Evolution von Code bei Major-Releases von Programmiersprachen

am Beispiel der Migration zu PHP7

Martin Duschek

23. Januar 2020

HTWK Leipzig

Inhalt

Grundlagen

PHP

Abbildung 1: Major-Releases von PHP im Zeitverlauf

Versionierung von Software

Semantic Versioning :

Major.Minor.Patch[-Pre-Release]

- **Major**: inkompatible API-Änderungen
- **Minor**: abwärtskompatible Änderungen oder *Deprecations*
- **Patch**: abwärtskompatible Bugfixes

Versionierung von Software

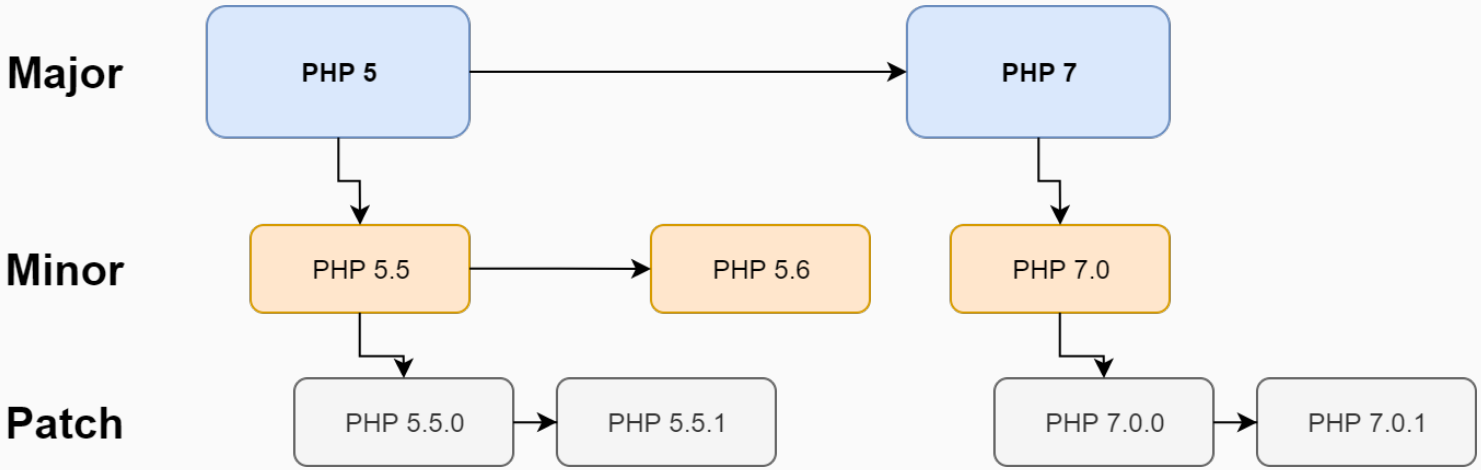


Abbildung 2: Semantic Versioning am Besipiel von PHP

ISO/IEC 14764

„Fahrplan" zur Migration :

- Anforderungsanalyse und Definition der Migration
- Entwicklung von Werkzeugen zur Migration
- Entwicklung der angepassten Software
- Verifikation der Migration
- Support der alten Umgebung

Ziele von PHP7

Ziele

- Verständlichkeit des Codes
- Erhöhung der Sicherheit
- Höhere Ausführungsgeschwindigkeit

Verständlichkeit des Codes

Indirekter Variablenzugriff

Ausdruck: $foo->$bar['baz']

Interpretation in PHP 5

$foo->$bar['baz']

Interpretation in PHP 7

$foo->$bar['baz']

Sicherheit

mysql

- keine Prepared Statements
- Weiterentwicklung aufgegeben
- modernere Alternativen

Geeignete Mittel

Manuelle Erkennung

Probleme bei komplexen Strukturen:

```
<?php
switch(1) {
  default:
    echo("Never evaluated");
    break;
  default:
    echo("Evaluated")
    break;
}
?>
```

Automatisierte Erkennung

Vorteile:

- Erkennung komplexen Codes
- Beliebige Projektgröße
- Abschätzung über den Aufwand möglich
- Bericht als Arbeitsplan

Versionsverwaltung

Vorteile:

- Protokollierte Änderungen
- Historische Versionen zurückverfolgbar
- Einfache Bearbeitung durch mehrere Personen
- Wartung alter Versionen weiter möglich

Lauffähigkeit historischen Codes

Lokale Umgebungen

- Unabhängige Installation
- keine Abhängigkeiten
- keine Versionsverwaltung

Docker-Integration

- Zentrale Konfiguration
- komplexes Ökosystem
- Versionsverwaltung per *git*

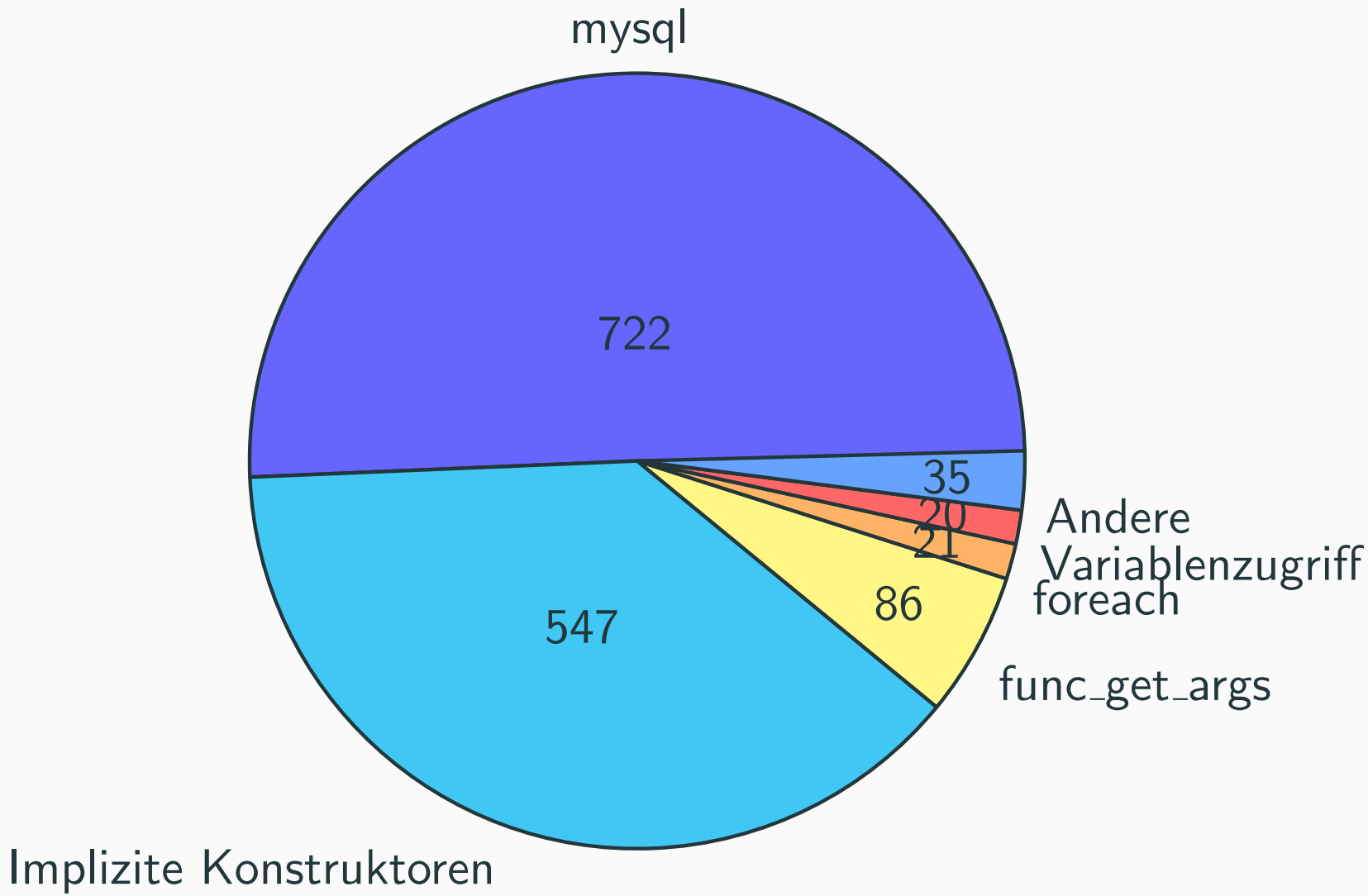
Migration

Anforderungsanalyse

Tabelle 1: Anteil zu migrierender Codeteile an der gesamten Codebasis

	Gesamt	Betroffen	Anteil
Dateien	5732	690	12,04%
Codezeilen	596198	1431	0,24%

Anforderungsanalyse



Entwicklung von Werkzeugen zur Migration

- Weiterentwicklung von *php7mar*
- Einführung von Docker
- Konfiguration von *XDebug*

Migration der Software

```php
//veralteter Aufruf von mysql
$result = mysql_query($query);
if(!$result) $output .= mysql_error();

//Ersatz durch mysqli
$result = mysqli_query($db_link, $query);
if(!$result) $output .= mysqli_error($db_link);
```

Migration der Software

```php
<?php
class order {
  function __construct($order_id) {
    $this->order($order_id);
  }

  function order($order_id) {
    print($order_id);
  }
}
?>
```

Verifikation

- Manuelles Testen
- keine Unit Tests
- Monitoring mittels *New Relic*

Support der alten Software

- Für hauseigene Software weniger relevant
- Parallelbetrieb während Übergangszeit
- Historischer Code weiter in Versionsverwaltung

Auswertung

Ziele

Sicherheit:

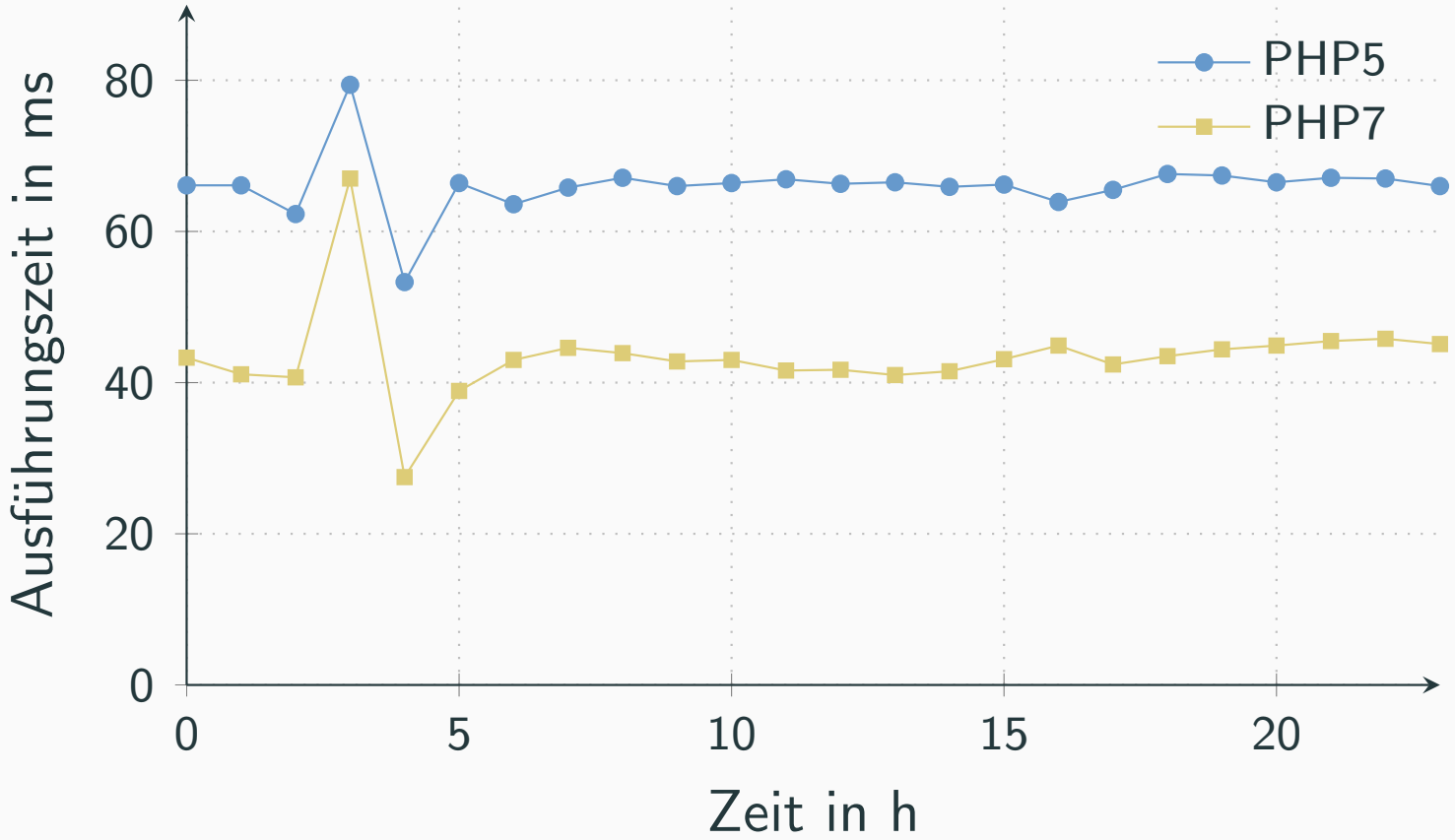
- Entfernung von *mysql*
- Entfernung potentieller Risiken

Klarheit des Codes:

- Einheitliche Konstruktoraufrufe
- Einheitliche Variablenaufrufe

Ziele

Geschwindigkeit:



Ausblick

Referenzen i


C. Anderson.
Docker [Software engineering].
IEEE Software, 32(3):102–c3, May 2015.


PHP Group.
PHP: Backward incompatible changes - Manual - Changes to variable handling.


ISO/IEC.
ISO/IEC/IEEE International Standard for Software Engineering - Software Life Cycle Processes - Maintenance.
ISO/IEC 14764:2006 (E) IEEE Std 14764-2006 Revision of IEEE Std 1219-1998), pages 1–58, September 2006.

Referenzen ii


Oracle.
MySQL :: MySQL 8.0 Reference Manual :: 13.5 Prepared SQL Statement Syntax.


Tom Preston-Werner.
Semantic Versioning 2.0.0.