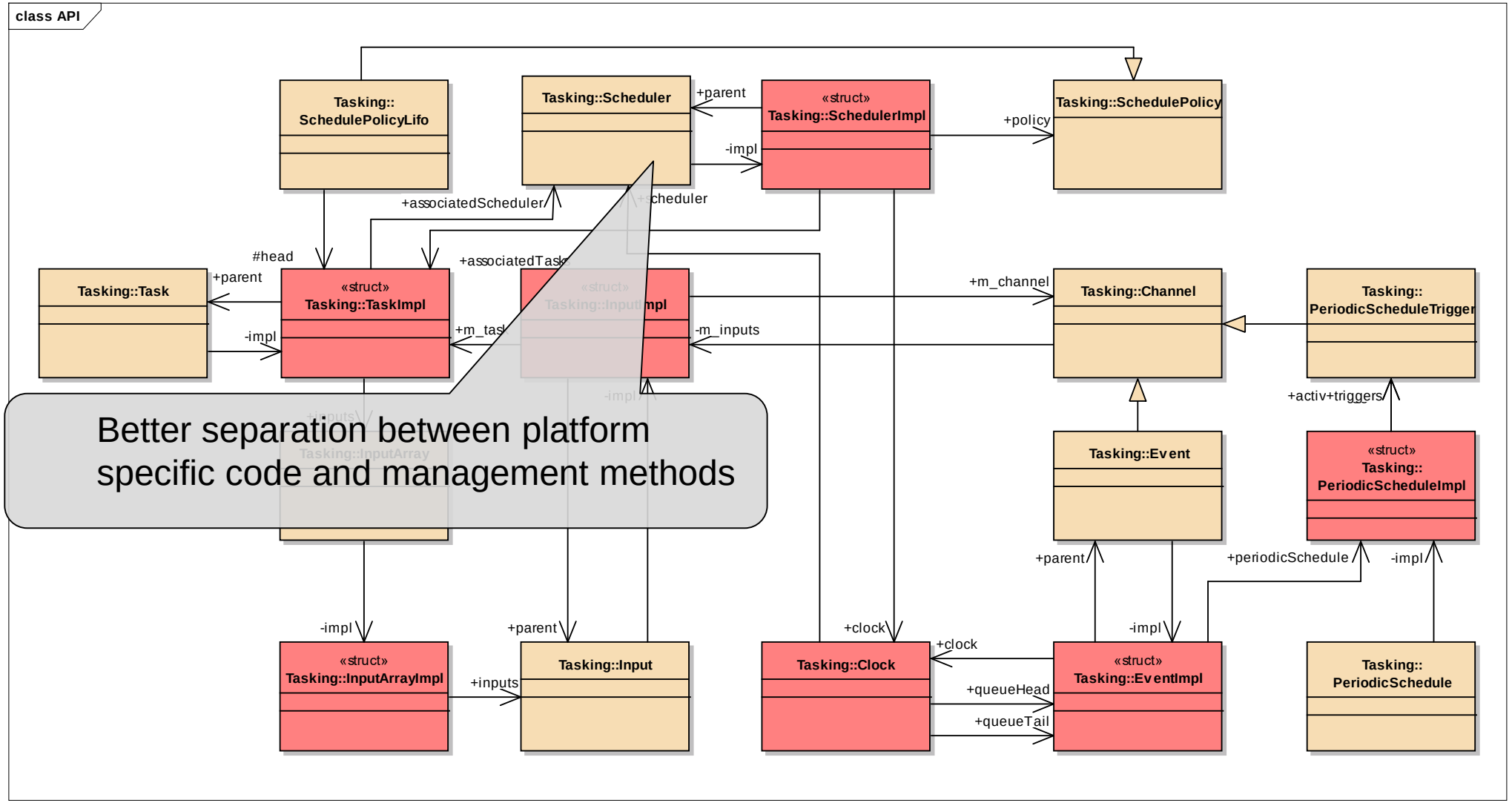
# Software Architecture Changes



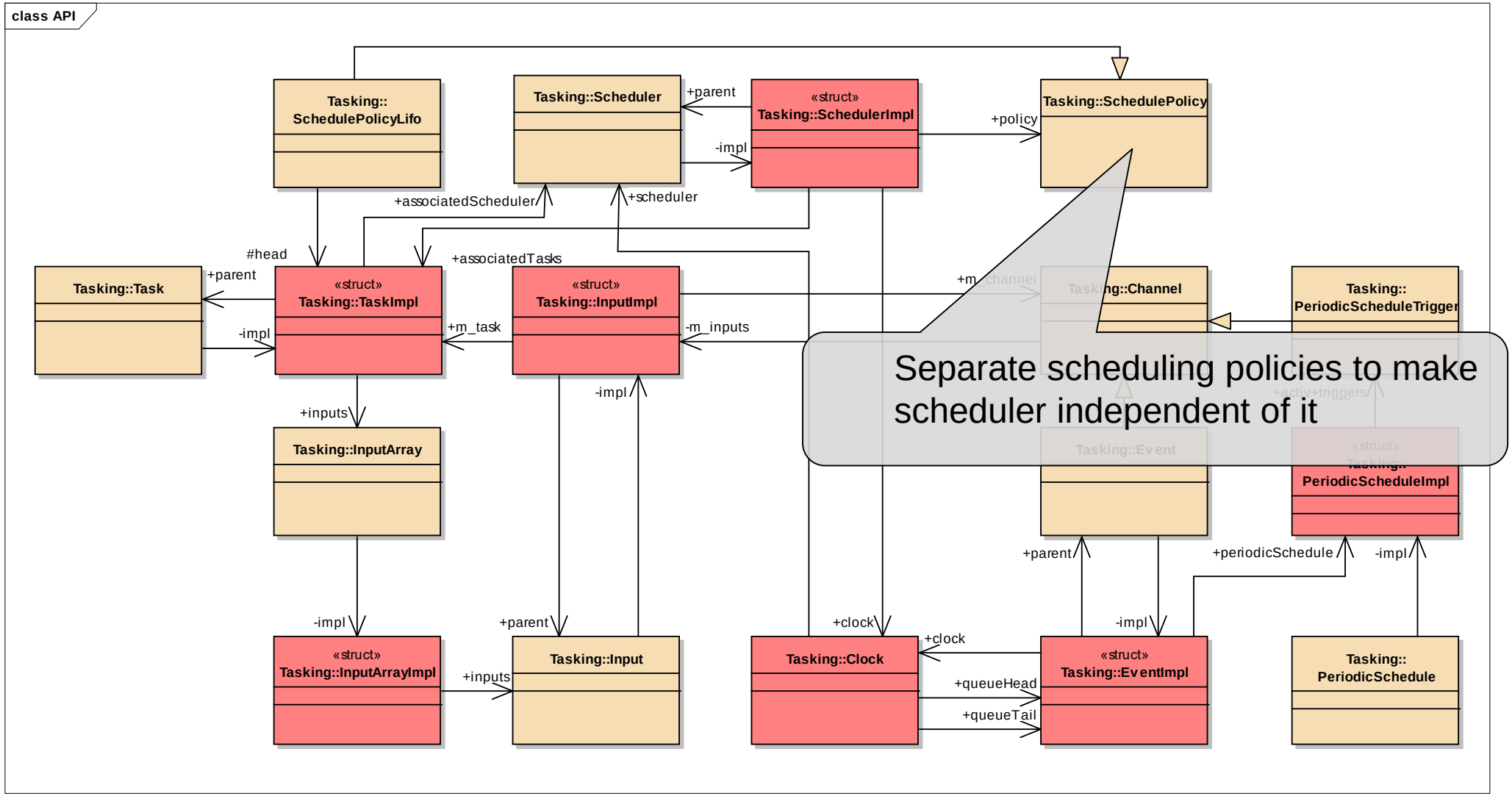
# Software Architecture Changes



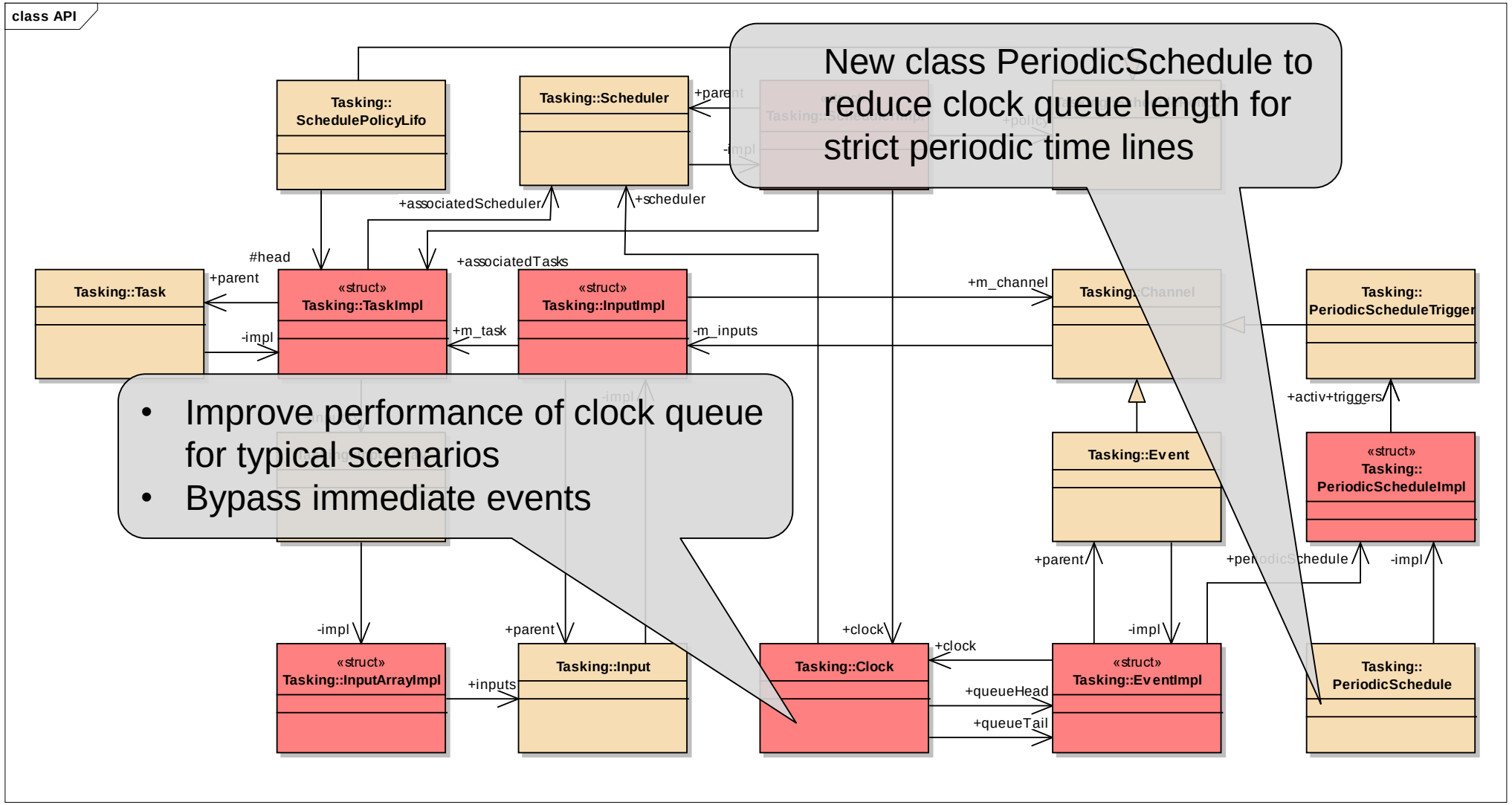
# Software Architecture Changes



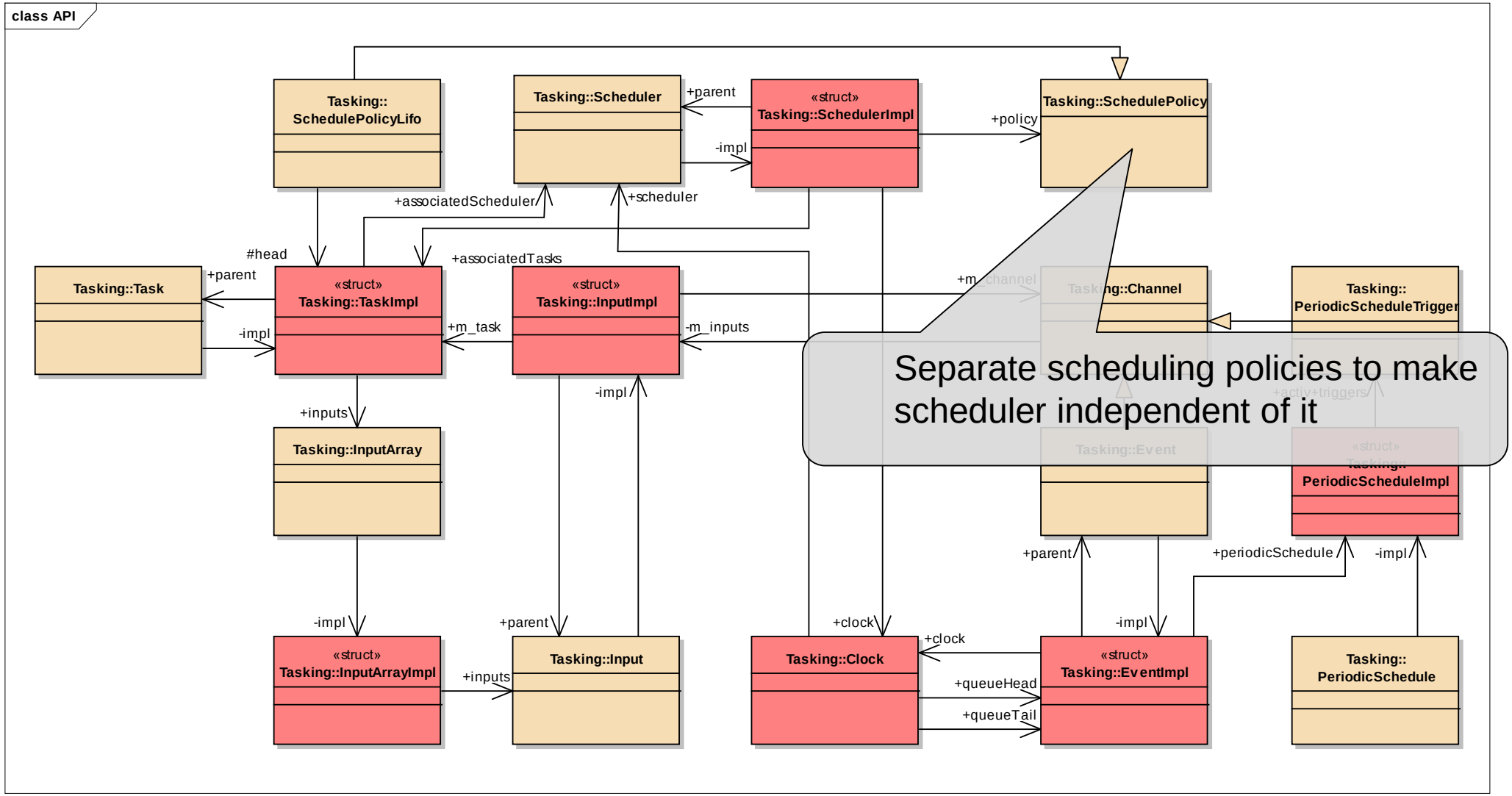
# Software Architecture Changes



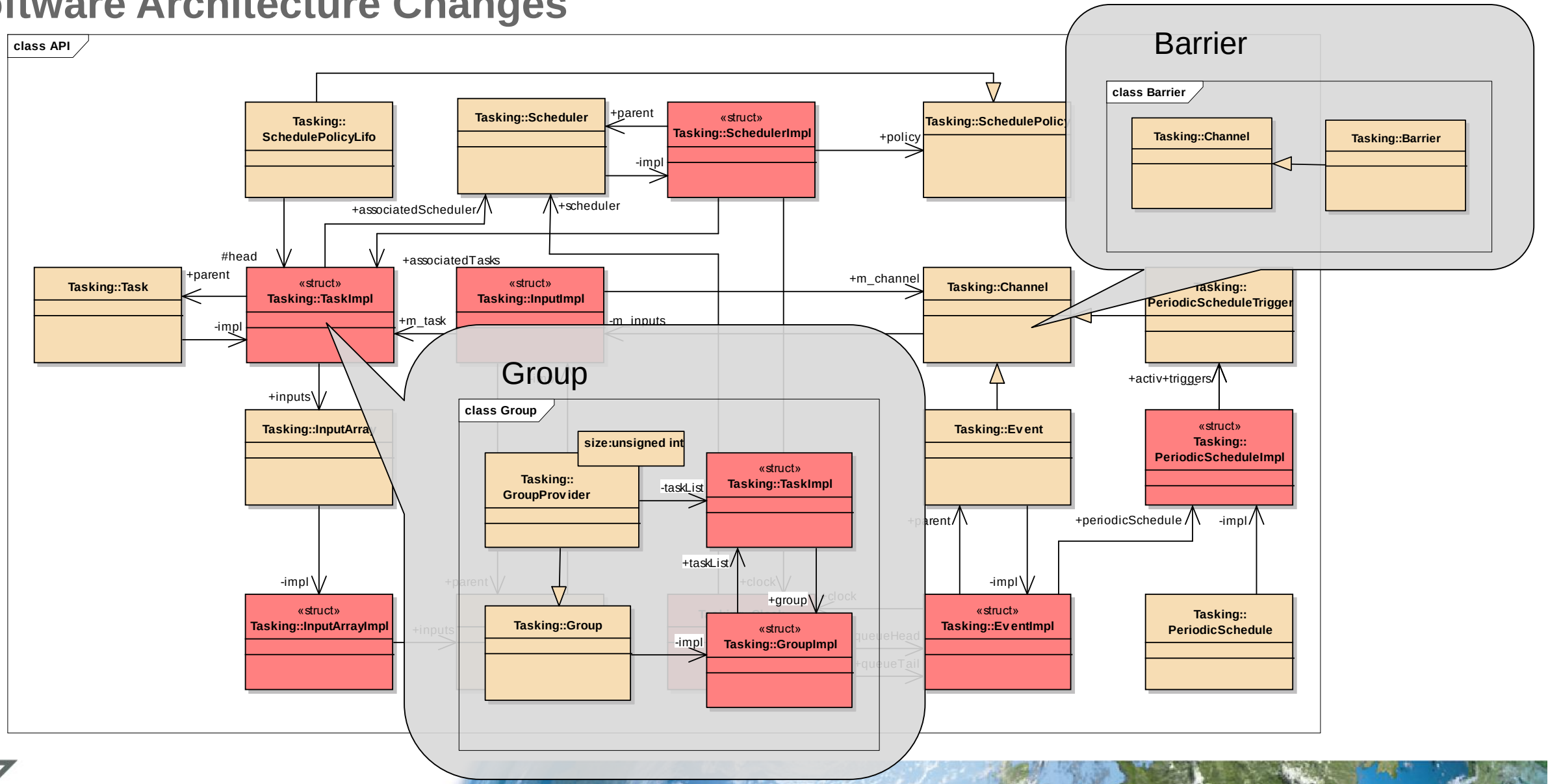
# Software Architecture Changes



# Software Architecture Changes



# Software Architecture Changes



# Live Q&A session

Sunday 20.09.2020
at 11:00 – 12:00 am EDT
at 05:00-06:00 pm Berlin

11:00am

**Tutorial 4: Tasking Framework: An open-source software development library for on-board software systems**

11:00am - 12:00pm, Sep 20

Tutorials

---

* This tutorial is **a live Q&A session** with **pre-recorded videos made available before the event**.

Tasking Framework is a C++ software development library and an event-driven multithreading execution platform. It is developed by the Institute for Software Technology, German Aerospace Center (DLR). Tasking Framework is dedicated to improve the reusability in developing embedded software systems and to reconcile the embedded software with model-driven software development. It can be used to develop, but not dedicated for, critical as well as non-critical embedded software on single-core as well as parallel architectures. Tasking Framework gives software developers the ability to implement their applications as task graphs with arbitrary activation patterns (periodic, aperiodic and sporadic) using a set of abstract classes with virtual methods. It is compatible with the POSIX-based operating systems, mainly Linux and RTEMS. The Tasking Framework was successfully used in, for instance, the attitude orbit

Read more