Driving SAFe

James Janisse, VP

World Class Software Delivery Program
Agenda

• Our Situation in 2012
• Our Transition to SAFe
• Experience and results
  – The Good
  – The Bad
  – The Ugly
TomTom @ a Glance

- Founded in Amsterdam in 1991
- 1 billion euros annual revenue
- 4,000 employees in 35 countries worldwide
- Maps cover 112 countries
- Real-time traffic in 32 countries
- 70 million PNDs sold since 2004
- 3 million in dash navigation systems sold since 2009
- 1-2 Million lines/code per navigation product
- 750-1000 engineers in 11 development sites
TomTom January 2012

• Organised as waterfall projects
• Many projects working in all parts of the code with minimal module or component ownership
• Many releases are months-quarters late
• Multiple code lines and branches
• Negligible automated testing & no continuous integration
• “downstream” teams spend 3,4,5 months accepting the code and often changing it
• Poor visibility and facts based decision making
Conclusions

• Waterfall and staged gate milestones are not working for us
• Project orientation is wrong
• Branching the code is evil
• Complexity is too high
• Waste
• Insufficient information to make effective decisions

Strategy

• Transform from one-time project orientation to component-based continuous integration and delivery
Agenda

• Our Situation in 2012
• Our Transition to SAFe
We First Invented Our Own Solution: Feature Flow
Then We Made a Discovery
Then We Made a Discovery
Adoption Timeline

1. Book arrives on-site and is a read by a few Agilistas
2. Replace homegrown *Feature Flow* with SAFe
3. Give Book to SVP who reads it cover to cover on his vacation
4. SVP buys book for CTO and other SVPs
5. Attend SAFE Training
6. Trained 50 Certified Scrum Masters & 50 CPOs
7. Re-org
We Re-organised from Scrum Teams up

1. Key assumption is value is only created by scrum teams
2. Organise into Product clusters and component scrum teams
   – One Agile Release Train per Product
   – Every active/viable module/component is allocated to one and only one scrum team
3. Adjust/supplement all scrum teams to have scrum master, developers, etc
4. Everyone not in a scrum team is put in a backlog i.e. Project Managers, Resource Managers, Team leads….
5. Design a thin/lean program support team to feed the scrum teams: Product Owner, Architect, Systems Team
6. Design a thin/lean portfolio team
6 Months Into the Transition a New Goal

Launch the 4th Generation of our Consumer navigation product

- You have 126 Days till launch
  - usually 1 year project

- Bring in teams from 2 other (waterfall) Product Units and work as one integrated team
Doubled Number of Agile Release Trains

**NavUI**
- 5 Scrum teams
- 3 locations

**NavKit**
- 14 Scrum teams
- 4 locations

**Panda**
- 4 Scrum teams
- 2 locations

**Estrella**
- 5 Scrum teams
- 2 locations

**Nav4 Core**

---

**Location Dates**
- **Baikonur**: Dec 7th
- **Cape Canaveral**: Mar 8th
- **Vancouver**: Nov 2nd
- **Ingstad**: Feb 1st
- **Da Gama**: May 3rd
- **Lecce**: Dec 14th
- **Gomera**: Mar 8th
“There is no doubt in my mind that without SAFe and Rally we would not have launched this in only 140 days. It is also our best new product ever”
2014 to Present

- SAFe is adopted by all large product teams
  - Approximately 750 FTEs
    - Navigation software
    - Online services
    - Map creation software
    - Sports software
  - 515 users of CA Agile (Rally)
  - 200+? people trained in SAFe
Agenda

• Our Situation in 2012
• Our Transition to SAFe
• Experience and results
What is your pain?

- Increasing driver expectations on quality, ability to update, fresh look and feel....
- Big Up Front Design (BUFD) is not working
- How to switch from one-time delivery to frequent in-service updates
- Project success rate is low (on-time, on-budget, value delivered...)
- Maintaining multiple code branches in parallel
- Too many surprises at the end when it all comes together
- Early assumptions (technical, functional, UX...) prove incorrect
- Time to market is increasing
- Project orientation vs continuous flow (integration and deployment)
- Project costs are increasing
- Sustainable development pace
- Projects take too long
Agile in Automotive, Stuttgart, November 2015

What is stopping your from adopting Agile practices

- Supplier resistance
- Internal resistance
- Just a crazy fad
- Don't believe in it
- Management support
- Not compatible with automotive practices
- No desire to change
- No expected benefits
Question

What are the differences between…

Delivering value via projects and Delivering value via continuous integration & delivery
Flow

2012

• Batch size = 1
• Work is organized by stages:
  Plan > Analyse > Develop > Test
• Risk remains till end
• Value only delivered at end

Now

• Uncouple required features to flow independently and as fast as possible
• High risk & high value working first
• Release value in pieces ASAP

The Principles of Product Development Flow
Donald Reinertsen
Organisation

2012
• Bring people to the requirements
• Overhead of “Resource Managers”
• New team has no history of working together
• Throughput is unknown
• Impact of schedule risk

Now
• Bring requirements to the team(s)
• Minimal overhead
• Proven historical velocity
• Established way of working
• Clear long term ownership
• Self managing teams
• Team not managers commit
Requirements

2012

• Assumes that we, or our customers, can fully understand all the requirements up front
• Change is discouraged
• One chance to be “perfect”
• One chance so ask for everything

• Does not work for high user experience scenarios, where you need to see it working first
• Discourages innovation

Now

• Add features and fine tune performance over time
• Fail fast, learn, improve
• Assume that change will be constant, and we deliver in small increments to better track change

• Fine tunes and improves the user experience
• Reach minimum marketable product faster & with less waste
Architecture

2012
• Big Up Front Design (BUFD)
• Cannot spend months and months designing a future proof perfect architecture
• Often based on many untested assumptions and hypothesis

Now
• Architectural vision & emerging runway
• Better to have a vision and “barely sufficient” architecture
• Test assumptions & hypothesis ASAP
• Change is constant so architectural runway is built just in time to prevent waste
• The need to refactor is not failure
Performance KPIs

2012

- On-time
- On-budget
- Code Maturity
- Timesheets & estimate to complete
- Never ask if project X is an improvement over Project Y
- Mostly time accounting focus
- Time spent does not easily equate to business value delivered

Now

- Measure actual value delivered
- Cycle time trend
- Velocity trend
- Sprint and Release Burndown
- Focus is on trends and continuous improvement
- No more timesheets
Demo

2012
• Runs near the end of the project
• Hope for a miracle near the end of the project
• Very little opportunity to improve or re-factor
• Cost to change behavior is high

Now
• System always runs
• Demo production ready, tested software every 2-3 weeks
• Release to beta testers every 2-3 weeks if not daily
• Opportunity to verify designs, make improvements or move on
# Code

## 2012
- Branched for each project
- Ownership unclear
- Significant waste and overhead
- Effort to merge code back in was around 20% of total effort
- Single defect may have to get fixed in each branch
- Projects often break features
- Developers did not have to maintain their code

## Now
- Only one mainline
- Ownership 100% clear
- Negligible refactoring or waste
- Each product variant is built from the same code line so it benefits from the entire install base
- Developers maintain their own code and control tech debt
Integration

2012

• Infrequent & near the end
• effort to schedule integrations
• significant effort to conduct integration months after branching
• Each integration would trigger significant corrective actions
• Catching up could take 20% of the total effort

Now

• Continuous and ongoing
• Each developer submits their code several times/day
• Each submission requires success test automation results
• Whoever breaks it fixes it without delay
• Daily full regression suite
• Daily integration with downstream systems
# Test Automation & Continuous Integration

<table>
<thead>
<tr>
<th></th>
<th># tests running on CI manner</th>
<th>Nightly</th>
<th>Per release</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Map Validation</strong></td>
<td></td>
<td></td>
<td>~1,5 Million test cases/new map release</td>
</tr>
<tr>
<td>NAVKIT</td>
<td>9700 unit tests</td>
<td>4600 Component tests</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Avg. 3 times/day</td>
<td>3930 Reflection Tests</td>
<td></td>
</tr>
<tr>
<td>NAVAPP</td>
<td>5856 unit tests</td>
<td>2200 Unit tests</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Avg. 3 times/day</td>
<td>6200 Functional tests</td>
<td></td>
</tr>
<tr>
<td>ONP</td>
<td>1300 unit + Functional</td>
<td>1081 Unit Tests</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Avg. 3 times/day</td>
<td>222 Functional Tests</td>
<td></td>
</tr>
<tr>
<td>ONA</td>
<td>11 Unit tests</td>
<td>11 Unit Tests</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Avg. 5 times/day</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NC</td>
<td>1216 Unit tests</td>
<td>1216 Unit Tests</td>
<td>Load Tests</td>
</tr>
<tr>
<td></td>
<td>Avg. 5 times/day</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ON</td>
<td>2346 Unit + Functional</td>
<td>1766 Unit Tests</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Avg. 3 times/day</td>
<td>580 Functional Tests</td>
<td></td>
</tr>
<tr>
<td>E2E testing</td>
<td>30 E2E tests</td>
<td>30 E2E functional tests</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Avg. 4 times/day</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td>85.700 automated tests/day</td>
</tr>
</tbody>
</table>
## Summary of “The Good”

<table>
<thead>
<tr>
<th>Observation</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Always</strong> release on fixed schedule</td>
<td>• Reliable and predictable releases of production code</td>
</tr>
<tr>
<td></td>
<td>• Establishes fixed rhythm</td>
</tr>
<tr>
<td>Release <strong>quicker</strong> and more <strong>often</strong></td>
<td>• Fail fast (&lt;2 weeks) is better than after 6 months</td>
</tr>
<tr>
<td></td>
<td>• Validate and adapt sooner</td>
</tr>
<tr>
<td></td>
<td>• Adapt to change/learnings</td>
</tr>
<tr>
<td>Run automated tests suite <strong>per</strong> submission &amp; <strong>per</strong> day</td>
<td>• Detect/prevent issues with each new submission</td>
</tr>
<tr>
<td></td>
<td>• Mainline is always able to run</td>
</tr>
<tr>
<td></td>
<td>• No bottleneck at the end</td>
</tr>
<tr>
<td></td>
<td>• Reduces waste as others stay up to date</td>
</tr>
<tr>
<td>Single <strong>shared</strong> backlog available to <strong>all</strong> to view</td>
<td>• Improved transparency and info sharing</td>
</tr>
<tr>
<td></td>
<td>• Done means working capability not task complete</td>
</tr>
<tr>
<td><strong>Perpetual</strong> teams</td>
<td>• Teams establish ways of working &amp; esprit du corps</td>
</tr>
<tr>
<td></td>
<td>• Improves estimating by allowing historical comparisons</td>
</tr>
<tr>
<td></td>
<td>• Enables estimation accuracy analysis</td>
</tr>
<tr>
<td></td>
<td>• Team controls their own commitments</td>
</tr>
<tr>
<td></td>
<td>• Sustainable development</td>
</tr>
</tbody>
</table>
## Summary of the “Bad”

<table>
<thead>
<tr>
<th>Observation</th>
<th>Impact</th>
</tr>
</thead>
</table>
| Teaches the business that Agile = 100% predictable     | • What happened to iterative development  
• What happened to incremental development               |
| Enables the business to change direction/ strategy/ priorities often | • Let’s face it, too much Agility is just an inability to make choices and decisions i.e. chaos |
| Everyone can see everyone’s backlog, priority, throughput… | • Everyone can second guess your prioritisation  
• Everyone can second guess your estimates                |
| Shows week teams                                       | • Teams that normally staid below the radar which no one new what they did are suddenly very exposed to the daylight |
| Goes on forever without a break (HIP downtime)         | • Projects used to have a nice slow start up and shut down phase, so cyclical rhythm  
• Now work is harder and does not let up                 |
## Summary of the “Ugly”

<table>
<thead>
<tr>
<th>Observation</th>
<th>Impact</th>
</tr>
</thead>
</table>
| Some teams will resist                   | • Reject the need to be part of an ART  
• Reject the need to have common ways of working… |
| Some people loose power                  | • Project managers lose scope, resourcing, and budget  
• Resource managers are no longer needed |
| Centralised decision making shifts to decentralised | • Evolving decisions to the lowest level threatens central portfolio level experts like architects and makes guiding independent teams hard |
| Fully defined transforms to barely sufficient | • Architects still love to make future proof architectural designs and plans  
• UX want to make pixel perfect designs for all use cases |
Thank You

Any questions?

http://nl.linkedin.com/in/jamesjanisse