Číslo 1, ročník XIX, duben 2019

FORMÁLNÍ METODY VE VÝVOJI SOFTWARU BEZPEČNOSTNĚ KRITICKÉHO SYSTÉMU

FORMAL METHODS IN DEVELOPMENT OF SAFETY CRITICAL SYSTEM SOFTWARE

Michal Bubeník[1]

Anotace: Článek se zaměřuje na využití formálních metod při vývoji softwaru pro bezpečnostně kritické systémy v dopravě. Takové systémy mají narůstající složitost a musejí vyhovovat určitým požadavkům na kvalitu. Formální metody jsou zařazeny do procesu vývoje softwaru. Použití vybrané formální metody Event-B je demonstrováno na specifikaci chování železničního přejezdového zabezpečovacího zařízení.

Klíčová slova: software, formální metody, bezpečnost.

Summary: This article focuses on using formal methods in development of safety critical system software in transport. Such system has an increasing complexity and has to meet certain quality requirements. Formal methods are set in the software development process. The use of formal method Event-B is demonstrated on a railway level crossing protection equipment specification.

Key words: software, formal methods, safety.

ÚVOD

Na aktuálně vyvíjené bezpečnostně kritické systémy pro dopravu narůstají nároky na jejich složitost, přičemž musí být dodržena jejich kvalita. Případná chyba v takovém systému může mít za následek ztráty na životech či značné materiální škody. V neposlední řadě je s narůstající složitostí také nezbytné zvyšovat efektivitu vývoje v odpovídající míře, užíváním vhodných technik vývoje. Komplexní systémy vyžadují podrobnou a úplnou specifikaci, podle které jsou následně navrhovány a verifikovány.

Formální metody jsou jednou z možných technik pro zvýšení kvality vývoje softwaru. Jejich ukotvení je uvedeno například v DO-178C (9) pro letecké systémy či v ČSN EN 50128 (4) pro drážní systémy.

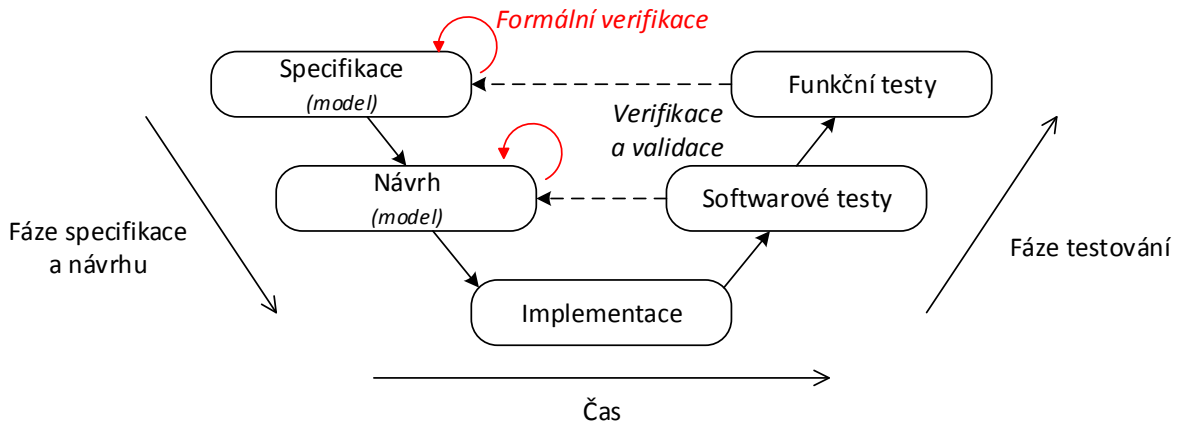
Pod formálními metodami si lze představit matematicky přesné techniky a nástroje pro specifikaci, návrh a verifikaci systému. Existuje řada formálních metod, například B metoda (1) a na ni navazující Event-B (5), Z metoda (2), TLA+ (6) nebo Petriho sítě (7). Volba metody závisí na charakteru řešené problematiky, na poskytované podpoře od jejích tvůrců a případně také na jejím dalším rozvoji do budoucna. Článek se dále zaměřuje na formální metodu Event-B (5) a příklad její praktické aplikace.

[1] Ing. Michal Bubeník, Univerzita Pardubice, Dopravní fakulta Jana Pernera, Katedra elektrotechniky, elektroniky a zabezpečovací techniky v dopravě, Studentská 95, 532 10 Pardubice, Tel.: +420 730 168 132, E-mail: michal.bubenik@student.upce.cz

1. ROLE FORMÁLNÍCH METOD VE VÝVOJI SOFTWARU

1.1 Vliv na proces vývoje

Jednou z možností popisu procesu vývoje softwaru je tzv. „V-model“. Diagram procesu znázorňuje Obr. 1.



Zdroj: Autor

Obr. 1 – Základní V-diagram procesu vývoje softwaru

Proces je rozdělen na fázi specifikace a návrhu („sestupná fáze“) a na fázi testování a ověřování („vzestupná fáze“). Typickou součástí návrhu a specifikace je modelování požadovaného chování systému. Takový model lze zpočátku tvořit zjednodušeně (abstraktně) a v dalším postupu vývoje zvyšovat jeho podrobnost, směrem k požadovanému cíli.

Hlavním smyslem využití formálních metod je možnost provést matematicky přesnou verifikaci modelu. Verifikační aktivity na Obr. 1 jsou tím přesunuty do „sestupné fáze“, kde je prováděna verifikace již ve stádiu specifikace a návrhu (oproti testování výsledného, implementovaného produktu).

1.2 Vliv na styl zápisu specifikace

Formální zápis lze významově pojmout v axiomatickém nebo operačním stylu. Axiomatický styl staticky popisuje vlastnosti systému. Podle takového popisu pak lze do výsledného programu zakomponovat například kontrolní podmínky. Operační styl řeší sekvenční chování systému a spíše tak zachycuje logický průběh vykonávání výsledného programu.

Specifikace se vyznačuje jednoznačnou syntaxí a sémantikou. Eliminuje tak možnou chybnou interpretaci při čtení či výkladu, v porovnání s běžným jazykem.

2. EVENT-B

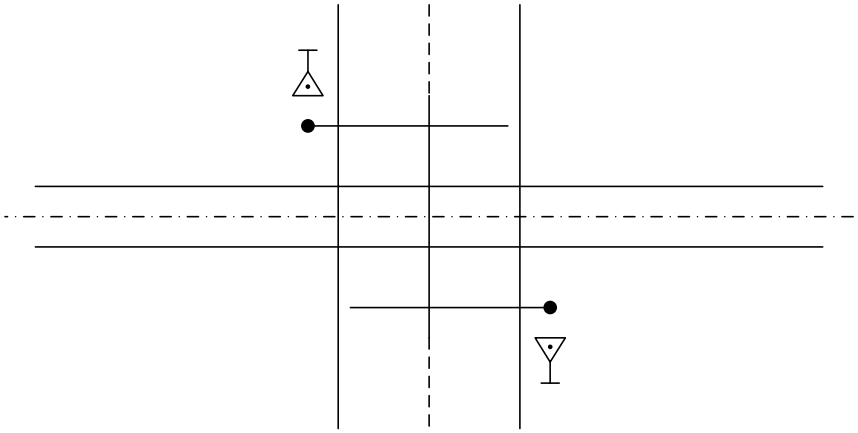
Event-B je formální metoda pro modelování a analýzu. Navazuje na původně vyvinutou metodu B (1), která pracuje s tzv. „Abstraktním automatem“ („Abstract-machine“). Zaměřuje se na obecné modelování požadovaného výsledku, které je následně postupně upřesňováno do finální, podrobné podoby.

Použití této metody je demonstrováno na specifikaci přejezdového zabezpečovacího zařízení. Pro práci s Event-B je dostupná platforma Rodin (5), což je vývojové prostředí

poskytující podporu pro tvorbu specifikace a matematický důkaz. Platformu je možné dále rozšiřovat o další funkce, například o nástroje pro formální dokazování, vizualizaci a modelování.

2.1 Přejezdové zařízení v Event-B

Při modelování funkčního chování přejezdového zařízení je vycházeno z ČSN 34 2650 (3) a zvolena je jednoduchá situace, zachycující základní principy. Řešenou situací je přejezd se světelnými výstražníky, celými závorami a bezpečně řešeným spuštěním i ukončením výstrahy (PZS 3). Modelovou situaci ilustruje Obr. 2.



Zdroj: Autor

Obr. 2 – Modelová situace železničního přejezdu

2.1.1 Abstraktní model

Pro modelování je použit zápis definovaný pro Event-B na platformě Rodin. Nejprve je vytvořen základní, abstraktní model požadovaného chování. Prvním krokem je definování proměnných *train_approach* pro přibližující se drážní vozidlo k přejezdu a *warn_active* pro aktivní výstrahu na přejezdu. Dále jsou definovány invarianty, což jsou tvrzení, která musejí platná pro každý dosažitelný stav.

```
MACHINE
    lx0
VARIABLES
    train_approach
    warn_active
INVARIANTS
    inv1:   train_approach ∈ BOOL not theorem
    inv2:   warn_active ∈ BOOL not theorem
    inv3:   train_approach = TRUE ⇒ warn_active = TRUE theorem
```

Zdroj: Autor

Obr. 3 – Abstraktní model – definice proměnných a invariantů

První dva invarianty jsou potřebné pro určení typu proměnných jako booleovský. Poslední invariant definuje vztah mezi přibližujícím se vlakem a výstrahou na přejezdu. Základním požadavkem je spuštění výstrahy na přejezdu, pokud se k němu blíží drážní vozidlo. Záměrně je použita implikace (nikoliv ekvivalence), protože výstraha může být spuštěna i z dalších příčin (například porucha detekčního prvku).

Pro změnu stavu modelu slouží události. Každá událost může mít definovanou podmínku jejího použití (tzv. „guard“) a v rámci události se vykonají definované akce. Pro přibližování vlaku je definována událost *set_train_approaching* a pro opačnou situaci, vzdalování, je událost *set_train_leaving*. V rámci těchto událostí se mění stav proměnných *train_approach* a *warn_active* (viz Obr. 4), což simuluje přibližování vozidla a stav výstrahy na přejezdu.

```
EVENTS
    INITIALISATION:     not extended ordinary ›
    THEN
        act1:   train_approach ≔ FALSE ›
        act2:   warn_active ≔ FALSE ›
    END

    set_train_approaching:      not extended ordinary ›
    WHERE
        grd1:   train_approach = FALSE not theorem ›
    THEN
        act1:   train_approach ≔ TRUE ›
        act2:   warn_active ≔ TRUE ›
    END

    set_train_leaving:      not extended ordinary ›
    WHERE
        grd1:   train_approach = TRUE not theorem ›
    THEN
        act1:   train_approach ≔ FALSE ›
        act2:   warn_active ≔ FALSE ›
    END
```

Zdroj: Autor

Obr. 4 – Abstraktní model – události

2.1.2 Podrobnější model

V dalším kroku je vytvořený model dále upřesňován. Konkrétně je lépe popsán postupný pohyb drážního vozidla, od ovlivnění detekčního prvku pro spuštění výstrahy, přes detekování průjezdu drážního vozidla přejezdem, až po vzdalování a navrácení do základního stavu. K tomu slouží proměnná *train_position*, která nabývá jedné z hodnot: výchozí hodnota *NoTrain*, hodnota *Approaching* pro přibližování vozidla, *Passing* pro míjení přejezdu a *Leaving* pro vzdalování vozidla. Tyto stavy jsou definovány jako konstanty v kontextu *context0*, viz Obr. 5.

```
CONTEXT
    context0    ›
SETS
    TrainPosition    ›
CONSTANTS
    Approaching  ›
    Passing  ›
    Leaving  ›
    NoTrain  ›
AXIOMS
    type:   partition(TrainPosition,{NoTrain},{Approaching},{Passing},{Leaving})
END
```

Zdroj: Autor

Obr. 5 – Podrobnější model – kontext

Pro jednotlivé konstanty je nutné stanovit jejich nerovnost, což se provede definováním výčtové množiny tvořené těmito konstantami (viz „partition“ na Obr. 5). Zápisem *partition(...)*

se definuje, že daná množina je tvořena vyjmenovanými prvky, které jsou vzájemně rozdílné. Kontext je zpřístupněn modelu, viz vazba „SEES“ na Obr. 6.

Dále je vytvořen druhý model, provázaný s tím předchozím. Jednak k tomu v Event-B slouží vazba „REFINES“ (viz Obr. 6), a potom také definice invariantu stanovujícího vztah mezi vybranými proměnnými obou modelů (viz invariant *link_approach*). Tento invariant mapuje hodnotu *TRUE* na polohu vozidla *Approaching* a *Passing*. Modely jsou tedy provázány přes rozlišení polohy vozidla.

```
MACHINE
    lx1
REFINES
    lx0
SEES
    context0
VARIABLES
    train_position
    annul
    warn_active
INVARIANTS
    inv1:   train_position ∈ {NoTrain,Approaching,Passing,Leaving} not theorem
    inv2:   annul ∈ BOOL not theorem
    inv3:   warn_active ∈ BOOL not theorem
    link_approach:  train_approach = TRUE ⇔ train_position ∈ {Approaching,Passing}
    inv4:   annul = TRUE ⇒ warn_active = FALSE not theorem
    inv5:   train_position ∈ {Approaching,Passing} ⇒ warn_active = TRUE not theorem
```

Zdroj: Autor

Obr. 6 – Podrobnější model – definice proměnných a invariant

Dále je přidána také proměnná *annul* pro stav anulace, tedy vyloučení vlivu detekčního prvku na spuštění výstrahy pro opačný směr jízdy. V modelu přeneseně slouží pro simulaci závislosti stavu anulace na stavu výstrahy, což definuje invariant *inv4*. Poslední invariant, *inv5*, definuje aktivní výstrahu na přejezdu, pokud se drážní vozidlo k přejezdu blíží nebo jej právě projíždí.

Simulování změny jednotlivých stavů modelu je opět zajištěno pomocí událostí (viz Obr. 7). Tentokrát je nadefinována událost pro detekování přibližujícího se vozidla *train_detected*, událost pro míjení přejezdu vozidlem *train_passing* a událost pro opuštění přejezdu a vzdalování *train_passed*. Události jsou doplněny ještě návratem modelu do základního stavu, pomocí události *default_state*.

Jízda vozidla probíhá přes definované stavy vždy sekvenčně, a proto je ke každé události definována podmínka („guard“), která tuto sekvenčnost zajistí s pomocí proměnné *train_position*. Tato proměnná se při každé události mění a reprezentuje tak aktuální polohu vozidla.

Události *train_detected* a *train_passed* upřesňují příslušné události z abstraktního modelu pomocí vazby „REFINES“, viz Obr. 7. Představují tak jejich další provázání.

Na závěr jsou do dílčích událostí v sekci „EVENTS“ doplněny akce pro nastavení stavu výstrahy a stavu anulace. V případě přibližujícího se či míjejícího drážního vozidla je požadována aktivní výstraha. Po opuštění přejezdu vozidlem je nastavena anulace a zrušena výstraha. Tím je naplněn invariant *inv4* a *inv5*, viz Obr. 6.

Model lze dále rozšiřovat například o uvažování poruch, nouzový a poruchový stav přejezdového zařízení, pozitivní signalizaci nebo měření přibližovací a vyklizovací doby.

```
EVENTS
    INITIALISATION:     not extended ordinary ›
    THEN
        act1:   train_position ≔ NoTrain ›
        act2:   annul ≔ FALSE ›
        act3:   warn_active ≔ FALSE ›
    END

    train_detected:     not extended ordinary ›
    REFINES
        set_train_approaching
    WHERE
        grd1:   train_position ∉ {Approaching,Passing} not theorem ›
    THEN
        act1:   train_position ≔ Approaching ›
        act2:   warn_active ≔ TRUE ›
        act3:   annul ≔ FALSE ›
    END

    train_passing: not extended ordinary ›
    WHERE
        grd1:   train_position = Approaching not theorem ›
    THEN
        act1:   train_position ≔ Passing ›
        act2:   warn_active ≔ TRUE ›
        act3:   annul ≔ FALSE ›
    END

    train_passed:   not extended ordinary ›
    REFINES
        set_train_leaving
    WHERE
        grd1:   train_position = Passing not theorem ›
    THEN
        act1:   train_position ≔ Leaving ›
        act2:   annul ≔ TRUE ›
        act3:   warn_active ≔ FALSE ›
    END

    default_state: not extended ordinary ›
    WHERE
        grd1:   train_position = Leaving not theorem ›
    THEN
        act1:   train_position ≔ NoTrain ›
        act3:   annul ≔ FALSE ›
    END
```

Zdroj: Autor

Obr. 7 – Podrobnější model – události

Pro kontrolu modelu byl z několika možností vybrán nástroj „ProB“ (8). Pomocí tohoto nástroje je možné automatizovaně provést kontrolu modelu, zahrnující detekci absorpčních stavů, porušení invariant a teorémů a detekci chyb ve stavovém prostoru.

Výhodou nástroje ProB je existence certifikátu, který potvrzuje splnění požadavků normy EN 50128 na třídu nástrojů T2 (nástroje pro podporu verifikace a testování).

Platforma nabízí možnosti rozšíření o doplňky na prokazování a vizualizaci, se kterými je možné interaktivně provádět dokazování invariant či provádět některé uskutečněné kroky

v grafické podobě (grafické modelování). Díky otevřenosti platformy lze předpokládat, že se bude nadále rozvíjet a rozšiřovat nabízené funkce.

ZÁVĚR

Cílem tohoto článku bylo ukázat roli formálních metod ve vývoji bezpečnostně kritického systému v dopravě.

Formální metody umožňují jednoznačný a matematicky ověřitelný popis systému, ve kterém je eliminován vznik nejasností plynoucích z běžného jazyka. Požadavky na systém psané běžným jazykem však stále mají své místo ve vývojovém procesu, protože znalost formálních metod je obvykle omezena na úzký okruh lidí. To ztěžuje například komunikaci se zákazníkem.

V druhé části článku bylo demonstrováno použití metody Event-B na situaci železničního přejezdového zabezpečovacího zařízení. Modely byly záměrně zjednodušeny, jelikož cílem bylo ukázat princip metody. Aplikace metod například na celý staniční či vlakový zabezpečovací systém obnáší modelování značného rozsahu. Úsilí vložené do modelování se však vyplatí z hlediska udržení kvality systému po provedení změn, což lze podložit automatickými kontrolami modelu. Přesto je aplikace formálních metod málo rozšířená.

POUŽITÁ LITERATURA

(1) Abrial J.R. *The B tool*. In: Bloomfield R.E., Marshall L.S., Jones R.B. (eds) VDM '88 *VDM — The Way Ahead*. VDM 1988. Lecture Notes in Computer Science, vol 328. Springer, Berlin, Heidelberg.

(2) Bowen J. *Formal Specification and Documentation Using Z: A Case Study Approach.* International Thomson Computer Press, 1996. 302 s. ISBN 978-18-5032-230-6.

(3) ČSN 34 2650. *Železniční zabezpečovací zařízení – Přejezdová zabezpečovací zařízení.* ed. 2. Praha: Úřad pro technickou normalizaci, metrologii a státní zkušebnictví, 2010.

(4) ČSN EN 50128. *Drážní zařízení – Sdělovací a zabezpečovací systémy a systémy zpracování dat – Software pro drážní řídicí a ochranné systémy*, ed. 2. Praha: Úřad pro technickou normalizaci, metrologii a státní zkušebnictví, 2012.

(5) Event-B.org [online]. [cit. 2019-01-12]. Dostupné z: <http://www.event-b.org/install.html>.

(6) Lamport, L. *A High-Level View of TLA+* [online]. 2018 [cit. 2019-01-14]. Dostupné z: <http://lamport.azurewebsites.net/tla/high-level-view.html>.

(7) Petri, C.A., Reisig, W. Petri net. *Scholarpedia* [online]. 2008 [cit. 2018-12-29]. Dostupné z: <http://www.scholarpedia.org/article/Petri_net>.

(8) The ProB Animator and Model Checker [online]. [cit. 2019-01-12]. Dostupné z: <https://www3.hhu.de/stups/prob/index.php/The_ProB_Animator_and_Model_Checker>

(9) RTCA DO-178C. *Software Considerations in Airborne Systems and Equipment Certification*. Version C. 2011.