Temporal Engineering™ Manifesto

Gordon Morrison, Ann-Marie Scarborough, Tom Bowman

Temporal Engineering is the logical ordering of events based on an anticipated occurrence.

In Temporal Engineering:

Time dominates process – Process dominates data
Time → Process → Data → Implementation

This approach is an architectural implementation that improves the quality of software

TIME

In Temporal Engineering, Time is the anticipated dynamic aspect of the execution of the application.

Temporal Engineering manages and maintains temporal components.

These components are:

Engine(s) → Rule(s) → Step(s) → Elements

An important aspect of time management is found in the use of Temporal points/pointers. In Temporal Engineering there are three types: Static Temporal Pointers, Dynamic Temporal Pointers, and Static Temporal Points.

- **Static Temporal Pointers** are found in the design of the Rules/Steps as well as the Engine.
- **Dynamic Temporal Pointers** are the result of the Engine accessing a Static Temporal Pointer (T/F Behaviors) for the next iteration of the Engine.
- **Static Temporal Points** are a part of the sequential design for the Rules/Steps.

Engines

A Temporal Engineering Σngine™ is the driving force through discrete time.

- The **event-driven Temporal Engine™** manages the selection of Rules.
- The **Temporal Engine manages** the selection of a Step within a Rule.
- The **selection** of the Static Temporal Pointer in the Function produces the true or false result:
    - **True Decision**: Selects the True Behavior Static Temporal Pointer for control-flow followed by the next True Dynamic Temporal Pointer.
    - **False Decision**: Selects the False Behavior Static Temporal Pointer for control-flow followed by the next False Dynamic Temporal Pointer.
- **A Dynamic Temporal Pointer** is the simplest Element being a cardinal value associated with each Step as a "step in time".
- **Static Temporal Pointers** are accessed to produce a Dynamic Temporal pointer.

Rules

A Temporal Engineering Rule is a named ordered sequence of Steps defining control-flow.

- Rule Names are chosen by the Subject Matter Expert (SME).
- A Rule is determined by an SME.
- A Rule has one or more Steps.
- A Rule can only be entered at Step Zero.
- A Rule can iterate on any Step.
- A Rule can exit from any Step.
- Static Temporal Pointers are only contained in Rules/Steps.

Steps

A Temporal Engineering Step is an increment in discrete time for a process.

- A Temporal Engineering Step is made up of Elements.
- A Temporal Engineering Step prefers but is not limited to seven Elements.
- The number of Steps is defined by the SME.
- Each Step executes a State Test, then executes a True or False behavior based on the State Test and then moves a Static Temporal Pointer to the next respective True/False Step to be executed.
- Each Step provides an optional Trace.

Elements

A Temporal Engineering Element is the atomic breakdown of a step-in-time.

The preferred Elements of a Step consist of a:

1. Rule-name/Step (a Static Temporal Point)
2. State test reference
3. True behavior reference
4. Next true step (Static Temporal Pointers)
5. False behavior reference
6. Next false step (Static Temporal Pointers)
7. Trace value

Also:

- Each True and False decision has a next logical Static Temporal Pointer as part of control-flow.

- A Temporal Engineering State test reference result is true or false.
- Element names and Trace values are chosen by the SME.

PROCESS

Process is the implementation of the dynamic anticipated SME-defined order.

- **Architecture first** – In Temporal Engineering Architecture is the underlying structure of the anticipated sequence of the logic.
- **The Subject matter expert** (SME) drives architectural creation.
- **Every process thread** must be sequenced.
- **Process doesn't include** manipulating data in the control-flow.
- **Data** is manipulated outside of the control-flow.

DATA

Data is the anticipated dynamic value of the process.

- Data is **manipulated** in simple coherent routines.
- Data **processing** contains no control-flow logic.
- Manipulation **logic** is for data processing only, not control-flow.
- Most data are **created in a temporal domain** and must be processed in that order.
- **Asynchronous data** is collected to be ordered by process to create value.

IMPLEMENTATION

Implementation is the discipline for the execution of Temporal Engineering providing fundamental structures for an authored approach to writing software.

Temporal Engineering is made up of sequential Rules with the following restrictions:

1. Step zero is the only entry point to a Rule.
2. Rules can exit at any step.
3. Rules can iterate within a Rule.
4. Rules can call other Engines as a subroutine.
5. Rules can call other Rules as a subroutine.
6. Temporal Engineering Steps do not transition to another Step, Steps are created as a sequence of binary truths.
7. In routines (the Steps) are coherent - do one thing, do it well.