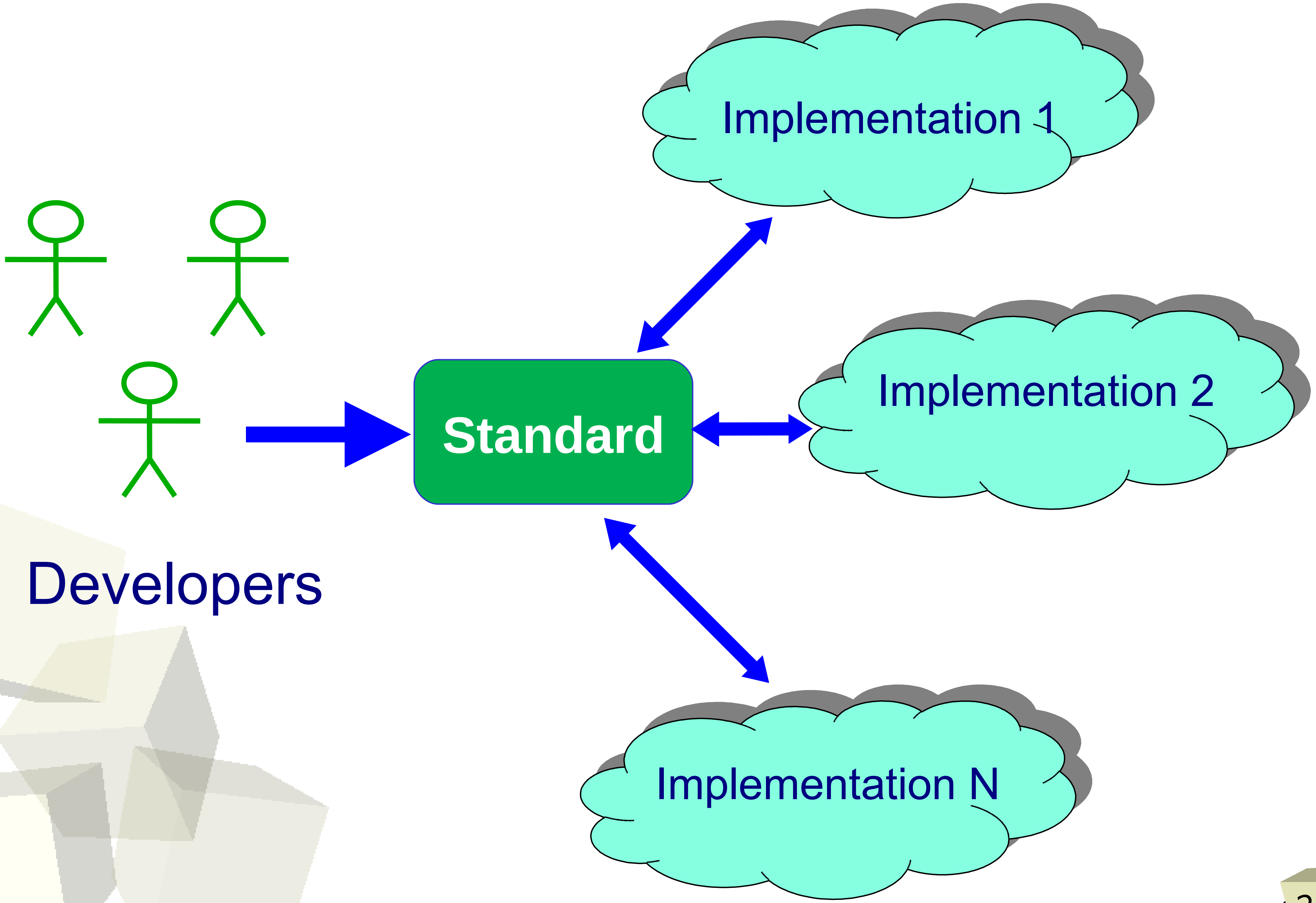
Designing a Development Environment to Support Creation of Standard-Compliant Applications

Denis Silakov

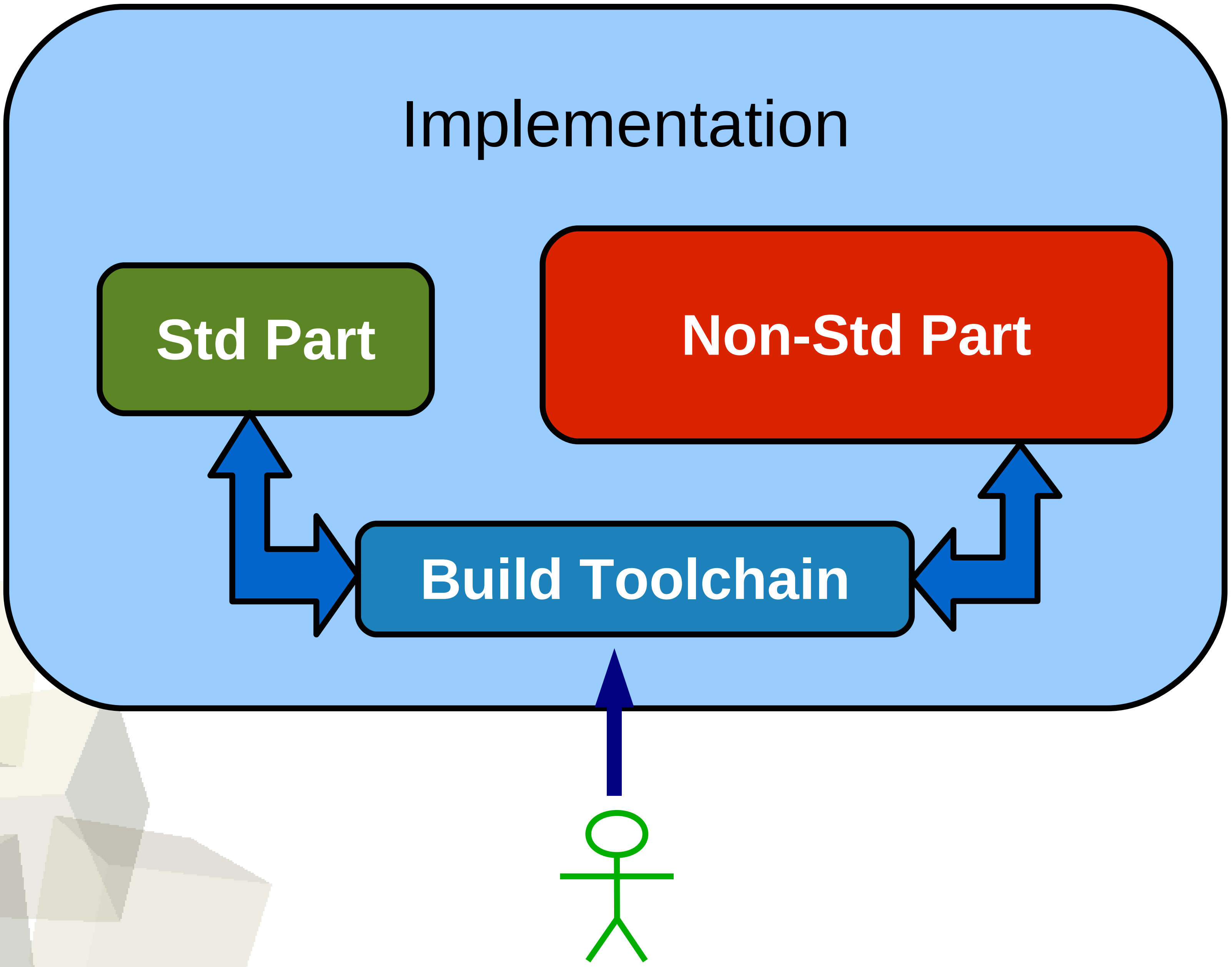
Institute for System Programming, RAS
http://ispras.ru/
Linux Verification Center
http://linuxtesting.org/

SYRCoSE 2009. 28-29 May, 2009, Moscow

Using Standards



Develop for Standard?



Possible Approaches

- “Careful” development
- Ideal implementation
  - mobile device emulators
  - “sample implementations” (LSB, OpenGL)
- Systematic testing
- Restricted environment inside the real implementation
  - '-std' option of gcc
  - LSB Development Environment (LSB SDK)

Implementation-based Environment

Idea

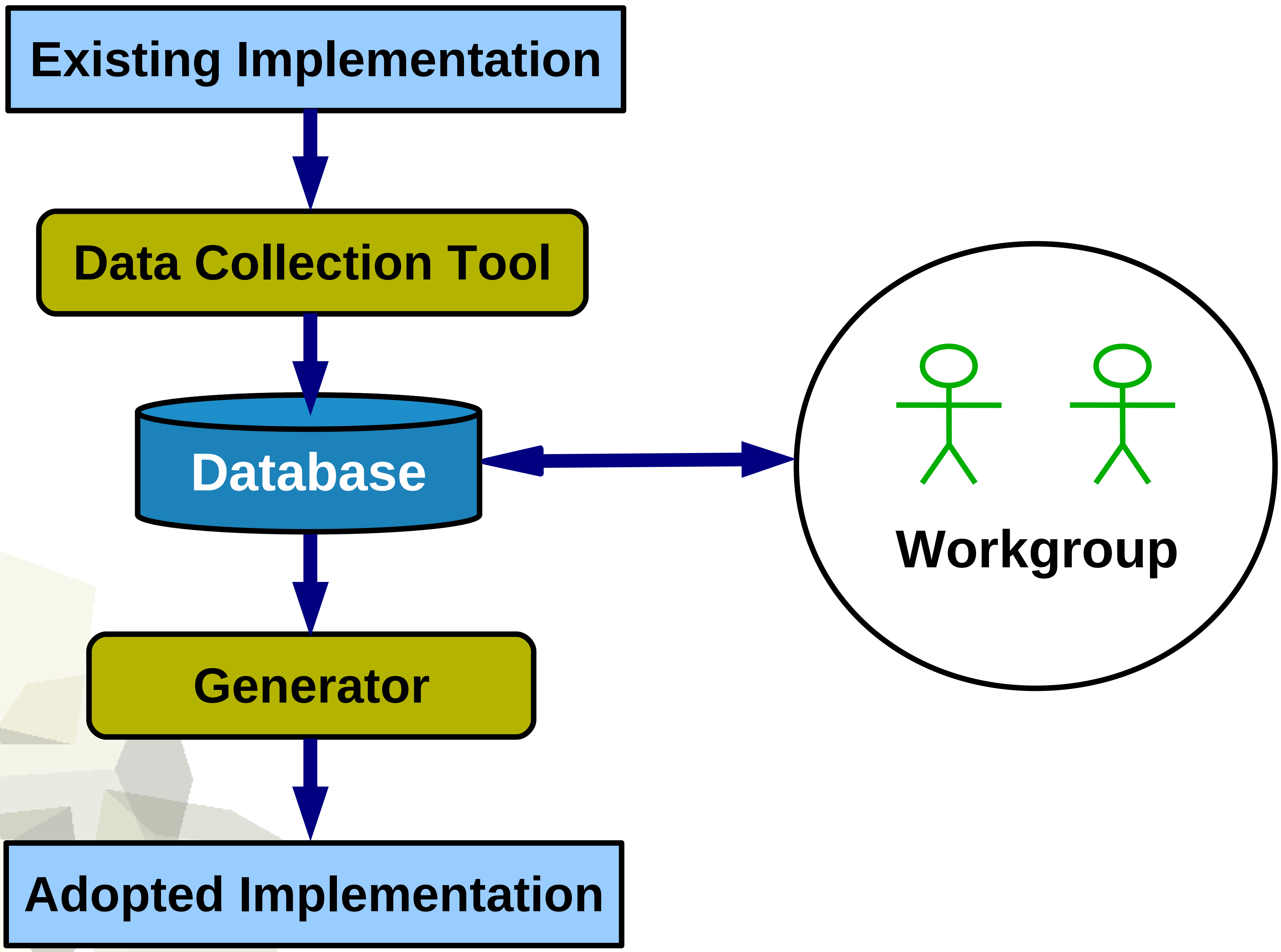
Take any compliant implementation and drop non-standardized items

Challenge

- Standard evolves
- Implementation evolves

How to reflect the changes in the environment?

Database Driven Approach



Database Design

What to store?

- Everything that depends on the standard
- Data used in more than one tool

Configuration flags

- Is particular entry is included in the standard?

Item interdependencies

- Dependencies in the real world → foreign keys in the database

Generator

Skeleton

+

Data from the DB

=

Generated Environment

Configuration Flags

Function	*Header*	*Incuded?*
gets	stdio.h	No
fgets	stdio.h	Yes
puts	stdio.h	Yes
fputs	stdio.h	Yes

```
/* begin stdio.h */
extern int puts (const char *);
extern int fputs (const char *, FILE *);
extern char *fgets (char *, int, FILE *);
/* end stdio.h */
```

Support Multi-Version Data

Temporal Database

- Time intervals – *appeared in v1*, *dropped in v2*
- Extra fields for extra status – *optional in v3*

Specifics

- Discrete time values
- *Valid* time only
- Small number of possible values
- Correlations in time intervals for interdependent items

Time Intervals

Function	***Header***	***Appeared***	***Withdrawn***
gets	stdio.h	1.0	1.2
fgets	stdio.h	1.0	*NULL*
puts	stdio.h	2.0	*NULL*
fputs	stdio.h	2.0	*NULL*

LSB Development Environment

- Header files *(generated)*
- Stub Libraries *(generated)*
- Compiler wrapper – forces system compiler to use LSB headers and link against LSB libraries

LSB SDK – MutliVersion Tool

Headers – driven by LSB_VERSION constant

```
#if LSB_VERSION >= 10
#if LSB_VERSION < 20
  extern char *gets (char *);
#endif
  extern char *fgets (char *, int, FILE *);
#endif
```

Libraries – separate file for every LSB version

Generated Code vs Generator Code

	Generators	**Generated Code**
SLOC	2,500	156,300
Development effort estimate	0,6 person-years (7 person-months)	39 person-years
Total estimated cost to develop	$70,000	$5,250,000

Approach Advantages

- Create environment not from scratch
- Consider only important parts of implementation (database schema = abstraction model)
- Automated synchronization (in case of automated tools)
- Support for several versions of the standard

URLs & Contacts

- LSB Infrastructure Project
  http://ispras.linuxfoundation.org
- LSB at the Linux Development Network
  http://ldn.linuxfoundation.org/lsb
- Denis Silakov
  silakov@ispras.ru