Early Time-Budgeting: A Methodology for Developing Component Based Embedded Control Systems

Manoj Dixit and S. Ramesh, ISL, GM R&D
Pallab Dasgupta, IIT Kharagpur
Background

- Modern embedded control systems have large number of features
- Safety features have strict real-time end-to-end requirements
- Many components interact together to meet system level requirements
- System is distributed in nature

How to do a **timing layout** of entire system to meet end-to-end real-time requirements?
Illustrative Example

In a crash scenario, the airbags blow up to enable passenger safety

Crash Scenario → Airbags blown

Crash Detect → Disable Power Steering → Collapse Steering → Blow Airbags

Functional Specification

In a crash scenario, the airbags must blow up within 20ms to enable passenger safety

Crash Scenario → Airbags blown

Crash Detect → Disable Power Steering → Collapse Steering → Blow Airbags

Safety Requirement
**Illustrative Example**

In a crash scenario, the airbags blow up to enable passenger safety

- Crash Scenario → Airbags blown
- Crash Detect → Disable Power Steering → Collapse Steering → Blow Airbags

In a crash scenario, the airbags must blow up within 20ms to enable passenger safety

- Crash Scenario → < 20 ms → Airbags blown
- Crash Detect → Disable Power Steering → Collapse Steering → Blow Airbags

**Functional Specification**

**Standards/Statistical data**

**Safety Requirement**

**Actual Implementation**

**Time Budgeting**
More General Interactions

**End-to-End Latency < 100 ms**
Emerging Challenges

- Increasing complex features
- Multiple functions in a single computational unit, e.g. AUTOSAR
- More component sharing promoted by the smaller component sizes
- Need for advance planning of resources for extensibility

This is leading to...
- Increasing Real-time interdependencies between components
Example: Timing Dependencies

End-to-End Latency < 500 ms
Prevalent Approaches

- Ad-hoc estimates about component response time
- Architecture exploration to do component-task mapping, component-ECU mapping etc.
- On failure, difficult to trace the culprit component

How to budget time for each component is not clear

Part of the figure from:
Embedded System Design for Automotive Applications, Marco Di Natale et.al.
IEEE Computer 2007,
Our Proposal

- Rigorous-Systematic methodology
- To be applied during early stages at the requirements level
- Timing requirements are captured in parametric form
- Use formal specification and analysis methodology to generate constraints on parameter valuations

Part of the figure from: 
Embedded System Design for Automotive Applications, Marco Di Natale et.al. IEEE Computer 2007,
How we propose to do this?

In a crash scenario, the airbags **must** blow up **within 20ms** to enable passenger safety.

- Crash Scenario → < 20 ms → Airbags blown
- Crash Detect → Disable Power Steering → Collapse Steering → Blow Airbags

**Parametric Specifications**

**Time-Budgeting:** What values of x, y, w, z are good-enough?
The Problem

- We are given a set of features and their real-time requirements
- We are given a set of components and their parametric-time requirements for implementing these features
- Propose Early stage Time-Budgeting Methodology
  - Find constraints over parameter values
  - Design space exploration to select suitable valuation
  - Scalable
Our Contributions


Hierarchical Time-Budgeting Methodology

- Preliminary version in DATE 2010
- Patent under review
**Time-Budgeting Methodology**

- **Feature Requirements** (known-timing)
- **Component Requirements** (unknown-timing)
- **Formalized Feature Requirements** (Real-time)
- **Formalized Component Requirements** (Parametric-time)
- **Design Constraints**
- **Optimization Decisions**

**New Algorithms**

**Constraint Solving:** Well known

**Time Budgeting Algorithm**

- **Formal Analyzer of Parametric Specs**
- **Linear Constraints On Parameters**
- **Constraint Optimizer**

**Novelty:** System level optimization gets converted to constraint solving, scalability is much better this way
Time-Budgeting Challenges

- Component Decompositions are hierarchical: DAG
- Need to budget timing for all components
- Each feature component has 10s of requirements
- **simultaneous budgeting not advisable**
Hierarchical Time-Budgeting Algorithm

- Budget time for all top level components from feature requirements
- Requirements of features and top level components are analyzed together
Hierarchical Time-Budgeting Algorithm

- Split component time-budgeting into smaller subproblems and repeat
- Compositional approach
- DF-traversal with back-tracking takes care of component re-use case
Main Step in Time-Budgeting

- A collection of real-time requirements which is refined into a collection of parameterized requirements

- Find suitable parameter valuation so that, the instantiated requirements satisfy all real-time requirements from which they have been derived

- Actually a standard verification check!! But reduces to constraint synthesis here!
Formalization of Refinement Step

- Requirements refinement step is formalized as a collection of **requirement decomposition pairs**
- \( f \): feature requirement
- \( g_1, \ldots, g_k \) are component requirements identified for \( f \)
- \((f, \{g_1, \ldots, g_k\})\) is a requirement decomposition pair

**Theorem:** It is enough to analyze each pair separately, compute **validity constraint**. Any solution to conjoined constraint defines a suitable time-budget

In a crash scenario, the airbags must blow up within 20ms to enable passenger safety
Formal Specification for Requirements

- Parametric Linear Temporal Logic (PLTL)
  - Extends well known Linear Temporal Logic
  - Semantics is defined by using a parameter valuation

<table>
<thead>
<tr>
<th>Feature/Component Name</th>
<th>PLTL Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC Feature</td>
<td>$\phi_1 : \square (lead_slow \Rightarrow \Diamond \leq_{500} apply_brake)$</td>
</tr>
<tr>
<td>Sensor Component</td>
<td>$\psi_1 : \square (lead_slow \Rightarrow \Diamond \leq_{x_1} lead_kinematics_info)$</td>
</tr>
<tr>
<td>ACC Controller Component</td>
<td>$\psi_2 : \square (lead_kinematics_info \Rightarrow \Diamond \leq_{x_2} apply_brake)$</td>
</tr>
</tbody>
</table>

$((\phi_1, \{\psi_1, \psi_2\}))$ is the RDP
Validity of a Requirement Decomposition Pair

<table>
<thead>
<tr>
<th>Feature/Component Name</th>
<th>PLTL Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC Feature</td>
<td>$\phi_1 : \Box (lead_slow \Rightarrow \Diamond_{\leq 500} apply_brake)$</td>
</tr>
<tr>
<td>Sensor Component</td>
<td>$\psi_1 : \Box (lead_slow \Rightarrow \Diamond_{\leq x_1} lead_kinematics_info)$</td>
</tr>
<tr>
<td>ACC Controller Component</td>
<td>$\psi_2 : \Box (lead_kinematics_info \Rightarrow \Diamond_{\leq x_2} apply_brake)$</td>
</tr>
</tbody>
</table>

The requirement decomposition pair is valid if and only if PLTL formula $\psi_1 \land \psi_2 \Rightarrow \phi_1$ is valid.

Due to the parameters, this reduces to constraint computation.
Given a PLTL formula $\phi$, we want to find the representation of the solution region in the form of a constraint.
Validity Region Characteristics...

<table>
<thead>
<tr>
<th>Feature/Component Name</th>
<th>PLTL Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC Feature</td>
<td>$\phi_1 : \Box(\text{lead_slow} \Rightarrow \Diamond \leq 500 \text{ apply_brake})$</td>
</tr>
<tr>
<td>Sensor Component</td>
<td>$\psi_1 : \Box(\text{lead_slow} \Rightarrow \Diamond \leq_{x_1} \text{ lead_kinematics_info})$</td>
</tr>
<tr>
<td>ACC Controller Component</td>
<td>$\psi_2 : \Box(\text{lead_kinematics_info} \Rightarrow \Diamond \leq_{x_2} \text{ apply_brake})$</td>
</tr>
<tr>
<td>ACC Controller Component</td>
<td>$\phi_2 : \Box(\text{apply_brake} \Rightarrow \Diamond \leq_{y_2} \text{ host_decel})$</td>
</tr>
<tr>
<td>Engine Controller Component</td>
<td>$\psi_3 : \Box(\text{apply_brake} \Rightarrow \Diamond \leq_{y_1} (\text{thrtl_adj}) \lor (\text{brake_adj}))$</td>
</tr>
<tr>
<td>Brake Controller Component</td>
<td>$\psi_4 : \Box(\text{thrtl_adj} \Rightarrow \Diamond \leq_{y_2} \text{ host_decel})$</td>
</tr>
<tr>
<td>Brake Controller Component</td>
<td>$\psi_5 : \Box(\text{brake_adj} \Rightarrow \Diamond \leq_{y_3} \text{ host_decel})$</td>
</tr>
</tbody>
</table>

$(y_1 + y_2 \leq 200) \land (y_1 + y_3 \leq 200)$

$(y_1 + y_2 \leq 200) \lor (y_1 + y_3 \leq 200)$
Validity Region: Some Takeaways

- Validity region can be bounded/unbounded, convex/non-convex
- Different points in the region can satisfy different constraints - we need to combine them
- Does not merely depend on underlying component data-flow graph
- Linear predicates looks good representation
  - but are they enough?
Emptiness, Universality and Finiteness of Validity Region

+ Unsuitability of Linear Predicates for Representing Validity Region

Information Processing Letters
# Emptiness, Universality and Finiteness Problems

<table>
<thead>
<tr>
<th>Formula</th>
<th>Emptiness</th>
<th>Universality</th>
<th>Finiteness</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$F$</td>
<td>$S_\Phi$</td>
<td>$V_\Phi$</td>
</tr>
<tr>
<td>PLTL$\Box$</td>
<td>${\Phi(\alpha_0)}$</td>
<td>$\Leftrightarrow$</td>
<td>sat</td>
</tr>
<tr>
<td>PLTL$\Diamond$</td>
<td>${\Phi}$</td>
<td>$\Leftrightarrow$</td>
<td>sat</td>
</tr>
<tr>
<td>PLTL</td>
<td>${\Phi(0_y)}$</td>
<td>$\Leftrightarrow$</td>
<td>sat</td>
</tr>
</tbody>
</table>

$\alpha_0$: Parameter valuation assigning 0 to all parameters.
$0_x$: Partial parameter valuation assigning 0 to all members of $X$.
$0_y$: Partial parameter valuation assigning 0 to all members $Y$.
$0_y \setminus \{y\}$: Partial parameter valuation assigning 0 to all members of $Y \setminus \{y\}$.
$\Leftrightarrow$: Solution provides both necessary and sufficient condition.
$\Rightarrow$: Solution provides only sufficient condition.
sat: Satisfiability check required of a member formula.
val: Validity check required of a member formula.

- **Defined** *parameter abstraction operation* for PLTL

\[
\Phi = \Box \leq y p_1 \land \Box \leq 10 p_2 \\
\widetilde{\Phi} = \Box p_1 \land \Box \leq 10 p_2
\]

- **Improved Complexity**

\[
O(c_{\Phi} 2^{n_{\Phi}}) \quad \text{Our complexity:} \quad O(c_{\Phi}^{k_{\Phi} + 1} k_{\Phi}^{k_{\Phi}} 2^{n_{\Phi}(k_{\Phi} + 1)}) \quad \text{Earlier:}
\]
Bounded-Response Constraint Extraction Method

Under Submission to Journal
A Scalable Method for a widely used Requirement Pattern

At higher levels most of the requirements are based on a specific pattern... *bounded-response*

- ϕ: Antecedent formula
- ψ: Succedent formula
- x: Parameter or constant

We consider validity checks of requirement decomposition pairs using this pattern

Bounded-response formula in PLTL

\[
\square (\phi \Rightarrow \Diamond \leq x \psi)
\]
**Constraint Extraction Method**

- Reasoning over temporal formulae reduced to Boolean reasoning... hence scalable

- And-or tree constructed from formulae
  - We have defined a notion of an irreducible cover for Boolean formulae

- Assign path constraints

- Final constraint is conjunction/disjunction of path constraints

---

Let $\gamma$ and $\gamma_1, \ldots, \gamma_n$ be Boolean formulae. An irreducible cover of $\gamma$ is a minimal subset $H$ of $\{\gamma_1, \ldots, \gamma_n\}$ such that $\gamma \Rightarrow \bigvee_{\gamma' \in H} \gamma'$ is valid.
Example: Bounded-Response Tree

\[ \Phi = \Box (a_1 \lor a_2 \Rightarrow \Diamond_{\leq 10} b_1 \lor b_2) \]

\[ \psi_1 = \Box (a_1 \Rightarrow \Diamond_{\leq x_1} c_1 \lor c_2 \lor b_2) \]
\[ \psi_2 = \Box (a_2 \Rightarrow \Diamond_{\leq x_2} c_1) \]
\[ \psi_3 = \Box (c_1 \Rightarrow \Diamond_{\leq x_3} b_1) \]
\[ \psi_4 = \Box (c_2 \Rightarrow \Diamond_{\leq x_4} b_1) \]
\[ \psi_5 = \Box (c_2 \Rightarrow \Diamond_{\leq x_5} b_2) \]
\[ \psi_6 = \Box (c_2 \Rightarrow \Diamond_{\leq x_6} c_4) \]
\[ \psi_7 = \Box (a_1 \land a_2 \Rightarrow \Diamond_{\leq x_7} c_3) \]

\[
\{((x_1 + x_3 \leq 10) \land (x_1 + x_4 \leq 10)) \lor ((x_1 + x_3 \leq 10) \land (x_1 + x_5 \leq 10)) \lor ((x_1 + x_3 \leq 10) \land \text{false})\} \land (x_2 + x_3 \leq 10)
\]
Corner Point Constraint
Extraction Method

Preliminary Version in DATE 2010
Journal Version under Submission
A General Constraint Extraction Method for PLTL

- Suitable for complex temporal properties
- We focus on PLTL fragments and their geometric properties
- PLTL Global properties are downward-closed
- Downward Closed Region have finite number of Corner-Points
  - Include Point-at-infinity
  - Constraint definition using Corner-Points

Monotonicity of PLTL operators

\[ \square \leq m \phi \Rightarrow \square \leq (m-1) \phi \]
\[ \Diamond \leq m \phi \Rightarrow \Diamond \leq (m+1) \phi \]

\[ \bigvee_{i=1}^{3} x_1 \leq \alpha_i(x_1) \land x_2 \leq \alpha_i(x_2) \]
Algorithm Overview

- **Prune and Search Approach**
  - Find a corner point
  - Partition the further search

- **Search Step**
  - Obtain a farthest useful valuation along diagonal starting from a base
  - Decide sub-set of parameters for which max limit is reached
  - Fix them and re-iterate till all parameters are over

- **Prune Step**
  - Partition and identify the region(s) where no corner-point can lie
  - Adjust new base valuation so that those regions get ignored from later search
  - Repeat Search step recursively for them
Demonstration of the Methodology
Integrated Time-Budgeting Methodology

- NuSMV for LTL checks
- Yices for constraint solving
- Eclipse, Java
Case Studies

- Adaptive Cruise Control, Collision Mitigation
- 120+ feature and component properties
- 100+ add-on constraints in the exploration
- Budgeting for 3 feature combinations: ACC only, CM only and ACC-CM
Some Results

### Brake Controller Component

<table>
<thead>
<tr>
<th>Component</th>
<th>Parameter</th>
<th>Only ACC</th>
<th>Only CM</th>
<th>ACC-CM</th>
</tr>
</thead>
<tbody>
<tr>
<td>w_1</td>
<td>200</td>
<td>160</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>w_2</td>
<td>70</td>
<td>-</td>
<td>-</td>
<td>70</td>
</tr>
<tr>
<td>w_3</td>
<td>100</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>w_4</td>
<td>100</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>w_5</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>10</td>
</tr>
<tr>
<td>w_6</td>
<td>150</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

### Electronic Brake Control

<table>
<thead>
<tr>
<th>Component</th>
<th>Parameter</th>
<th>Only ACC</th>
<th>Only CM</th>
<th>ACC-CM</th>
</tr>
</thead>
<tbody>
<tr>
<td>w_11</td>
<td>35</td>
<td>20</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>w_12</td>
<td>100</td>
<td>80</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>w_13</td>
<td>30</td>
<td>20</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>w_14</td>
<td>10</td>
<td>40</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>w_15</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>w_16</td>
<td>60</td>
<td>-</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>w_17</td>
<td>60</td>
<td>-</td>
<td>60</td>
<td></td>
</tr>
</tbody>
</table>

![Graph](image_url)
Summary of Contributions

● A Hierarchical Time-Budgeting Methodology
  □ Integrates all of the below techniques

● Emptiness, Universality and Finiteness Problems for PLTL
  □ Non-triviality of the solution region

● Bounded-Response Constraint Extraction Method
  □ A specially tuned method for a widely used requirements pattern

● Corner Point Constraint Extraction Method
  □ Complex temporal relationships

● Case Studies
  □ Tool framework and demonstration on automotive features ACC and CM
Our Publications


References


Thank You