Towards Model-Driven Middleware Maintenance

Jörg P. Wadsack[1]
Department of Computer Science
University of Paderborn
Warburger Str. 100
33098 Paderborn
Germany
maroc@upb.de

Jens H. Jahnke
Department of Computer Science
University of Victoria
PO Box 3055
Victoria B.C.
Canada V8W3P6
jens@cs.uvic.ca

Abstract

Over the last decades, network-centric information management has tremendously changed the business processes of organizations in the private and public sector. The introduction of middleware technology and standards has played an instrumental role in this development by enabling integration of distributed heterogeneous information systems (IS). Modern (so-called third-generation) net-centric systems use middleware among various types of software components ranging from mainframe computers down to PDAs and cell phones. The maintenance and evolution of such middleware software is a key challenge because of rapidly evolving hardware and software platforms. In this paper, we outline a systematic process for engineering and maintaining middleware solutions for third generation systems; this process is based on the Model-Driven Architecture™ approach of the Object Management Group.

Keywords:

Software integration and maintenance, reengineering, model-driven engineering, middleware, network-centric systems, software generation

1. Middleware in Net-Centric Organizations

Today's requirements for distributed heterogeneous systems have to consider large amounts of information spread over multiple locations and platforms. Third generation network-centric systems need middleware

within corporate organisations,

among corporate organisations and

between corporate IS and mobile, embedded smart devices.

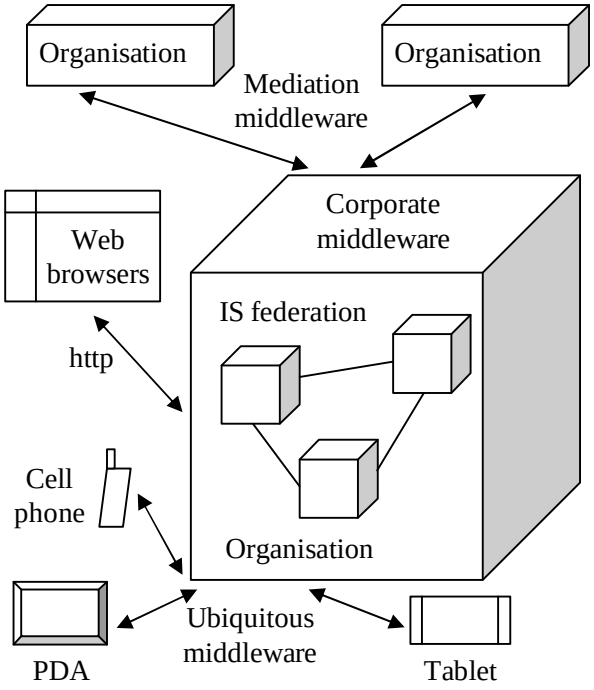

Figure 1 Third generation middleware

Those three middleware types are depicted in Figure 1. We call middleware for federating IS within organisations *corporate middleware*. The middleware required for mediating different organisations is called *mediation middleware*. Finally, *ubiquitous middleware* denotes middleware that integrates organisational IS to "every-day" embedded devices.

Today more than ever before, industrial organizations are constantly evolving. Since modern organizations largely depend on IT infrastructures, there is a strong requirement that these infrastructures (middleware) facilitate this evolution.

[1] Jörg P. Wadsack was a visiting researcher with the NET-lab group at the University of Victoria from January to April 2002.

2. Model-Driven Middleware Maintenance

Currently model-driven approaches like "Model-driven software development", "Model-driven engineering" or "OMG's Model-Driven Architecture™" are discussed in the software engineering community. Another (re)emerging topic is model-driven "Information System Integration". How are model-driven approaches and IS integration related? Integration of IS generally takes place by means of some sort of middleware, i.e. modeling an architecture to realize interoperability of the IS to integrate. Model-driven middleware maintenance is the combination of middleware architecture and model-driven software development.

2.1 Model-Driven Architecture™

The OMG Model-Driven Architecture™ (MDA) is a general approach of the OMG for building distributed heterogeneous systems. MDA is build upon the Unified Modeling Language™ (UML), the Meta-Object Facility™ (MOF) and the Common Warehouse Meta-model™ (CWM), which are accepted modeling standards [MDA01].

The core idea of MDA is a process model to unify the analysis and the design of distributed heterogeneous systems. To facilitate this, MDA separates structure and function of a system from its technical realization. The process is anchored on two models, namely the "Platform Independent Model" (PIM) and the "Platform Specific Model" (PSM). Figure 2 shows an overview of the MDA process. The PIM is separated in two sub-models, one for the business logic (computational independent) and one for the component view (computational dependent) of the system.

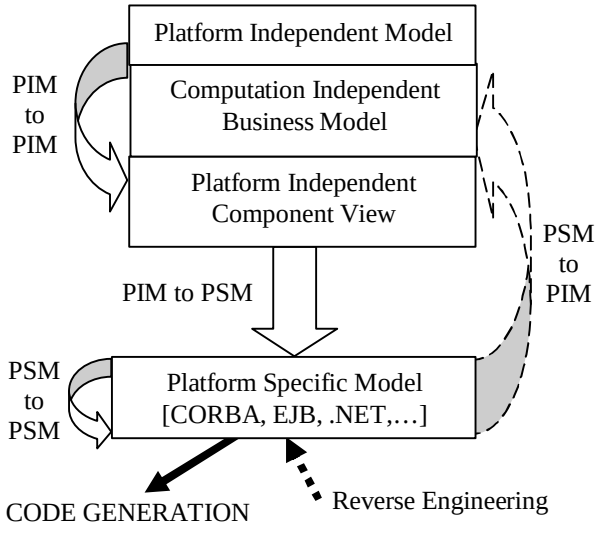

Figure 2 MDA process

The mappings in a model or between the two models are described in [MDA01]:

> *PIM to PIM* mappings are model refinements during the development lifecycle that do not need any platform dependent information. Those transformations also relate the business models and the component views. They build the bridge between requirements, analysis and design.
>
> *PIM to PSM* mappings are performed once the PIM is elaborated enough to be associated to the characteristics of the chosen platform. It is a projection to the execution infrastructure of the platform. The projection from a conceptual component view model to existing specific commercial middleware platforms like CCM for CORBA [COR01] or EJB for J2EE [GOS+01].
>
> *PSM to PSM* mappings are model refinements during the realization and deployment of components. An example for PSM to PSM transformation is the selection of services and preparation of their configuration.
>
> *PSM to PIM* mappings are model reverse engineering operations. Those transformations are needed to build abstract models from existing implementation of specific middleware technologies. Those model transformations are part of a "mining" process, which can hardly be fully automated.

The *code generation* can be done by commercially available UML tools depending on the chosen platform and specific technology. Unfortunately, the *reverse engineering* aspect is almost left open by the MDA. Here, the OMG recommends the use of adequate reverse engineering tools, but these are hardly available to date [MJS+00].

The forward engineering aspects of MDA are well described in literature and supported by existing CASE tools. The reverse engineering issue and the "mining" process of the PSM to PIM mapping are only described vaguely in the MDA related documents. However, these aspects are of crucial importance for middleware and mediation technologies, which have to deal with pre-existing IS legacies. The following sections approach this problem in the general framework of the OMG Model-Driven Architecture™.

2.2 Model-Driven Middleware Maintenance Process

The Model-driven Middleware Maintenance (MMM) process outlined in this section deals with existing legacy distributed heterogeneous systems, which have to be integrated, renovated, redesigned, extended, etc. The MMM process is depicted in Figure 3.

The start of the process is a recovery of the different components of the distributed heterogeneous system. Second, we investigate the relationships and dependencies between these component information models. We call these intersections "join points". Third, a classification of components or component groups is made into the three middleware types introduced earlier, namely, corporate middleware, mediation middleware, and ubiquitous middleware. Finally, the generation of the middleware is performed.

The first activity in the MMM process is to reverse engineer the existing component information models and to define new ones if requested. Due to our experience, this task generally takes several iterations because the information that has to be collected is often obsolete and spread over the whole system. Moreover the reverse engineering task has to deal with uncertain assumptions. Therefore, only an interactive process makes sense because expert's knowledge is required.

Comparing this task to MDA, defining component information models and iterate them correspond to the PIM to PIM transformations. The reverse engineering steps correspond to PSM to PIM followed by PIM to PIM mappings, cf. Figure 2. For simplicity, we will write only PSM to PIM mappings instead of reverse engineering and PSM to PIM mappings, but a preliminary reverse engineering task is always needed to get a PSM.

Once all component information models are identified and completed, the definition of how they interoperate is needed. This task is similar to the first one and involves the reverse engineering of relationships and dependencies between component information models to be integrated. Again (only) a semi-automatic, iterative approach is feasible.

Those steps of the task are PIM to PIM transformations, which are triggered by PSM to PIM transformations. Defining join points are PIM to PIM mappings. Reverse engineering join points are PIM to PIM transformations, which are triggered by PSM to PIM mappings, i.e. the iterative revealing of dependencies between existing component information models.

The two first activities are part of a semi-automatic, iterative process. Such a process should be supported by tools that execute the tasks that can be automated, reflect intermediate results to the users and provide consistency control during process iterations.

The next activity is to classify the relationships among the component information models along with their join points into one of the three middleware types.

The join-points between two components should be classified as "organizational" if they require federation within the same organization. From a users perspective such components look like a single system. Corporate middleware is used for integrating such components.

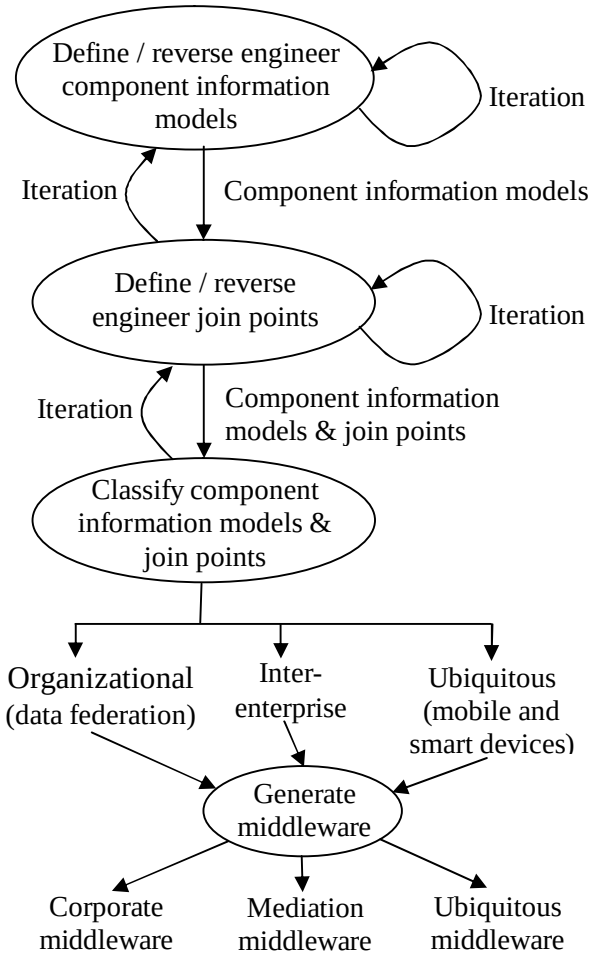

Figure 3 MMM process

Second, join-points between components can be classified "inter-enterprise". In this case, the components retain a large amount of their autonomy, i.e. information is typically replicated, duplicated and synchronized according to certain interoperability policies.

Third, some component information models might reside on mobile and embedded devices. Join-points of such embedded components with organizational IS components are classified as "ubiquitous".

Looking at the model mappings of MDA, the classification into the three middleware types are PIM to PIM and PIM to PSM transformations. The decision which type of middleware the component information model will be attributed is anchored (1) on the business

logic (PIM to PIM) and (2) on the existing infrastructure of (some or all) components (PIM to PSM).

From each activity, backward iterations to the precedent process activities are needed because of misinterpreted, incomplete or even erroneous intermediate results. Summarizing, the three first tasks of the MMM process are done in an iteration loop.

After classifying all component information models with the adequate join points, tools generate the software for the middleware and the join points of the distributed heterogeneous system. We distinguish between two different kinds of join point generation:

middleware for join points in the same category

portals for join points in different categories

Each middleware type (corporate, mediation, ubiquitous) generation includes both join point generation kinds.

In case of corporate middleware, our process encompasses the generation of a federated access layer that supports open nested transactions on the integrated component IS. In addition, interfaces (portals) for the two other middleware types are generated.

The deployment of mediation middleware involves the generation of information import/export portals based on wrappers to access the organisational IS components.

Finally, ubiquitous middleware is generated for implementing join point among organizational and ubiquitous (mobile) information models.

3. Tool support

The process described in the previous section requires tools for several aspects. The tool representing the different component information models and defining join points is called *Fujaba Middleware Toolkit* (FMT). FMT is based on the UML tool *Fujaba* [Fujaba] and supports PIM to PIM transformations, such that component information models can be defined. Figure 4 shows an overview of the tool support.

Varlet/Babel [JB01a, JB01b] enables the analysis and reverse engineering of the structure of data repositories and their PSM representation in (E)ER. The interactive reverse engineering process is based on semantic pattern detection of SQL code as described in [JSZ97]. The *microSynergy* tool [Jah01, DJ01] provides information models for ubiquitous components. *Reddmom* [Reddmom] imports the component information models and supports PSM to PIM mapping for the component information models. Furthermore the import of PSM representations in XML of other component information models, e.g. of semi-structured data models, from other tools is incorporated in Reddmom.

The definition and design of join points are part of the PIM to PIM transformation supported by FMT. The revealing of join points is part of Reddmom, which is also based on Fujaba. This join point reverse engineering functionality is based on a join point detection pattern catalogue and the pattern recognition mechanism [NSW+02] provided by Fujaba. This pattern recognition mechanism is based on the analysis mechanism described by Jahnke et al. [JSZ97] and infers the detection from an abstract syntax graph. Details of this part of the MMM process tool support are described in [WNGJ02].

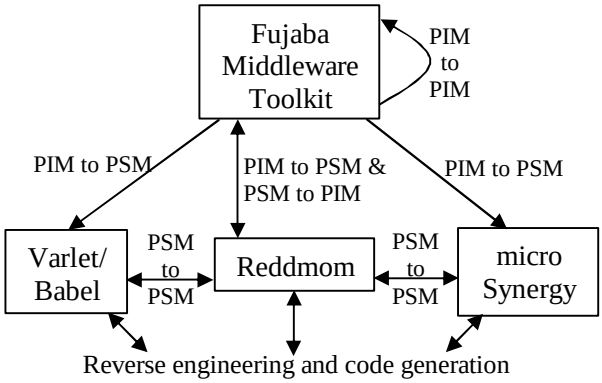

Figure 4 Tool support

The classification of the component information models in the three different middleware types has to be done by the engineer using FMT. The middleware generation of the different types is split to the three other tools.

The corporate middleware generation is integrated in Reddmom. Reddmom supports the generation of an object-oriented transactional component middleware in Java. The basis of corporate middleware generation in Reddmom is the mapping between each data repository structure and the conceptual architectural view of the data federation. Based upon the mapping, interfaces to the different data repositories are generated. A transaction can span several data repositories and transaction may be nested. The transaction nesting is supported by the object-oriented transaction management mechanism. The provided transaction management mechanism is based on four patterns proposed by Grand [Gra99].

To support ACID property for transactions, no operation affecting a data repository can be performed without a transaction object and transaction mechanisms from the data repositories are used if present. In addition, the data repository interfaces store all information valid before submitting the transaction and all information subject to be changed by the transaction. Some data repositories,

mainly databases, support a success check for transaction before committing. This support is used when it is available.

For the generation of e.g. .NET, CORBA or EJB middleware the corresponding UML representation of the component information models can de exported to XML and then imported to other tools, which support the desired middleware generation, e.g., Websphere Studio [WebSphere].

For the mediation middleware generation the *Varlet/Babel* tool is used. Valet/Babel can generate XML portals for (ODBC) data sources based on IBM's XML Lightweight Extractor [XLE]. Furthermore, Varlet/Babel can generate data mediation agents between two different XML structures based on their DTDs. If another kind of middleware is requested, the export of the component information models in XML is supported (for the import in third party tools).

The generation of the ubiquitous middleware is done by the *microSynergy* tool. It has been developed in cooperation with Intec Automation Inc., a local company in the area of network-centric embedded systems. The microSynergy tool enables the generation of embedded interoperability portals based on finite state machine models and SOAP. Using microSynergy, the user can select the kind of information (sensoric data, event signals etc.) should be imported and exported to, respectively from the embedded device. MicroSynergy then generates a limited-scale HTTP server on the embedded device that serves as the XML portal. UVic have begun to integrate microSynergy with wireless protocols as well, e.g., Bluetooth.

4. Related Work

Our presented approach is a specific instantiation of the OMG Model-Driven Architecture™ (MDA) process [MDA01]. As presented the MDA is a general approach for the analysis and design of heterogeneous distributed system development. The MMM process focuses on middleware for heterogeneous distributed systems and refines the MDA process, especially the reverse engineering direction.

Approaches and tools for component information model reengineering activities are e.g. DB-Main [EH99] and Varlet [Jah99]. Both tools cover all phases of database reverse engineering from schema recovery up to building a conceptual schema. In Varlet an interactive process to handle uncertainty and inconsistency during recovery of information models (comprising relationships) is based on Generic Fuzzy Reasoning Nets [JSZ97, Jah99] which revert to code and data analysis. DB-Main provides generation of conceptual wrappers, i.e., software layers that interface a database based on the conceptual schema [TCHH99]. Both approaches are restricted to one component information model at a time and lacks flexibility for recovering join points.

An overview of reverse engineering methods and tools that can be further used and adapted for the process presented in this paper is given in [MJS+00]. To our best knowledge the field of recovering join points is poorly explored.

Common solution for IS integration middleware within organisations is *distributed transaction processing* [XA94] as provided by transaction monitors [Hud94, Hal96] and middleware transaction services [JTS99, OTS98]. A more scalable solution is *reliable messaging* [Lew99, Hou98] which results in a reliable asynchronous processing scenario. Liebig and Tai propose an integration of message-oriented transactions and distributed object transaction to middleware mediated transactions [LT01].

Mascolo et al. present a data-sharing middleware for mobile computing, namely xmiddle [MCZE02]. The sharing of XML documents across heterogeneous mobile hosts is provided by xmiddle, allowing on-line and off-line access to data. Replication transparency is abandoned by xmiddle to achieve an acceptable performance and scalability.

5. Conclusions

Middleware is an important part of third-generation net-centric software systems. It is important to facilitate its maintenance and evolution in order to enable organisations to evolve and exploit the newest hard and software platforms. A model-driven middleware maintenance process can support this goal.

We choose a model-driven engineering rather than a round-trip engineering approach because model-driven development guarantees permanent consistency between model and code. Compared to round-trip engineering, productivity increases by more than 30% with a model-driven engineering approach [Softeam].

Acknowledgments

We thank the Advanced Systems Institute of British Columbia (ASI), the National Science and Engineering Research Council (NSERC) and Intec Automation Inc. for their ongoing support and funding of our research. For their work on the tools Reddmom, Varlet/Babel and microSynergy project we thank Yury A. Bychkov, Derek Church, David Dahlem, Mark d'Entremont, Mike Lavendar, Andrew McNeir, Adeniyi Onabajo and Felix Wolf.

References

[COR01] Object Management Group. *The Common Object Request Broker: Architecture and Specification Revision 2.6*. 492 Old Connecticut Path, Framingham, MA 01701, USA. http://www.omg.org/technology/documents/corba_spec_catalog.htm . 2001.

[DJ01] M. D'Entremont and J.H. Jahnke. *microSynergy - Generative Tool Support for Networking Embedded Controllers*. in 3rd Intl. Workshop on Net-Centric Computing (NCC'01). Toronto: ACM Press. May 2001.

[EH99] V. Englebert and J-L. Hainaut. *DB-MAIN: A Next Generation Meta-CASE*. Journal of Information Systems - Special Issue on Meta-CASEs, Vol 24(2), pp 99-112, 1999.

[Fujaba] *From UML to Java And Back Again*. http://www.upb.de/cs/fujaba.

[GOS+01] J. Griffin, D. O'Connor, D. Sarang, K. Gabhart, D. Young, A. Tost, T. McAllister, R. Adatia, M. Juric, T. Osborne, F. Arni, J. Lott, V. Nagarajan, A. Mulder and C. Berry. *Professional EJB*. Wrox Press Ltd., UK. ISBN 1-861005-08-3. July 2001.

[Gra99] M. Grand. *Transaction Patterns A Collection of Four Transaction Related Patterns*. Proceedings of Pattern Languages of Programs. Urbana, IL, USA. August 1999.

[Hal96] C. L. Hall. *Building Client/Server Applications Using TUXEDO*. John Wiley & Sons, Inc., 1996.

[Hou98] P. Houston. *Building Distributed Applications with Message Queuing Middleware.* Microsoft Cooperation, 1998.

[Hud94] E. S. Hudders. *CICS: A Guide to Internal Strucure*. John Wiley & Sons, Inc., 1994.

[Jah99] J.H. Jahnke. *Management of Uncertainty and Inconsistency in Database Reengineering Processes*. PhD thesis, University of Paderborn, Paderborn, Germany, September 1999.

[Jah01] J.H., Jahnke. *Engineering Component-based Net-Centric Systems for Embedded Applications*. in Joint European Software Engineering Conference and Foundations of Software Engineering (ESEC/FSE '01). Vienna, Austria: ACM Press. September 2001.

[JB01a] J.H. Jahnke and Y. Bychkov. *VARLET/BABEL: Toolkit for Net-Centric Legacy Data Integration*. in 3rd Intl. Workshop on Net-Centric Computing (NCC'01). Toronto, Canada: ACM Press. Mai 2001.

[JB01b] J.H. Jahnke and Y. Bychkov. *Interactive Migration of Legacy Databases to Net-Centric Technologies*. in Workshop on Data Reverse Engineering (DRE 2001). Stuttgart, Germany: ACM Press. October 2001.

[JSZ97] J.H. Jahnke, W. Schäfer, and A. Zündorf, *Generic Fuzzy Reasoning Nets as a basis for reverse engineering relational database applications.* Proc. of European Software Engineering Conference (ESEC/FSE), LNCS 1302, Springer Verlag, September 1997.

[JTS99] JTS. *Java Transaction Service (JTS).* Sun Microsystems Inc., December 1999. Version 1.0.

[Lew99] R. Lewis. *Advanced Messaging Applications with MSMQ and MQSeries*. Que, 1999.

[LT01] C. Liebig and S. Tai. *Middleware Mediated Transactions*. Proceedings of the 3rd International Symposium on Distributed Objects & Applications. Rome, Italy. September 2001.

[MCZE02] C. Mascolo, L. Capra, S. Zachariadis and W. Emmerich. *XMIDDLE: A Data-Sharing Middleware for Mobile Computing*. To appear in Int. Journal on Wireless Personal Communications. Kluwer. 2002.

[MDA01] *Model Driven Architecture* (MDA) Edited by Joaquin Miller and Jishnu Mukerji. http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01

[MJS+00] H.A. Müller, J.H. Jahnke, D.B. Smith, M.A. Storey and K. Wong. *Reverse Engineering: a roadmap*. Future of Software Engineering. International Conference on Software Engineering (ICSE), Limerick, Irland. Editor A. Finkelstein ACM Press. June 2000.

[NSW+02] J. Niere, W. Schäfer, J.P. Wadsack, L. Wendehals, and J. Welsh. *Towards Pattern-Based Design Recovery*. Proc. of the 24th International Conference on Software Engineering (ICSE), Orlando, Florida. May 2002.

[OTS98] OTS. *Transaction Service Specification*. Object Management Group, February 1998.

[Reddmom] *ReEngineering of Distributed (federated) Databases for Multimedia Objectoreineted Middleware*. http://www.upb.de/cs/reddmom.

[Softeam] Softeam. *Model driven Engineering (MDE) versus Roundtrip engineering (RTE).* White Paper. http://www.softeam.fr/us/smot_uml_white.htm. 2000.

[TCHH99] P. Thiran, A. Chougrani, J-M. Hick, and J-L. Hainaut. *Generation of Conceptual Wrappers for Legacy Databases*, DEXA'99 conference, Florence, September 1999.

[WebSphere] http://www.ibm.com/websphere.

[WNGJ02] J.P. Wadsack, J. Niere, H. Giese and J.H. Jahnke. *Revealing Data Dependencies in Web Information Systems*. ICSM 2002 Workshop on Database Maintenance and Reengineering (DBMR02), Montréal, Canada, October 2002.

[XA94] XA. *Distributed Transaction processing: The XA+ Specification, version 2*. X/Open Group, 1994. X/ Open Company, ISBN 1-85912-046-6, Reading, UK.

[XLE] http://www.alphaworks.ibm.com/tech/xle.