Digital Twin. Часть 2. Инструментальный Цифровой двойник

В первой части [DT1] были рассмотрены проблемы современного «Цифрового двойника» \ Digital Twin (ЦД \ DT) и общие подходы к его идентификации, в первую очередь, его «Трехкомпонентный состав DT» («три кита» двойника): реальный объект (физический, «физик»), его модель (собственно сам DT) и обратная связь – как передача эксплуатационных данных объекта в контекст его модели (в идеале двухсторонний обмен). В идеале должен быть не только двухсторонний обмен по эксплуатационным данным, но и обмен по состоянию самой структуры объектов (синхронизация структуры), что будет подтверждать актуальность используемой модели (структурную адекватность обоих двойников).

В большинстве случаев предлагаемые «примеры DT» представляют собой незамысловатый ребрендинг привычных (обычных) систем, т.е. скорее являются Pseudo Digital Twin \ Digital Impostor, а не Digital Twin, при этом даже имея все три компонента DT могут содержать модель не адекватную своему физическому близнецу («as-is» vs «as-really-is").

Кроме маскирования под DT обычных **SCADA - систем** и CASE \ BPMS типа ARIS (см. первую часть [DT1]), включая Enterprise Architecture (EA, архитектура предприятия как цифровой двойник предприятия), красивую вывеску «DT» прикручивают к системам:

- ERP, например, dia$par,

- architecture as code, например, dochub.info,

IoT-платформ, например, AWS IoT TwinMaker или **Winnum** (интерактивный цифровой двойник как цифровая тень).

https://habr.com/ru/companies/sberbank/articles/890926/

https://controleng.ru/innovatsii/cifrovye-dvojniki/chto/?ysclid=mcogad9c5p165298762

Современный ИИ пока лишь просто повторяет маркетинговую мантру (чепуху) из ранее прочитанного и поданного ему как «обучение», например, спросите его про тот же, **AWS IoT TwinMaker** и получите ответ, что это DT.

https://flexa.cloud/ru/%D1%87%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-%D0%B0%D0%B9%D0%BE%D1%82/

Путаница с DT возникает часто по двум причинам (подменам). Первая, - это подмена DT классическим мониторингом, что технологического процесса (АСУТП), что бизнес-процесса: технологический конвейер (производственная линия) с мониторингом через SCADA или кредитный конвейер с мониторингом через BAM (Business Activity Monitoring). В случае мониторинга на выходе будут считанные в реальном времени параметры и состояния объектов мониторинга, их метрика, агрегация метрик, аналитика, графики распределений величин, дашборты и т.п. Однако при таком мониторинге самой модели объекта вообще может не быть.

Вторая, – это подмена DT классическим моделированием (BPM\EA). Придуманные и визуализированные модели «as-is», ошибочно выдаваемые за DT, не гарантируют адекватность реальным процессам, реальному объекту, т.е. «as-really-is». Нарисовали модель, организовали обмен от датчиков реального двойника и почему-то считаем, что это и есть DT, хотя нет достаточных оснований (доказательств), что структура и состояния цифрового и физического двойников совпадают.

Непонятно зачем использовать новый термин для старых вещей и технологий. Ни одного подробного примера DT пока не видел, т.е. цифровой двойник - это видимо пока что лишь маркетинговый термин, за которым чаще всего лежит классическое математическое и иное (включая BPM) моделирование. Как и 15 лет назад чтобы увеличить продажи некоторые продавцы клеили на обычный сервер наклейку «cloud computing», точно также как 30 лет назад: Это вам не просто компьютер, а "Компьютер с курсором". Какой смысл, кроме маркетинга, называть старые вещи новыми именами?

Поражают хайп-фразы типа: *Сегодня Digital Twin — это не просто модное словосочетание, а ключевой инструмент оптимизации производства.*

https://habr.com/ru/companies/ibs/articles/897072/

Только к подобным статьям вопросы в комментариях остаются без ответов, а вместо «ключевой инструмент оптимизации производства» лучше использовать «ключевой инструмент маркетинга».

Вокруг DT вертятся также модные и непонятные «знаки внимания»: «digital enterprise» (предприятие с автоматизации 1?), Архитектор 2.0, **Enterprise 3.0** (Smart Space), Индустрия 4.0 и т.п.

https://vc.ru/future/1025126-enterprise-30

Однако сам концепт DT все же имеет основание и потенциал реализации. Ниже для более широкого «охвата проблемы» DT будем рассматривать различные сферы его обсуждений (термин «применений» - пока преждевременен): изделия – вещи (например, табуретка), автоматизация бизнес-процессов (BPMN) и Process Mining (**Манифест**), управление технологическими процессами (SCADA) и IoT и др. Странно: обсуждений DT – много, но вот простой и «человеческий» пример «DT табуретки» не видел, только «безотказный» ИИ **готов рассказать о нем**.

https://www.tf-pm.org/upload/1590128200840.pdf

https://github.com/bpmbpm/doc/blob/main/IT/software/DT/AI/stool.md

Ниже показаны некоторые технологии, которые на основе своей «инструментальности» позволяют говорить о DT с гарантированной адекватностью модели (DT) и физического близнеца.

1 Базовые условия DT

Про достаточное условие в определении DT сложно сказать, но про необходимое условие ясность есть. DT - это модель, которая (два базовых условия):

А) должна быть Гарантировано адекватна своему физику (реальности). Полагаю, что это достигается только инструментальными средствами, как минимум инструментами тестирования \ верификации, а как максимум генерации «физика» на основе модели или воспроизведения - восстановления модели из реального близнеца (mining, например, process mining);

Б) должна обмениваться информацией с «физиком», т.е. оба двойника (цифровой и физический) должны иметь возможность синхронизации. Просто «напечатать» изделие на 3D принтере – этого мало, нужна еще его синхронизация в процессе эксплуатации, а при изменении структуры изделия (например, сломалась ножка напечатанной табуретки) – передача в модель (DT) этой информации. Синхронизация в процессе эксплуатации может предусматривать как обмен (односторонней или двухсторонний) «по данным», так и обмен информацией «по структуре» для подтверждения структурной идентичности обоих двойников.

Адекватность должна быть подтверждённой «средствами объективного контроля». Синхронизации «по структуре» в предельном случае – это и есть генерация \ восстановление одного двойника из другого. Синхронизация «до данным» - это обмен информацией о состоянии \ конфигурации, параметрах (среды, процессов): обычно от «физика» к модели, но может быть и наоборот – для загрузки новой конфигурации в физический двойник. Классический пример синхронизация «по данным» - это мониторинг, например, средствами BAM, SCADA. Примером получения и обмена информацией «по структуре» может быть получение, передача и анализ рентгеновского снимка, данных от 3D-сканера и т.п.

Возможности анализа (использование цифрового двойника), например, поведения DT в моделируемых условиях, - это уже следующий шаг и на базовые условия определения DT не влияет.

Много придумано разных аббревиатур моделирования, от CASE (где S = system или software – не важно) и BPMS (где M - что modeling, что mgmt. – без разницы) до MBSE, MDSD, MDA, MDD и т.п. Следуя определению DT, любая исполняемая модель уже удовлетворяет базовому (она уже адекватна «физику», раз его породила), но ещё не достаточному условию.

1.1 «As is» vs «as-really-is»

С одной стороны, мы можем нарисовать (более веско звучит «смоделировать») очень детальную (абсолютно подробную) планировку помещения \ здания (СПДС, BIM) или схему процесса (EPC, BPMN) и наивно считать, что это «адекватная модель объекта». Кроме классических типов моделей «as is» & «to be» есть еще «as-really-is», которая отражает «как в реальности есть на самом деле» и она (объективная реальность) часто сильно отличается от «красивой» кем-то нарисованной схемы «as is» (субъективного взгляда).

Просто «моделирование», например, процессов предприятия в современной BPM-системе (т.е. ARIS образца 1992 года и подобные) позволяет делать только «as-is», а не гарантированный «as-really-is». Какие бы слова при этом не говорились: «полно-полное» описание компании (предприятия) начиная с онтологии и заканчивая «мелко-мелкими» физическими процессами производства \ административными операциями (бизнес-процессами), включая, мельчайшие детали («винтики»): детальные алгоритмы и правила, роли и ИТ-компоненты (ИТ-архитектура, CMDB и т.п.). Поэтому классическое моделирование будет противопоставлено инструментальному моделированию. «Инструментальное» не в смысле использования ARIS и подобных инструментов, а в смысле получение модели (или реального двойника) «автоматически».

«Все врут! или казуистика описания бизнес-процессов»:

Совет — при любой возможности приоритет отдавать данным, полученным инструментальным путем, а не субъективной оценки заинтересованных лиц.

https://habr.com/ru/articles/226689/

DT должен быть основан на «as-really-is» = «как есть на самом деле», а не «as is» = «как это видит субъективный взгляд», в том числе коллективный, т.е. группы архитекторов, разработчиков, пользователей и т.п.

В этом смысле можно согласиться с утверждением, что «DT не является моделью», см. **Подробное объяснение определения DT.** В том смысле, что DT это больше чем «просто модель», например, модель должна быть адекватная, например, исполняемая или автогенерируемая, или хотя бы верифицируемая инструментально (не ментально, не субъективно).

https://www.se-rwth.de/essay/Digital-Twin-Definition/

2 Инструментальный подход к DT

Инструментальный способ - это без участия человека. Имеем две разные технологии: модель -> реальность и реальность -> модель. DB обоих случаях «->» обозначает автоматический \ инструментальный переход. Такой (инструментальный) подход позволяет исключить как человеческий фактор и автоматизировать саму процедуру создания одного из двойников или их синхронизацию по структуре (синхронизация по данным – это скорее мониторинг) \ верификацию «структурной идентичности».

«Модель -> реальность» - например, исполняемая модель bpmn (что нарисовал, то и исполнил), т. е. модель формирует реальность (как и 3D печать). Второе направление (реальность -> модель) – это, например, process mining или «архитектура как код» (например, «боевой» репозитарий + «диаграмма как код»), когда реальные конфигурации автоматически формируют модель. Другой пример, - раскрытие сети lan/ wan (HP OpenView Network Node Manager и подобные).

Примеры инструментальных подходов:

- Модель -> Физический объект: 3D принтер, WFE (BPMN engine);

- Физический объект -> Модель: 3D сканер, Process mining (восстановление логики процесса по логам систем, исполняющих процесс).

Пример не инструментальных подходов:

- Модель -> Физический объект: разработать вручную комплект чертежей изделия (ЕСКД) и отдать на производство;

- Физический объект -> Модель: нарисовать схему процесса \ архитектуры в редакторе BPM \ EA (Aris \ Archi и т.п.).

Если на входе модель (DT), а на выходе физический двойник, то «модель порождает» (прямое проектирование). Если наоборот, то «физик порождает» DT (обратное проектирование). Когда DT создан не инструментальным путем, то обязательно должна быть **инструментальная** верификация модели, например, через тот же Process mining.

Примеры прототипов (DT пока еще как бы нет) инструментальных двойников «прямого цикла» для вещи (stool twin) и процесса (process twin):

а) 3D-модель (CAD) (далее компиляция, например, G-code) –> среда изготовления (станок ЧПУ \ 3D-принтер) –> физический экземпляр (предмет, например, табурет-монолит);

б) Схема процесса в BPMN моделере (далее компиляция, например, в BPMN XML) -> среда исполнения (WF Engine) -> исполняемый экземпляр процесса (приложение \ программа).

Развёртывание модели в физический объект («цифровая модель – прототип изделия») может называться: авто создание объекта через «цифровой мастер» (Digital master), Model Based конструктор и т.п.

Другими слова, имеем прямое \ обратное проектирование – как **Pre-Manufacture \ Post-Existing** «Digital Twin»: зеркало Pre-Digital Twin в лице физического объекта («физик») и зеркальное отражение «физика» в цифре.

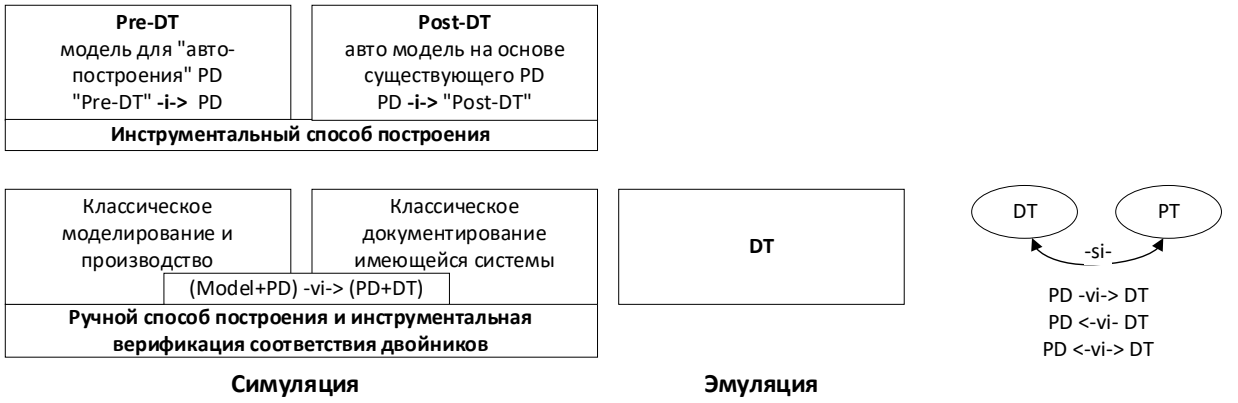
https://www.basen.net/two-types-of-digital-twins-post-existing-and-pre-manufacturing/

Как в философии: Что первично Идея (модель, концепт) или Материя (изделие, экземпляр)?

Одно из базовых условий DT– это адекватность модели, определяемая в том числе степенью детализации, т.е. уровень детализации должен быть достаточен.

Основное внимание выше было уделено идентификации DT и принципам построения путем выделения особого класса «Инструментальный DT» (Pre-Manufacture \ Post-Existing «Digital Twin»). Использование DT, включая предсказание поведения, симуляцию, тестирование и т.п. – это отдельное тема.

На рис. 1 а) показаны основные направления реализации DT инструментальным способом.



а) Типы DT (по инструментальности)

б) Варианты синхронизации PT и DT

Обозначения:
DT – (digital twin) цифровой двойник
PD - (physical twin) физический двойник
-i-> инструментальный способ построения (instrumental)
-vi-> инструментальный способ верификации (verify)
-si- инструментальная синхронизация по данным между DT и PT (обычно только от PT к DT)

Рис. 1 Основные направления реализации DT инструментальным способом

Когда модель (DT) автоматически строится на основании реальных конфигов систем – это тоже вариант реализации инструментального подхода (например, architecture as code), но конечно в конечной системе должны быть соблюдены оба базовых условия полностью.

В какой-то момент модель (DT) может рас синхронизироваться со второй половиной, при этом система контроля должна выдать сообщение «DT приболел, нужно подлечить». В идеале конечно предпочтительнее самосинхронизирующиеся модели.

3 MDA\MDD

В понятие «MDA\MDD» (Model Driven Architecture \ Model Driven Design \ Model Driven Development) иногда вкладывается разный смысл и много схожих терминов, типа MDE, MBE, MBSE, MDSD, CASE и т.п., тут мы имеем ввиду «исполняемый MDD» (executable model), т.е. когда реализация (код) **автоматически генерируется из моделей**.

https://modeling-languages.com/clarifying-concepts-mbe-vs-mde-vs-mdd-vs-mda/

Небольшую хронологию инструментов executable model **показал ИИ**, список исполняемых инструментов UML (Executable UML 2017) см. на **modeling-languages.com**.

https://github.com/bpmbpm/doc/blob/main/visualization/executable_model/mda_mdd_history.md

https://modeling-languages.com/list-of-executable-uml-tools/

Однако подобных инструментов намного больше, назовем пару отечественных: из «классики» (UML) – **Flexberry**, а Runa WFE [WFE24] поддерживает две нотации (**BPMN и AD UML**).

https://samag.ru/archive/article/3739

https://runawfe.ru/NotationsComparison
Современные low code / no code имеют тут же концепцию MDA\MDD (программирование без программирования): если ранее для этого использовали исполняемый UML (Rational Rose), то сейчас это в основном исполняемый BPMN (BPMN-engine).

На тему MDA\MDD недавно высказался автор плагина Rational Rose для инструмента Borland, Delphi – назвав его «анти-шаблоном» (см. **Episode transcript**).
https://www.thoughtworks.com/insights/podcasts/technology-podcasts/architecture-as-code

Полагаю, что OMG забросила развитие UML, переключившись с него на BPMN ввиду бесперспективности UML в части MDA\MDD и больших в этом ожиданий от BPMN-engine.

Вроде бы верное направление развития «BPMN – engine» - это все-таки шаг вперед. Однако все равно это технология для построения DT (модели процесса) еще недостаточна. Да и OMG похоже также как когда-то к UML потеряла интерес к развитию исполняемого BPMN. Качество, точнее «зрелость», современных технологий характеризует их совместимость, что low-code noBPMN, что только исполняемого BPMN (совместимость BPMN-engine): нет совместимости и похоже, что обеспечить ее нереально, см. **Оценку совместимости платформ low-code**.

https://modeling-languages.com/interoperability-low-code-tools/

Таким образом, инструментальный **переход** «от модели объекта к реальному объекту» (исполняемому коду) в программировании – это генерация кода на основе модели: что древний MDA\MDD на UML, что современный его «перепев» на BPMN или собственной нотации (noBPMN).

«MDA\MDD из 2025» позволяет говорить о прототипе DT: «цифровой двойник программы» (приложения) или, если ПО реализует бизнес-процесс (workflow), то о «цифровом двойнике процесса». В нем уже штатно реализуются подсистемы управления экземплярами \ мониторинг состоянием процессов (cockpit) и др. Обобщение по cockpit см. Definition: Digital Twin Cockpit (DTC)

https://www.se-rwth.de/essay/Digital-Twin-Definition/

Исполняемость модели позволяет инструментальный переход от модели объекта (образу) к физическому экземпляру объекта (исполняемому коду). Технология WFE (WF Engine – общий случай BPMN-engine) показана на простом примере «Hello Calculator» [WFE24]. Современные технологии WFE (в первую очередь на BPMN, т.е. BPMN engine) позволяют создание DT только в совсем простых случаях \ примерах, кроме того они еще не умеют моделировать данные для передачи в BPMN engine.

3 Цифровой тройник

Если в системе близнецов реализованы одновременно инструментальные Pre и Post подходы, то можно говорить о **«цифровом тройнике»**, например, если одновременно применить прямое (Pre-DT, где нет еще «физика») и обратное (Post-DT, анализ существующего «физика») проектирование,

то получим Цифрового Тройника (ЦТ). Варианты ЦТ: модель –> физик –> модель' или физик –> модель –> физик'.

Этапы «ЦТ первого типа»:

1) Pre-Digital Twin. подготовка исходной модели – прототипа, создание модели (project / model);

2) Physical Twin. Адекватный модели физический объект, который получен через развертывание (deploy), т.е. развертывание модели в физический объект (автоматическое изготовление по модели);

3) Обновление модели цифрового двойника: сканирование или исследование цифровых «следов и теней» (model mining), образованных функционированием объекта. Реализация связей «Связность - структура» и «Связность – монитор». На выходе: новая модель (модель №2), воспроизведенная «по следам» физического двойника;

4) Верификация исходной модели и модели №2. Если «физик» был изготовлен из модели №1 с дефектом (дефект производства), то сравнение двух моделей (1 и 2) это выявит.

В данном случае, имеется два инструментальных перехода: «модель-объект» и «объект-модель», т.е. формула «модель-объект-модель'» - как замкнутый инструментальный преобразователь или как система полного (замкнутого) цикла.

Термин «нет еще «физика» - условный, т.е. может происходить «трансформация текущего, существующего» по данным модели. Например, конфигурационные файлы из DT загружаются в сеть физических сетевых коммутаторов и маршрутизаторов, и физическая сеть перестраивается (фактически возникает новый сетевой сегмент, например, новые VLAN).

4 Ограничения «DT-образных» технологий

Есть прямой и обратный инжиниринг (проектирование). Мы рассматриваем автоматизированный (инструментальный) процесс инжиниринга. Прямой инструментальный инжиниринг – это «Инструментальный способ построения» изделия (что табуретки, что программы, что предприятия) по чертежам (моделям). Что 3D-принтер, что MDA\MDD.

Обратный инструментальный инжиниринг – это инструментальный анализ (сканировании и т.п.) существующего изделия и построение его модели. Термины «немного натянуты», например, «одни ИИ» считают Process Mining технологией обратного инжиниринга, другие – нет.

Активные и пассивные методы. Пассивные методы обратного инжиниринга и построения модели объекта – это, например, методы анализа цифровых теней \ следов, включая Process mining \ Task Mining. **Digital shadow vs digital footprint**
https://github.com/bpmbpm/doc/blob/main/IT/software/DT/AI/digital_shadow_digital_footprint.md

Для следов и теней должно иметь место «динамическое свойство», т.е. некая история «жизнедеятельности» объекта. Например, результат 3D-сканирования изделия (табуретки) – это не цифровые тень или след, а скорее итоговая цифровая копия / модель, которая уже была составлена по отдельным артефактам, которые условно можно назвать «цифровыми тенями» (синтез структуры на основе обработки «теней»). Кроме того, анализ следов и теней – это пассивные методы, а сканирование – активный.

Иногда вводят "**Development Digital Twin**" как противопоставление "Usage Digital Twin": в первом случае – создали двойника и «забыли», во втором - двойник предполагает взаимодействие с близнецом в процессе эксплуатации. "Development Digital Twin" – назвать двойником

проблематично, т.к. это всего лишь этап создания, т.е. мы предполагаем все же наличие DT на стадии эксплуатации (в связке со вторым близнецом), а этап создания DT (цифровой копии \ модели) или физического близнеца (по Pre-DT) - это только первый этап, за которым следует эксплуатация DT. Второе базовое условие (синхронизация в процессе эксплуатации) косвенно подчеркивает динамические свойства связки обоих двойников (динамику, взаимодействие).

https://www.se-rwth.de/essay/Digital-Twin-Definition/

Активные методы обратного инжиниринга и построения модели объекта – это, например, сканирование объекта, например, раскрытие вычислительной сети через ping, SNMP, WMI и т.п., т.е. это больше чем формирование сетевой топологии.

Активные и пассивные могут быть использованы как при построении модели объекта, так и при верификации и синхронизации (по данным и по структуре) модели с «физиком». Пример информации «по данным»: температура каждой ножки табуретки; пример информации «по структуре»: отсутствие одной ножки, трещина или деформация в ней.

Инструментальное развертывание (deploy), контроль – верификация как элементы DT пока ограничены и не позволяют «в лоб» формировать двойников. Тот же Process Mining даже при наличии подробного логирования («вход» для процедуры Process Mining) не гарантирует адекватность сгенерированной модели, например, даже при большом числе отработанных экземплярах есть ненулевая вероятность, что по некоторым возможным маршрутам ни один экземпляр так и не отработал за время формирования лога (для анализа нужно больше вариаций).

Если «вытащить» алгоритм из кода путем обратного проектирования (инжиниринга) – большая проблема, то «вытащить» из мозга исполнителя алгоритм, «который он действительно будет применять» – задача еще сложнее (фактически только пост-контроль). Даже при наличии описания алгоритма исполнитель может его не знать (не прочитать или прочитать, но не понять) или действовать со злым умыслом по иному алгоритму. Вообще DT с участием человека в процессе (моделирование поведения) – это отдельная тема, где некоторые процессы будут обозначены как «черные ящики» с непонятной логикой внутри.

Применительно к процессам (бизнес-процессам, технологическим и т.п.) – в основе модели (DT) лежит алгоритм действий. В человеко-машинной среде они распределены в «памяти» человека – исполнителя и в памяти машины, и только при 100% автоматизации (автоматическое устройство) – только в памяти машины. Каким-либо образом вводить в модель (DT) алгоритмы для человека – исполнителя и прогнозировать выполнение их на практике – сложно (по сути будет DT человека). Поэтому чаще разговор о DT возникает, когда имеется достаточно высокий уровень автоматизации, например, в SCADA- системах.

Таким образом, выше показаны только **направления** инструментальных подходов для применения в DT, но нет еще готового framework, гарантирующего реализацию DT на базе стандартных существующих технологий.

5 Эмулятор

С эмуляторами проще и это тоже отдельный вид DT. Рассмотрим «DT сети».

Допустим есть физическая сеть на устройствах Cisco. Развернули рядом стенд на Dynamips (**Cisco router emulator**). Все конфиги, таблицы маршрутизации и т п. копируются в реальном времени из «физика» в цифровой двойник (сеть на Dynamips).

https://habr.com/ru/articles/494504/

В момент времени t+1 мы можем на стенд подать пакет и увидеть, как он отработает. Т.е. увидеть, как отработан был бы пакет в реальной сети (на устройствах Cisco) на этот момент t+1 при условии синхронизации обоих двойников (Cisco и Dynamips) на время t.

Если "по-хорошему", то копирование конфигов должно быть не «узел – узел» (напрямую), а через промежуточный архитектурный репозитарий (история, аналитика и т п.). Плюс визуализация (и не только сетевых соединений) и т.п., т.е. полноценный разносторонний контроль, верификация, синхронизация. Раскрытие сети в обоих случаях покажет одинаковый результат.

Ограничение такого DT: производительность, надёжность и т п., но это нормально для DT (модели). В DT важна адекватность модели, а тут полная эмуляция на dynamips (работал с dynamips в середины 2000-х, тогда он идеально эмулировали парк Cisco).

На рис. 1 а) показаны две группы «Симуляция» и «Эмуляция». Примеры эмуляторов: Android на windows или Cisco iOS на Windows \ Linux (Dynamips). Они реально исполняют целевой процесс, но в другой среде, они могут выступать в качестве резервного контура.

Пример симуляторов: Формула 1 под DOS, промышленный авиа-тренажер.

Заключение

Напомним, что четкого «checklist» для подтверждения статуса «DT» еще нет и большинство терминов по DT имеют маркетинговый характер (задача продать старое под новой этикеткой).

Поэтому на протяжении последней пятилетки наряду с ростом интереса к теме DT мы имеем в основном примеры не DT, а псевдо DT, что дискредитирует саму идею DT. Зачем назвать старое новым термином в угоду хайпа?

Однако из запросов **про «checklist» у ИИ** уже можно «вытянуть клещами» требования про инструментальные методы создания моделей (см. пункт 7 ответа ИИ).

https://github.com/bpmbpm/doc/blob/main/IT/software/DT/AI/checklist.md

Наряду с формулированием двух базовых необходимых, но недостаточных условий (требований) к DT (гарантированная адекватность модели и наличие синхронизации в процессе эксплуатации и собственно сам этап эксплуатации) был предложен термин Инструментальный DT. Приставка «инструментальный» определяет использование технологий, гарантирующих адекватность модели и устраняющий субъективный фактор при создании модели (DT – это модель).

Выше были показаны базовые технологии, которые могут быть применены для построения «Цифрового двойника Инструментального типа» с гарантией адекватности модели ее физическому двойнику, т.к. один «порожден» (развернут, напечатан) из другого и верифицирован средствами объективного контроля (инструментального, т.е. не субъективного контроля).

Классические системы \ подходы к моделированию процессов и архитектур (ARIS и т.п.) конечно же не являются базовыми для DT и не позволяют строить ни DT процессов, ни DT предприятий, т.к. это обычное «субъективное» моделирование «as is», а далеко не «as-really-is». Не являются базовыми и классические системы имитационного моделирования, т.к. это тоже «субъективное» моделирование. Кроме того, в дополнение к принципу «as-really-is» должна быть хорошая визуализация этой модели (схемы, графики). «As-really-is» - как показано в модели, так и

исполнится в реальности. Можно нарисовать «тонны схем» в Арис (включая ARIS Simulation), но после запуска Process Mining убедиться, что все нарисованное есть не что иное как макулатура, т.к. не соответствует действительности (по многим причинам). Цифры экономии через Process Mining **тут**.

https://www.comnews.ru/content/239962/2025-07-07/2025-w28/1013/centrobezhnaya-sila-kak-sber-prinimaet-resheniya-o-vyvode-vnutrennikh-it-razrabotok-kommercheskiy-rynok

Выделим инструментальные технологии «прямого проектирования» DT: 3D-печать, executable MDA/MDD (executable model в разных нотациях: UML, BPMN, проприетарные).

Технологии «обратного проектирования» DT:

а) активные: сканирование, включая, 3D-scan, раскрытие сети (LAN\WAN) и т.п.

б) пассивные: Process mining (цифровые тени и следы).

Третья технология DT – эмуляторы (например, сеть на Dynamips) с синхронизацией с «физиком», т.е. в связке с объектом эмуляции.

Однако указанные базовые технологии «из коробки» пока не обеспечивают построение DT, т.е. им необходимы дополнения.

Рассмотрение DT велось в широком диапазоне: от DT табуретки – или иного простого изделия типа «вещь», где хорошо «вписываются» 3D-print / 3D-scan, далее DT программы (software), бизнес-процесса, промышленного \ кредитного конвейера и даже DT предприятия (офиса \ завода).

Для идентификации DT как минимум должны быть представлены следующие артефакты: схемы структур обоих двойников, доказательства адекватности модели своему физическому двойнику, описание синхронизации между двумя двойниками. Без указанных подробностей заявления о создании «изделия – DT» (включая модель предприятия) должны считаться не обоснованными.

Ссылки

[DT1] Digital Twin. Часть 1. Цифровой двойник vs цифровой самозванец

https://habr.com/ru/articles/877454/

[WFE24] Исполняемый BPMN в Open Source Runa WFE (WfMS). Hello Calculator и немного классификации

https://habr.com/ru/articles/866822/

Как СберМобайл завод оцифровал, и кому это вообще нужно

https://habr.com/ru/companies/sberbank/articles/890926/

Манифест Process Mining

https://www.tf-pm.org/upload/1590128200840.pdf

Process Mining. «Рентгеновская диагностика» бизнеса

https://habr.com/ru/companies/T1Holding/articles/756238/