
Walk

| Give | Pick up | Use |
|---|---|---|
| Open | Talk to | Push |
| Close | Look at | Pull |

# Die Altvordern

- Ein Anmeldeset ist …
  - Eine Menge von Anmelderegeln
  - Eine Menge von Veranstaltungen, für die diese Regeln gelten

# Anmelderegeln

- Bedingte Anmeldung
- Anmeldung zu maximal n Veranstaltungen
- Anmeldung gesperrt
- Anmeldung mit Passwort
- Zeitgesteuerte Anmeldung
- Beschränkte Teilnehmendenzahl
- Veranstaltungsbezogene Anmeldung
- Bevorzugte Anmeldung
- Kurs mit Teilnahmebedingungen
- Anmeldung nur über verknüpfte Veranstaltung

# Extras

- Priorisierung bei Kombination aus “Maximal n Veranstaltungen” und “Beschränkte Teilnehmendenzahl”
- Eigene zeitliche Gültigkeit von Anmelderegeln
- Personenlisten für Härtefälle/Malusregeln
- Austauschbarer Verteilungsalgorithmus (nicht über die Oberfläche)
- Regeln durch Plugins erweiterbar

# Was könnte besser sein?

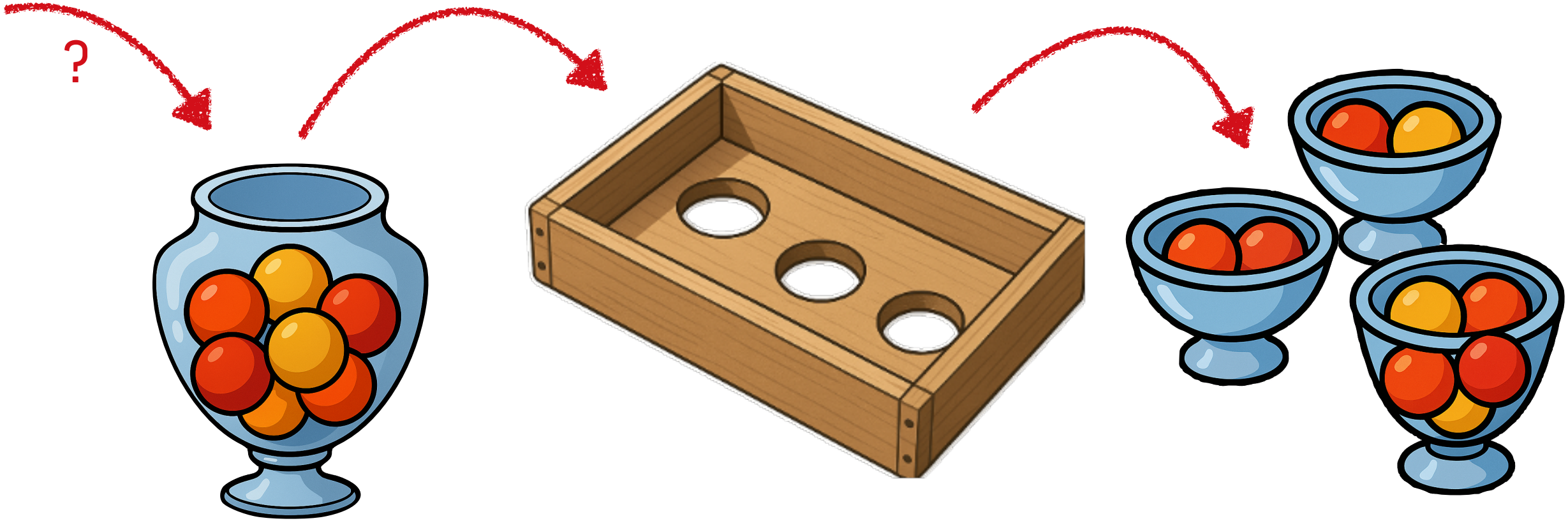
- Anmelderegeln wissen nichts voneinander / können nicht interagieren.
- Anmelderegeln sind gleichwertig und werden sequenziell abgearbeitet.
- “Spiel der Wahrscheinlichkeiten” - keine 100% Sicherheit
- Nachvollziehbarkeit der Platzverteilung
  - Vollständiges Logging
  - Löschen von Prioritäten
  - Bonus bei Prioritäten/Bevorzugung
- Veranstaltungen können nur einem Anmeldeset zugehören
- Code von 2013: u.a. keine Implementierung mit SORM
- Anmelderegeln gelten immer systemweit

```php
private function distributeByCourses($courseSet)
{
    Log::debug( message: 'start seat distribution for course set: ' . $courseSet->getId());
    $groups_quota = [];
    $conditional_rule_filter = array_filter($courseSet->getAdmissionRules(), function ($r) {
        return $r instanceof ConditionalAdmission
            && count($r->getConditionGroups());
    });
    $conditional_rule = array_pop( &array: $conditional_rule_filter);
    $conditiongroups = $conditional_rule ? $conditional_rule->getConditionGroups() : [];
    if (count($conditiongroups)) {
        foreach (array_keys($conditiongroups) as $group_id) {
            $groups_quota[$group_id] = $conditional_rule->getQuota($group_id);
        }
    } else {
        $groups_quota[] = 100;
    }
    foreach ($courseSet->getCourses() as $course_id) {
        $waiting_users = [];
        $course = Course::find($course_id);
        if (!$course) {
            Log::debug(sprintf( format: 'course %s not found, continue', $course_id));
            continue;
        }
        $free_seats_course = $course->getFreeSeats();
```

# Neue Anmeldeverfahren

- Jedes Anmeldeverfahren betrifft immer n Veranstaltungen (wobei n=1 der häufigste Fall sein dürfte)

# Neue Anmeldeverfahren



# Schritte des Algorithmus

- 1) Feststellen, ob jemand überhaupt in eine Veranstaltung darf
- 2) Registrierung mit Formular
- 3) Das Verfahren beginnt (zu einem bestimmten Zeitpunkt, Windhundverfahren)
- 4) Auswählen einer Liste durch Gewichtung oder durch Härtefälle
- 5) Zuteilung der Veranstaltung durch Gewichtungen
- 6) Entscheiden, ob die Kugel zurück gelegt wird
- 7) Abschluss des Verfahrens

# 1) Feststellen, ob jemand überhaupt in eine Veranstaltung darf

- Ist jemand im richtigen Studiengang?
- Ist jemand im falschen Fachsemester?
- Ist die Veranstaltung komplett gesperrt?
- Falls ja, soll das sofort kommuniziert werden, dass eine Eintragung nicht möglich ist.

# 2) Registrierung mit Formular

- Zusatzangaben der Studierenden
- Prioritätenwahl
- Passwort-Eingabe

# 3) Das Verfahren beginnt

- Ein Verfahren kann mehrmals durchlaufen werden
- Beim Windhundverfahren wird es zum Beispiel stets nach einer Registrierung einmal automatisch durchlaufen
- Beim Losverfahren wird es aber nur einmal gestartet

# 4) Auswählen einer Liste

- Die registrierten Personen werden in einer Liste gesammelt, die eine Reihenfolge hat.
- Dabei können sie gewichtet werden. Verschiedene Anmelderegeln können die Gewichte mit Faktoren beeinflussen.
- Anmelderegeln können aber auch die komplette Liste verändern, um Härtefälle auf Platz 1 zu heben.

# 5) Zuteilung der Veranstaltung

- Die Liste der Personen wird jetzt der Reihe nach abgearbeitet.
- Für Person X wird eine Veranstaltung aus dem Set ausgewählt.
- Anmelderegeln können hier jeder Veranstaltung eine Gewichtung vergeben.
- Dabei können Sie Angaben aus der Registrierung wie die Prioritäten berücksichtigen.
- Sie können auch den Faktor 0 vergeben, damit eine Veranstaltung gar nicht mehr zugeteilt wird - etwa wenn sie voll ist.

# 6) Entscheidung über Zurücklegen der Person

- Die Person ist jetzt einer Veranstaltung zugeteilt oder nicht.
- Eine Anmelderegel kann jetzt entscheiden, ob die Person in die “Urne” der Personen wieder zurück gelegt werden soll oder nicht (nicht in die Liste, die gerade abgearbeitet wird).
- Standard: Person wird nicht zurück gelegt.
- Aber vielleicht kann die Person in einem weiteren Durchlauf des Algorithmus erneut berücksichtigt werden.

# 7) Abschluss des Verfahrens

- Alle Personen sind von der Liste entfernt.
- Aber eventuell sind noch Personen in der Urne / Warteliste
- Jetzt kann theoretisch das Verfahren erneut gestartet werden. Das müsste aber von Hand geschehen.
- Anmelderegeln können definieren, ob das Anmeldeverfahren geschlossen wird in dem Sinne, dass sich niemand mehr registrieren kann.
- Beim Windhundverfahren soll es offen bleiben.

# Anmelderegeln

- Anmelderegeln haben spezifische Events, bei denen sie in den Algorithmus eingreifen können.
- Sie können definieren, ob jemand sich nicht einmal registrieren darf und kann (mit Meldung über den Grund).
- Sie können das Registrierungsformular erweitern um Formularelemente
- Sie können die Liste der Personen manipulieren durch Gewichtsfaktoren und noch einmal insgesamt.
- Sie können die Gewichte der Veranstaltungen mit einem Faktor (0 bis unendlich) manipulieren.
- Sie können verlangen, dass das Verfahren nach dem Losen erneut gestartet wird.

# Anmelderegeln

- Plugins können Anmelderegeln mit einbringen und haben dabei mehr Freiheiten als zuvor.
- Alle aktuellen Anmelderegeln lassen sich damit abbilden.
- Weil die Grundstruktur weniger von Fallunterscheidungen abhängt, kann man (hoffentlich) besser loggen.

# Diskussion

**Vielen Dank für Aufmerksamkeit und Mitarbeit!**

Thomas Hackl
hackl@data-quest.de

Rasmus Fuhse
fuhse@data-quest.de