

Автоматическая генерация базовых тестов для программных интерфейсов библиотек на основе заголовочных файлов

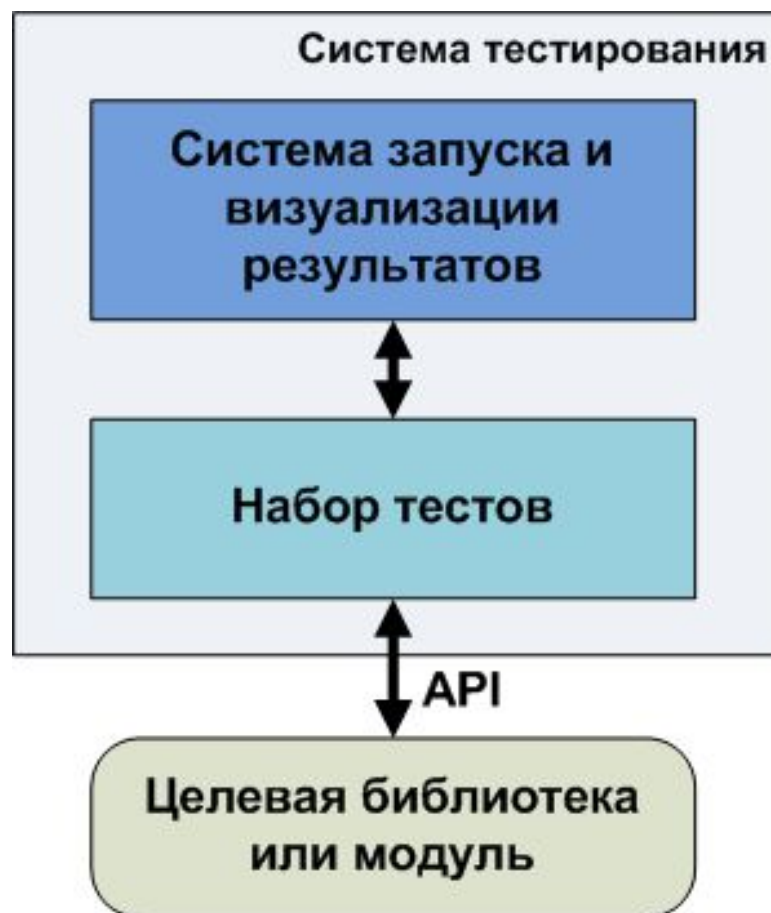
Владимир Рубанов,

Андрей Пономаренко

Институт системного программирования РАН



Тестируем библиотеки и модули



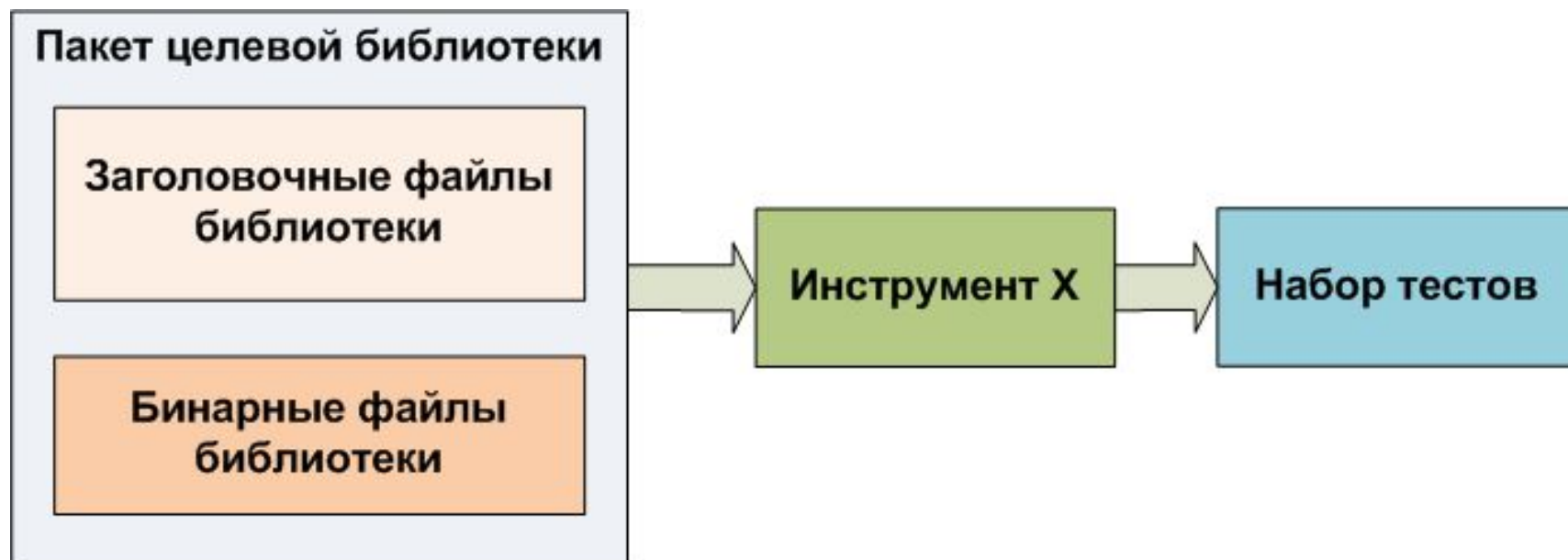
Какие тесты нужны?

- 1. Глубокие тесты** – вызывают каждую целевую функцию и их цепочки сотни раз с различными параметрами и в различных внутренних состояниях целевой системы. Тщательно контролируется корректность результатов работы.
- 2. Средние тесты** – вызывают целевые функции в нескольких основных сценариях использования. Контролируются основные результаты работы.
- 3. Базовые тесты** – каждая функция вызывается хотя бы один раз с некоторым корректным набором параметров. Контролируется отсутствие грубых ошибок.

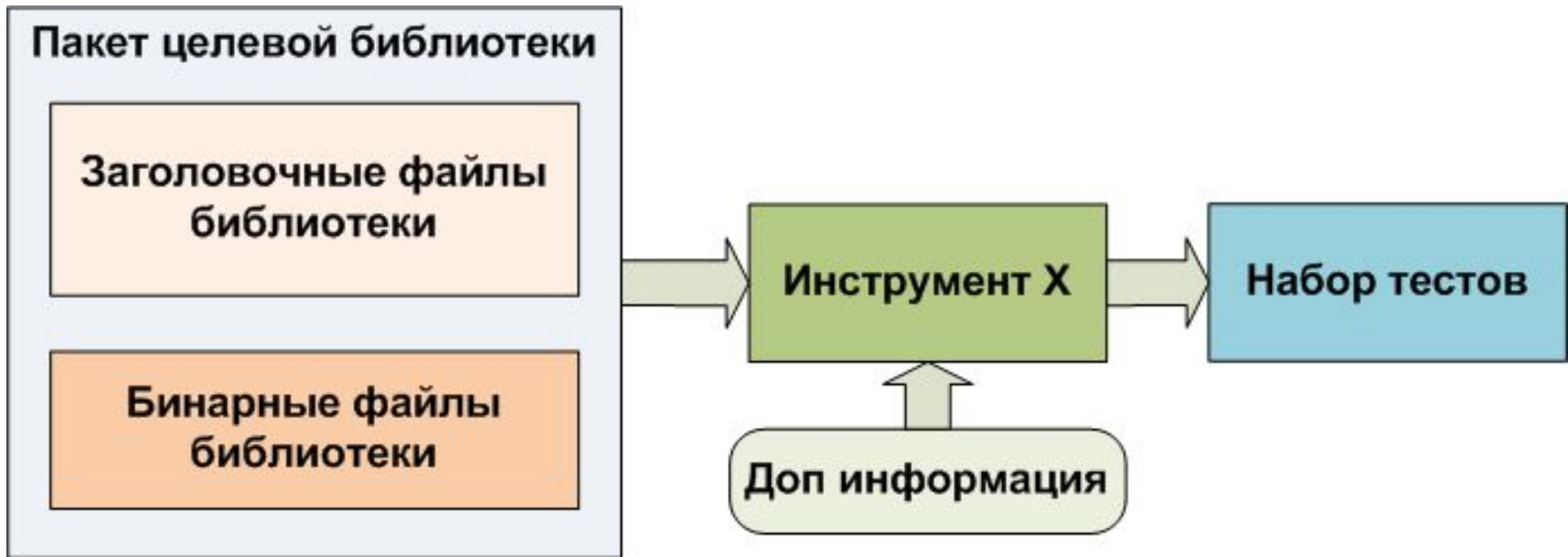
Технологии автоматизации: какой ценой?

- 1. Глубокие тесты** – тяжеловесные технологии, например model based на основе конечных автоматов (UniTESK)
 - ▢ *Высокая* удельная стоимость разработки тестов в расчете на одну целевую функцию.
- 2. Средние тесты** – классические unit тесты (CUnit, TET, T2C)
 - ▢ *Средняя* удельная стоимость разработки тестов в расчете на одну целевую функцию.
- 3. Базовые тесты** – ?
 - ▢ Целесообразно при *низкой* стоимости создания тестов.

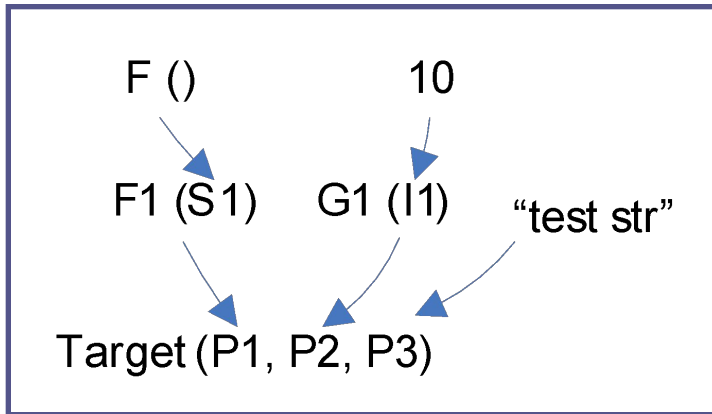
Базовая идея генерации базовых тестов



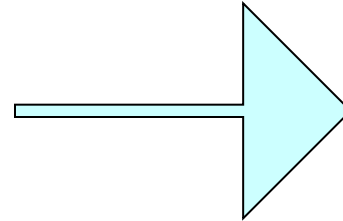
Базовая идея+



Построение цепочки инициализации



Дерево инициализации вызова



```
S1 = F ()
P1 = F1 (S1)
I1 = 10
P2 = G1 (I1)
P3 = "test string"
Res = Target (P1, P2, P3)
CHECK (Res != error)
```

Базовый тест

Дополнительная информация (опциональная)

- как правильно инициализировать библиотеку
- как получить корректное значение определенного типа данных
- каким должно быть корректное значение определенного параметра функции
- какие проверки можно сделать для возвращаемых значений определенного типа

Специальные конструкции в описании дополнительной информации

- **\$(type)** – создание объекта данного типа

```
void create_QProxyModel(QProxyModel* Obj) {  
    Obj->setSourceModel($ (QItemModel*));  
}
```

- **\$(function)** – вызов данной функции с корректными параметрами

```
xmlListPtr create_filled_list() {  
    xmlListPtr l = $(xmlListCreate);  
    int num = 100;  
    xmlListPushBack(l, &num);  
    return l;  
}
```

Характеристика технологии

- Даже без задания дополнительной информации получаются **вполне работоспособные тесты**, которые могут обнаруживать ошибки.
- Введение дополнительной информации позволяет **значительно повысить качество тестирования**, при этом такая информация **автоматически повторно используется** в сотнях и тысячах тестов.
- Получаемые тесты могут служить **готовой базой** для разработки более глубоких тестов.

Примеры использования (1)

- Официальные тестовые наборы Linux Foundation:
 - libxml2
 - Qt3
 - Qt4
 - ALSA
- Официальный тестовый набор rpm5
- Десятки upstream применений в качестве дополнительного шага контроля качества.

Примеры использования (2)

git index : freetype/freetype2.git
The FreeType 2 library

[summary](#) [refs](#) [log](#) [tree](#) **[commits](#)** [diff](#) [about](#)

author Werner Lemberg <wl@gnu.org> 2010-05-22 18:03:41 (GMT)
committer Werner Lemberg <wl@gnu.org> 2010-05-22 18:03:41 (GMT)
commit [e30de299f28370ed5aa65755c6be69da58eefc72](#) (patch)
tree [35355e4d7b42156baea7a21037d50f4c4ca5753c](#)
parent [09344385ee652cb8df33d35f07627c93e827f10d](#) (diff)
download [freetype2-e30de299f20370ed5aa65755c6be69da50eefc72.tar.gz](#)

Fix various memory problems found by linuxtesting.org.

* `src/base/ftgxval.c` (FT_TrueTypeGX_Free, FT_ClassicKern_Free),
`src/base/ftotval.c` (FT_OpenType_Free), `src/base/ftpfr.c`
(`ft_pfr_check`): Check `'face'`.

* `src/base/ftobjs.c` (FT_Get_Charmap_Index): Check `'charmap'` and
`'charmap >face'`.
(FT_Render_Glyph): Check `'slot->face'`.
(FT_Get_SubGlyph_Info): Check `'glyph->subglyphs'`.

Improve API documentation.

Примеры использования (3)

LIBSSH2

Search

[Login](#) | [OpenID Login](#) | [Preferences](#) | [Help/Guide](#) | [About Trac](#) | [Register](#)

[Wiki](#) | [Timeline](#) | [Roadmap](#) | [Browse Source](#) | **[View Tickets](#)** | [Search](#)

[← Previous Ticket](#) | [Next Ticket →](#)

Ticket #173 (closed defect: fixed)

Agent API doesn't call `_libssh2_error` consistently Opened [4 months ago](#)
Last modified [4 months ago](#)

Reported by:	alamaison	Owned by:	bagder
Priority:	normal	Milestone:	1.2.7
Component:	API	Version:	1.2.6
Keywords:		Cc:	
Blocks:		Blocked By:	

Description

All APIs should set an error message with `_libssh2_error` but the `libssh2_agent_*` API fails to do this in most places.

Распространение API Sanity Autotest

1. Лицензия GPL
2. Домашняя страница проекта:
 - http://ispras.linux-foundation.org/index.php/API_Sanity_Autotest
3. Принят в репозитории:
 - Alt Linux
 - Arch Linux
 - Debian
 - FreeBSD
 - ...

Ссылки

- API Sanity Autotest
http://ispras.linux-foundation.org/index.php/API_Sanity_Autotest
- Центр верификации ОС Linux
<http://linuxtesting.org>
- Институт системного программирования РАН
<http://ispras.ru>

Докладчик

- Владимир Рубанов, к.ф.-м.н.,
зав. сектором операционных систем ИСП РАН,
руководитель Центра верификации ОС Linux
- vrub@ispras.ru