Funktionale Sicherheit und Code-Generatoren

Die ISO-Norm 26262 und Funktionale Sicherheit stehen derzeit im Fokus der industriellen Softwareentwicklung. Um in der Softwareentwicklung hohen oder höchsten Qualitätsanforderungen zu genügen, erwartet die Norm ein systematisches Vorgehen sowohl in Prozessen, als auch im Erstellen von Prozessergebnissen. Im Sourcecode sollen interne Coding-Guidelines und externe Normen wie zum Beispiel MISRA beachtet werden. Designdokumente sollen an Dokumentvorlagen ausgerichtet sein. Teilweise werden ganze Software-Module oder Komponente vollständig über eine Norm definiert. Sogar der Umfang der einzusetzenden Programmiersparache (z.B. Embedded C++) wird festgeschrieben. Normen über Normen! "Wer soll da noch durchsteigen?" fragt sich der Entwickler.

Betrachtet man die letzten Jahre der Softwareentwicklung in der Industrie mit etwas Abstand, erkennt man, dass sich die zur Anwendung kommenden Paradigmen geändert haben und noch ändern. Softwaresysteme die (z.B. aus Gründen der funktionalen Sicherheit) hohen Qualitätsanforderungen genügen müssen, verlangen nach systematischem Vorgehen. Systematisches Vorgehen erfordert Normen und Normen führen zu genormten und automatisch bearbeitbaren Ergebnissen. - Das ist nicht nach dem Geschmack von kreativen Softwareentwicklern. Wo bleibt da der Mensch? Da kann man ja gleich alles den Maschinen über lassen! - Ja genau so ist es! - Fast!

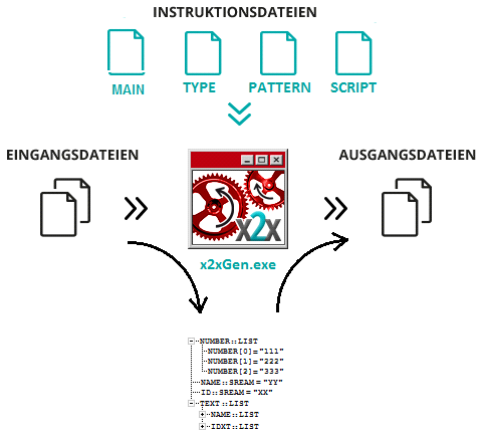
"Kreativität" ist nichts für Maschinen und stupides Ausführen immer wieder gleicher Vorgänge ist nichts für Menschen. Das schreit doch geradezu nach Arbeitsteilung. Routine für die Maschine - Leben für den Menschen. Genau die Veränderung hin zu dieser Aufteilung findet seit geraumer Zeit statt.

Es ist schon lange nicht mehr die Hauptaufgabe der hiesigen Softwareentwickler Programmcode zu schreiben. Vielleicht mal einen komplizierten Algorithmus zum Ausprobieren "reinhacken". Das war's auch schon. Die überwiegende Tätigkeit eines heutigen Softwareentwicklers ist das Spezifizieren bzw. Erstellen von Designdokumenten. Idealer Weise ist das Design wieder normiert und man kann daraus direkt den Sourcecode generieren. Jetzt sind wir beim Thema: "Code-Generatoren".

"Kann man Code-Generatoren trauen? Wer weiß was da rauskommt? Den 'Spagetti-Code' versteht doch eh' keiner!? 'Funktionale Sicherheit' verlangt nach Kontrolle! Und wie soll man kontrollieren worauf man keinen Einfluss hat? - Das ist nicht die gewollte Aufteilung! Die Maschine generiert

irgendwas und dann muss doch wieder der Mensch ran und kontrollierten, ob der Mist auch richtig ist?".

Das sind Fragen und Aussagen die der Vergangenheit angehören, als Code-Generatoren noch unbeeinflussbare Bestandteile von Software-Modellierungsprogrammen waren. Moderne industrielle Softwareentwicklung sieht ganz anders aus! Nicht nur "was" ein Generator generiert, bestimmt man heute selbst, sondern auch "woraus" er es generiert. Und es wird nicht nur Code generiert, sondern auch Dokumentvorlagen oder gar fertige Dokumente und Berichte. Ein Generator kann auch den Output eines anderen Generators kontrollieren und ein Prüfprotokoll dazu erstellen.



"Oh Gott, oh Gott! Und was ist, wenn der Generator der kontrollieren soll selbst einen Fehler hat!? Na? Was dann??" Na was soll schon sein. Kontrolle kann immer nur die Wahrscheinlichkeit erhöhen, dass ein bestimmtes Ergebnis auch so ist, wie man es erwartet. Und die Wahrscheinlichkeit, dass ein Mensch z.B. beim Übertagen von Hunderten von Datensätzen in Sourcecode einen Fehler macht und ein zweiter Mensch, der dies kontrollieren soll, diesen nicht findet, ist ungleich höher als, dass ein Generator beim Übertragen einen Fehler macht und ein zweiter Generator diesen nicht findet. Schließlich haben Generatoren haben den entscheidenden Vorteil, dass sie immer gleich arbeiten und, dass man sie testen kann.

Genau 'das' ist mit der Veränderung der Paradigmen gemeint. Es wird nicht mehr stupide immer und immer wieder ähnlicher Programmiercode erstellt, sondern das geistige Potential wird dazu eingesetzt Modelle und Modelltransformationen zu entwickeln und über Generatoren instanziierte Modelle in Ergebnisse umzusetzen. Zu überlegen, wie man die Qualität der Ergebnisse automatisiert kontrollieren kann oder automatische erstellte Berichte so aufbereiten kann, dass der Mensch leicht den Überblick halten kann.

" 'Modelle und Modelltransformationen' das klingt schon wieder sehr abgehoben! 'Modellbasierte Softwareentwicklung' und dann noch 'Meta-Modelle' und schon brauchst Du ein zweijähriges Studium, um das zu verstehen! Und überhaupt, so eine Code-Generator schreibt man auch nicht in ein paar Tagen, sondern vielleicht in Wochen oder Monaten."

Lassen Sie sich überraschen. Die Praxis hat es längst bewiesen. In Realität ist alles viel einfacher. Ein einfaches Excelsheet kann schon das Modell für eine Softwarekomponente sein und den Generator dazu schreiben Sie im "Handumdrehen" mit X2X.

Und Sie werden sehen, so kann man ein Fundament für 'Funktionale Sicherheit' legen und den Kopf dafür frei halten, was wirklich wichtig ist.

Mehr zu Thema:

http://www.sss.de/x2x

http://www.sss.de/x2x-downloads?file=files/triple-s/downloads/downloads_X2X_2015/Modellbasierte%20Softwareentwicklung.pdf

http://www.sss.de/x2x-downloads?file=files/triple-s/downloads/downloads_X2X_2015/Sourcecode_generieren.pdf