

## API test approach for video processing systems

- ☑ In this paper, an approach followed to test API based video processing system is presented
- ☑ The approach is a combination of test methods and elicitation of implicit requirements based on fault categories in video processing systems

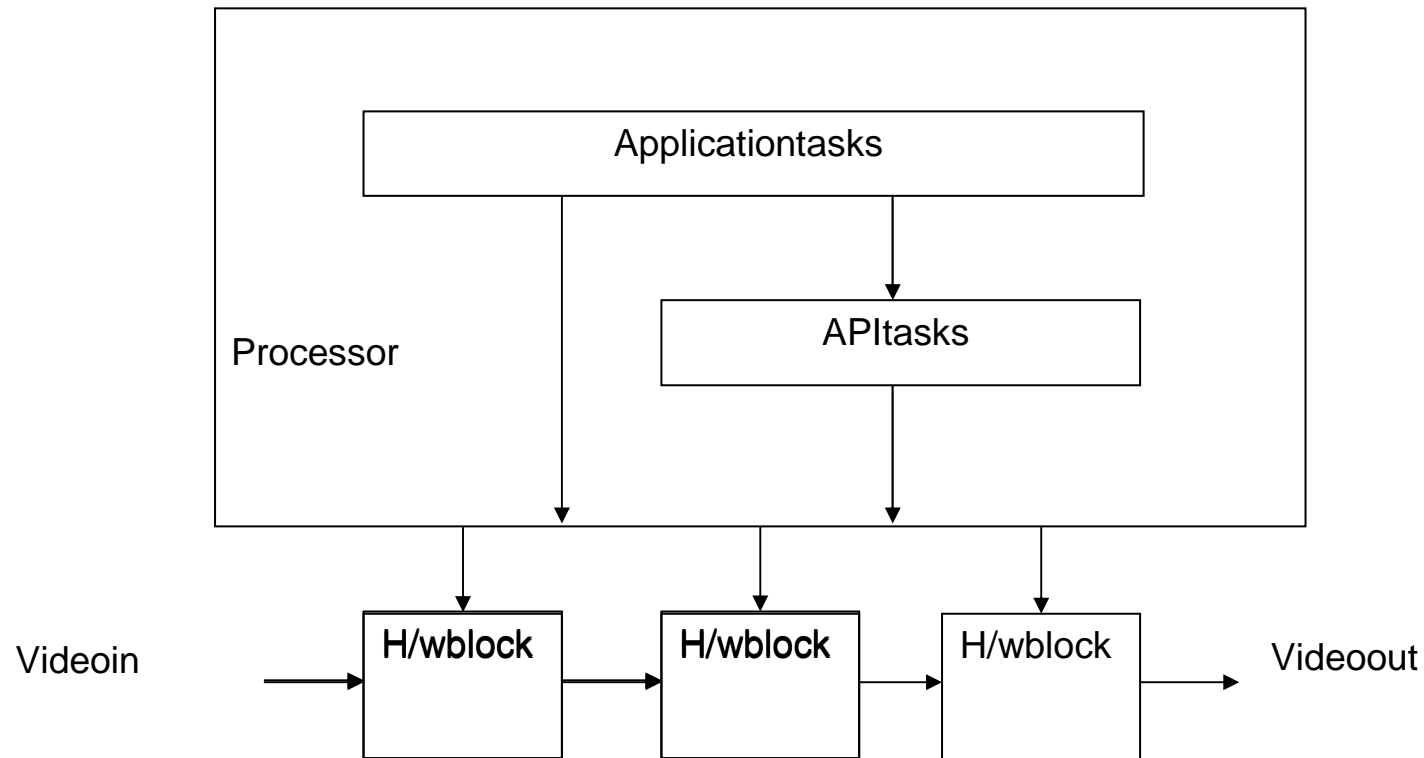
## Outline

- ✓ Introduction
- ✓ API based video processing systems
- ✓ Related work
- ✓ Fault categories
- ✓ Applicability of Test methods
- ✓ Implicit requirement elicitation and test approach
- ✓ Results and conclusions
- ✓ References

## Introduction

- ✓ Video processing systems are increasingly becoming complex
  - ⊗ User expectation about quality of the video and
  - ⊗ Number of source formats supported is increasing
- ✓ Application behavior on video processing APIs may be different
- ✓ From OEM application development view point
  - ⊗ APIs should be simple,
  - ⊗ Support multiple sources and
  - ⊗ Easy to integrate
- ✓ Hence, test team faces task of validating **complex API software** with **simple API specification**
- ✓ A test approach is needed that uses such API specification and elicited implicit requirements to validate API software

## API based video processing systems



## Related Work

- ✓ Category partition method
  - ⊗ Input space is partitioned
  - ⊗ Representative values are used for test
- ✓ Test generation based on software model
  - ⊗ Software system is modeled as state machine
  - ⊗ State machine traversals are used to generate tests
- ✓ Random calls approach
  - ⊗ Formal specification is available
  - ⊗ Pre-condition is known and verifiable
  - ⊗ Useful when application usage scenarios are not clear
  - ⊗ Used as reliability test method

## Fault categories

- ☑ Since API specification is abstract, we need an approach based on type of faults in such systems
- ☑ The following fault categories are identified while analyzing the problems reported
  - ☒ Parameter/return values
    - ☒ API behavior against API specification
  - ☒ Configuration data based errors
    - ☒ Default initialization values
  - ☒ Source dependant errors
    - ☒ API does not work for a particular source
  - ☒ Default initializations
    - ☒ When API is called, any default behavior
  - ☒ Behavior in group of APIs
    - ☒ When one of the APIs fails in a group of APIs, Is the behavior expected?
  - ☒ Implementation method of an API
    - ☒ How does an API configures the hardware? Does it cause any visual abnormalities?

## Applicability of Test methods

- ✓ Test methods can be used to address fault categories related to
  - ⊗ Parameter/return values
  - ⊗ Configuration data based errors
- ✓ The following methods can be employed to address above fault categories:
  - ⊗ Category partition
    - ⊗ Each distinct system parameter is partitioned into equivalent partitions
    - ⊗ Test values are chosen from such partitions
  - ⊗ Boundary value analysis
    - ⊗ Similar to above method, values are selected at boundaries of partitions
  - ⊗ Cause effect analysis
    - ⊗ Causes or system events and system responses are identified
    - ⊗ Tests are generated to verify their relationship

## Implicit requirement elicitation and test approach

- ✓ Other fault categories need to elicit implicit requirements from various documents
  - ⊗ Product specification
    - ⊗ Supported sources, formats
    - ⊗ Supported devices
  - ⊗ Use model
    - ⊗ Use of a feature from user and application developer perspective
    - ⊗ Type of artifacts not expected
  - ⊗ Functional Use Instruction
    - ⊗ Required configuration for the hardware block

## Results

- ☑ The test suite based on API documentation consists of 900 test cases.
- ☑ These test cases verify all documented input/output parameters & their relation and functionality.
- ☑ The requirement elicitation activity resulted in 120 new test cases.
- ☑ This increased the test suite by 13% approximately.

## Results

<b>Release</b>	<b>Total</b>	<b>Faultsrelatedtoa) andb)</b>	<b>Faultsrelatedtoc), d),e)andf)</b>
<b>Release1</b>	39	16	23
<b>Release2</b>	28	8	20
<b>Release3</b>	31	2	29
	<b>98</b>	<b>26</b>	<b>72</b>

## Results

- ☑ The approach enables us to find the defects early in the development life cycle.
- ☑ Otherwise these defects would be reported during customer product testing.
- ☑ The results also strengthen a software testing fact that APIs work fine when tested individually and fail when integrated.
- ☑ Hence, API based test for video processing system should develop comprehensive test suite with implicit requirement elicitation from various documents along with API documentation.

## Conclusions

- ☑ The approach was deployed in a product development cycle
- ☑ API based systems are popular since they reduce application integration effort since API specification just specifies only input/output relation.
- ☑ However, test can not only rely on such information.
- ☑ Customers report problems as specified in this paper.
- ☑ Hence, test should be comprehensive in eliciting requirements from various documents and developing tests to address them. However, the activity can be time consuming and cumbersome.
- ☑ This effort gives good benefit in terms of identifying the defects early in the development cycle and thus improving the product quality.